"This paper is a postprint of a paper submitted to and accepted for publication in IET Networks and is subject to Institution of Engineering and Technology Copyright. The copy of record is available at the IET Digital Library"
IET Networks

ElGamal Cryptosystem-based Secure Authentication System for Cloud-based IoT Applications

Submitted on: 17-02-2019
Submitted by: Tanmoy Maitra, Mohammad Obaidat, Debasis Giri, Subrata Dutta, Keshav Dahal
Keywords: SECURITY, AUTHENTICATION, INTERNET OF THINGS
ElGamal Cryptosystem-based Secure Authentication System for Cloud-based IoT Applications

Tanmoy Maitra*, Mohammad S. Obaidat†, Fellow of IEEE, Debasis Ghimire‡, Subrata Dutta§, Keshav Dahal

† School of Computer Engineering, KIIT, deemed to be University, Bhubaneswar, India
‡ Department of Information Technology, Maulana Abul Kalam Azad University of Technology, Ranchi, Jharkhand, India
§ Department of Computer Science & Engineering, National Institute of Technology, Jamshedpur, India
* E-mail: tanmoy.maitrafcs@kiit.ac.in and msobaidat@gmail.com

Abstract: Life in modern society becomes easier due to rapid growth of different technologies like real-time analytic, ubiquitous wireless communication, commodity sensors, machine learning, and embedded systems. Nowadays, there seems to be a need to merge these technologies in the form of Internet of Things (IoT) so that smart systems can be achieved. On the other hand, cloud computing is a pillar in IoT by which end users get connected through the cloud servers for getting different services. However, to recognize the legitimacy of communicators during communication sessions through insecure channels like the Internet, serious issues in cloud based IoT applications need to be addressed. Thus authentication procedure is highly desirable to remove the unapproved access in IoT applications. This paper presents an ElGamal cryptosystem and biometric information along with a user’s password-based authentication scheme for cloud based IoT applications refereed as SAS-Cloud. Security of the proposed scheme has been analyzed by well popular random oracle model and it is found that SAS-Cloud has ability to defend all the possible attacks. Furthermore, performance of SAS-Cloud has been evaluated and it was found that SAS-Cloud has better efficiency than other existing competing ElGamal cryptosystem-based authentication schemes.

1 Introduction

In modern society, connection with everyone and everything through Internet enabled electronic devices has become common for smart living [1]. To facilitate this, network research community has been trying to develop such systems so that efficient and reliable communication can be done from remote places. This networking system is known as “Internet of Things” (IoT). IoT can be stated as: it is a system, in which interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with distinctive identifiers and the ability to transfer data over a network. There are many application areas for IoT implementation, such as Health Care, Transportation, Industry, Market, Education, Vehicles, Smart Home and Agriculture, among others. The IoT applications are developed on the top of cloud systems [2], where the cloud system acts as the enabler for the IoT applications as shown in Fig. 1. The cloud has three main features such as SaaS (Software as a Service), SaaS (Software as a Service), and IaaS (Infrastructure as a Service) [2]. Therefore, end users get different services such as medical, educational, industrial and so on, by accessing the cloud server, which is known as the service provider. In IoT applications, Internet enabled things like vehicles and sensors collect data from the area or environment and supply the data to the cloud server. The cloud server processes the data and provides a corresponding feedback. In order to get the data from remote places by accessing the server, the end users have to get permission from the server first and then agree upon a shared secret key for further secure communication within the current session which is known as remote user authentication (see Fig. 1). After getting the data, end users can also provide a feedback to the Internet enabled e-devices. A huge amount of data transaction takes place in any IoT system. For each case of data transaction, the system needs to check whether the user is authentic or not by the proper efficient and secure authentication protocol. This work concentrates on developing a secure remote user verification scheme in cloud environment of IoT applications.

● Motivation

There are several challenges in case of user authentication techniques. History says that no security could prove absolute secure over long period due to the smart and updated attacker. However, this study finds that most of the existing authentication schemes (can be applied in cloud based applications) do not protect systems from all security attacks. Furthermore, the existing protocols have lack of efficiency in terms of: (a) computational cost, (b) communication cost, (c) inability to detect wrong inputs during login as well as password phases, (d) extra communication overhead to alter the users’ password, and (e) disclosure of the users’ identity to the attacker. A proper efficient and secure authentication scheme for cloud based IoT applications should overcome or alleviate all the aforementioned issues and provide user friendly facilities.

● Contribution

This paper proposes a secure scheme using biometric information of users and ElGamal cryptosystem. We refer to this here as SAS-Cloud (Secure Authentication Scheme in Cloud based IoT Systems), to build a concrete authentication system for cloud applications. The security of SAS-Cloud is examined using well popular random oracle model, and the efficiency of SAS-Cloud is evaluated and compared with other reported competing schemes.
The study is structured as follows. A quick overview of existing authentication protocols is highlighted in the next section. Section 3 describes some mathematical definitions, which are used in the proposed scheme. Section 4 demonstrates the adversary model as well as network model to introduce the proposed scheme. Our ElGamal-based three factor authentication scheme for cloud based IoT application, SAS-Cloud, is described in section 5. Security analysis of the proposed SAS-Cloud scheme and performance comparison of SAS-Cloud with related competing schemes are provided in section 6 and section 7, respectively. Advantages to use the proposed SAS-Cloud is given in section 8. The concluding remarks of this paper are stated in section 9.

2 Related Work

Lamport [3] first introduced a password-based authentication scheme using one way hash function. Thereafter many user authentication schemes [4–10] have been presented in this regard, which are based on only password for various Internet based applications. Jain et al. in [11] mentioned that biometric information based technology produces an effective verification tool in wireless communication. Furthermore, in many commercial, civilian, and forensic applications, biometric systems have been installed to verify identity of users [11]. Therefore, the researchers have appraised biometric with the password to amplify the degree of security [12]. Research community of this study have suggested various password and biometric based authentication schemes in [12–20]. Tan [13] presented a three-factor authentication scheme in 2013. According to Yan et al. [14], the scheme [13] is insecure from the Denial-of-Service (DoS) attack and suggested their own scheme. However, Mishra et al. [15] showed that the scheme in [14] can not protect the off-line password guessing attack and it has incompetent login, and password change phases. Chuang and Chen’s scheme [16], and introduced a countermeasure scheme in [17]. Very recently, Waizid et al. [19] also introduced a biometric-based authentication scheme, which can be applied in cloud environment. Maitra and Giri [17] stated that an adversary can create forge message on Chuang and Chen’s scheme [16], and introduced a countermeasure scheme in [17]. Very recently, Waizid et al. [19] also introduced a biometric-based authentication scheme in cloud environment and after evaluating their proposed scheme through formal security analysis, the authors claimed that the proposed scheme is secure from security threats. However, the discussed authentication protocols are based on only hash function, thus the security of those schemes are dependant on hardeness of one-way hash function.

On the other hand, authentication using public key cryptography like RSA based [21, 22], ElGamal based [23], Robin cryptosystem based [24], ECC-based [25–27], Bilinear Pairing based [28, 29] and so on is also well popular in the literature. However, this paper aims to design an authentication scheme for cloud based IoT application using ElGamal cryptosystem [30]. Hence we discuss only the authentication protocols using ElGamal cryptosystem reported in [23, 31–38]. Hwang and Li [23] proposed an ElGamal based authentication protocol without using any password verification table in order to eliminate password stolen attack at the server end. The authors claimed that the proposed protocol can resist different known attacks. Chan and Cheng [31] identified that different kinds of security attacks like password guessing attack, impersonation attacks, man-in-the-middle attack and DoS attack can be mounted in the protocol [23]. Shen et al. [32] also proved that the protocol provided in [23] is defenceless against masquerading attack, therefore Shen et al. demonstrated a solution to prevent the masquerading attack on Hwang and Li’s scheme by proposing an enhanced scheme in [32]. However, the modified protocol [32], proposed by Shen et al. is not totally secured as pointed out by Leung et al. in [33]. Yoon et al. [34] proposed a new smart card based client server authentication protocol using ElGamal signature and claimed that their protocol can resist forgery attack. However, Tian et al. in [35] argued that the protocol in [34] cannot make absolute protection against forgery attack and proposed a modified protocol in [35]. Ramasamy and Muniyandi [36] introduced a smart card based authentication protocol using ElGamal cryptosystem claiming that their protocol can resist forgery attack. However, Tian et al. in [35] argued that the protocol in [34] cannot make absolute protection against forgery attack and proposed a modified protocol in [35]. Ramasamy and Muniyandi’s scheme [36] cannot prevent all kind of attacks and they have proposed a new smart card based authentication protocol in [37] to overcome the shortcoming. Very recently, Maitra et al. [38] showed that an adversary can mount forgery attack as well as password guessing attack on Lee et al.’s scheme [37] after stealing the smart card of a legal user.

3 Preliminaries

Definition 1. A cryptographic hash function [17, 20] can be represented as: $H: S_1 \rightarrow S_2$, where $S_1$, a binary string of random length is taken as an input to produce a binary string $S_2$ of fixed length $l$. The cryptographic hash function $H(\cdot)$ is said to be collision-resistant, if the following condition is maintained:

$$Adv^H_{\mathcal{A}}(t_1) = Pr[| (a_1, a_2) \in S_1 \times S_1 | a_1 \ne a_2 \land H(a_1) = H(a_2)] < \epsilon$$

where $Pr[\mathcal{E}]$ represents the random event $\mathcal{E}$ produced by an adversary $\mathcal{A}$ for the time span $t_1$ and $Adv^H_{\mathcal{A}}(t_1) \le \eta_1$, for any small $\eta_1 > 0$ is the probability of advantage to find two different binary strings $a_1$ and $a_2$ over time span $t_1$.
4 Models

This section will discuss network and adversary models to introduce the proposed SAS-Cloud.

4.1 Network Model

According to the architecture of the proposed scheme, through enrollment procedure, users have to register to a service provider $SP$ to get their registration confirmation (See Fig. 2(a)). For this purpose, users send a request for registration to $SP$ in off-line or personally. After getting the request, $SP$ provides some registration information to the users so that the users can use this information in login time.

Whenever a registered user wants to get service from $SP$ by accessing the mobile application via insecure channel, the user transmits a login message to $SP$. Upon checking the login message, $SP$ gives reply to the user. After getting the reply, the user verifies the reply message (See Fig. 2(b)). For the correct reply, both the user and $SP$ agree on a secret and common session key [40].

4.2 Threat Model

This study has considered the threat model proposed by Dolev-Yao [41] to evaluate the security of the SAS-Cloud. According to this model [41], the communicating parties convey their message through an insecure channel during login as well as authentication phases. Therefore, an attacker $A$ can capture the transmitted messages, and furthermore $A$ can alter or delete the contents of the messages as shown in Fig. 2(b). The attacker $A$ also acquires the information, which is stored in the user’s electronic device like mobile phone, tablet or laptop by monitoring the consumption of power [42]. According to the threat model, this paper considers the following two attackers:

- **Attacks by Outsider**: A third party $A$ (as an attacker), who is unrelated to the system may try to hamper in the authentication procedure by mounting various attacks.
- **Attacks by Insider**: A valid user $A$ (as an attacker), who is a part of the system may try to obtain confidential information of the server so that $A$ can inject several attacks on the authentication system.

5 SAS-Cloud: The Proposed Scheme

In this section, we present a secure authentication scheme for IoT application using fuzzy extractor and ElGamal Cryptosystem, called as SAS-Cloud. Symbols and their uses are given in Table 1. SAS-Cloud has five phases namely, (a) set-up phase, (b) enrollment phase, (c) login phase, (d) authentication with key agreement phase and (e) password update phase.

5.1 Set-up Phase

A service provider $SP$ executes algorithm $\mathcal{A}$ to get a large prime number $q$. $SP$ picks a cyclic multiplicative group $G$ of order $q$ with a generator $g$. Then it picks a number $s$ randomly such that $s \in R Z_q^*$ and computes $PK = g^s \mod q$. Furthermore, it selects a cryptographical hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$, where $k$ is an integer number with fixed length. Ultimately, $SP$ declares public information $\text{Param} = G, PK, g, q, H(\cdot)$ and keeps $s$ as secret key.

5.2 Enrollment Phase

Whenever a new user $U_i$ likes to enroll in the service provider $SP$, registration phase is invoked using the steps shown below:

1. The user $U_i$ opens the application from his/her electronic gadget and inputs his/her biometric feature (i.e., fingerprint) to a sensor...
enabled device like mobile. The device creates a corresponding biometric information \( B_i \) (defined in Definition 2 of Section 3) and provides it to \( U_i \).

2. \( U_i \) chooses an identity \( ID_i \), password \( PW_i \), and generates an unique pair \((\theta_i, \phi_i)\) from \( B_i \), computing \((\phi_i, \theta_i) \rightleftharpoons \text{Gen}(B_i) \). \( U_i \) then computes \( PW_i' = H(PW_i || \phi_i) \) and sends \( (ID_i, PW_i', \theta_i) \) to \( SP \) through a private channel.

3. After getting a registration request \((ID_i, PW_i', \theta_i)\) from \( U_i \), \( SP \) computes \( A_i = H(ID_i || ID_i') \) and \( D_i = H(A_i) \). \( SP \) then checks that \( D_i \) exists in its list or not. If it is exist, \( SP \) gives a negative acknowledgement (i.e., decline message) to \( U_i \) because, received \( ID_i \) is not unique and it may be used by another user. In such case, \( U_i \) has to selects another identity until unique identity is not acquired. If \( D_i \) does not exist in its list, \( SP \) calculates \( C_i = A_i \oplus PW_i' \), and sends a registration information \((C_i, D_i, des(\cdot), \delta_i)\) to \( U_i \) through a private channel, where \( des(\cdot) \) is a distance measurement function (defined in Definition 2 of Section 3). \( SP \) then updates its list \( U_i \) with \( (ID_i, PW_i') \) by incorporating \( D_i \) into it.

4. After receiving the registration information \((C_i, D_i, des(\cdot), \delta_i)\), \( U_i \) computes \( B_i = B_i \oplus H(ID_i || PW_i') \) and \( \theta_i = \theta_i \oplus H(ID_i || PW_i') \). Finally, \( U_i \) stores \((C_i, D_i, B_i, \theta_i, des(\cdot), \delta_i)\) into the memory of his/her electronic gadget like mobile phone. Note that, this study assumes that extracted feature from biometric i.e., binary string \( B_i \) and result of hash value are same bits long, which are \( n \) bits.

Fig. 3 shows the pictorial view of enrollment phase.

5.3 Login Phase

If a registered user \( U_i \) likes to get entry into the system by accessing the service provider \( SP \), login phase is invoked. \( U_i \) opens the application from mobile and provides his/her biometric information \( B_i \) via sensor, identity \( ID_i \), and password \( PW_i \) to the mobile. The mobile then computes the following procedures:

1. The mobile executes \( B_i' = B_i \oplus H(ID_i || PW_i') \) and verifies \( des(B_i', B_i') \leq \theta_i \). If it does not satisfying the condition, \( U_i \) will be rejected; otherwise, it computes the next step.
2. The mobile computes \( \theta_i' = \theta_i \oplus H(ID_i || PW_i') \) and \( \phi_i' = \text{Repl}(\theta_i', \phi_i) \). \( PW_i' = H(PW_i || \phi_i') \), \( A_i = C_i \oplus PW_i' \), \( D_i' = H(A_i) \) and checks \( D_i' = D_i \). If equality does not preserve, the mobile refuses \( U_i \); otherwise, it computes the next step.
3. The mobile selects a number \( r_i \in \mathbb{Z}_n \) randomly and computes \( E_i = PK_i \cdot q \), \( G_i = g_i^{r_i} \cdot q \), \( D_i = ID_i \cdot (E_i \cdot \cdot E_i') \mod q \), \( D_i = (ID_i \cdot D_i) \mod q \) and \( F_i = H((ID_i, A_i, D_i')) \). Then, \( U_i \) transmits a login request message \((D_i', G_i, F_i, E_i)\) to \( SP \) through Internet (a public channel). Note that \( U_i \) operates his/her mobile thus, mobile of \( U_i \) sends the login message on behalf of \( U_i \). However, in this study, we use mobile device of \( U_i \) and \( U_i \) alternatively.

5.4 Authentication with Key Agreement Phase

Upon getting the login request message \((D_i', G_i, F_i)\) from \( U_i \), \( SP \) computes the following steps:

1. \( SP \) computes \( E_i = G_i^q \cdot \langle \cdot \rangle \), \( G_i = H(ID_i || PW_i') \mod q \) and calculates \( A_i' = H(s || D_i') \) and checks that \( D_i' = H(A_i') \) exists into \( U_i \)'s list or not. If it does not find it, \( SP \) rejects \( U_i \); otherwise, it executes next step.

[Verification of \((D_i || r_i) = D_i \cdot (E_i')^{-1} \):

\[
D_i = D_i \cdot (E_i')^{-1} \mod q
\]

\[
\equiv (ID_i || r_i) \cdot (E_i')^{-1} \mod q \quad \text{since} \quad D_i = (ID_i || r_i) \cdot E_i
\]

\[
\equiv (ID_i || r_i) \cdot PK_i \cdot (G_i')^{-1} \mod q
\]

\[
\equiv (ID_i || r_i) \cdot g_i^{r_i} \cdot (g_i')^{-1} \mod q
\]

\[
\equiv (ID_i || r_i)
\]

1. \( SP \) calculates \( F_i' = H(r_i' || A_i') \) and further checks \( F_i' = \? F_i \). For the inequality, \( SP \) rejects the login message of \( U_i \); otherwise, it goes to the next step.

3. \( SP \) selects a number \( q_i \in \mathbb{Z}_n \) random and further computes \( Q_i = A_i' \oplus q_i \cdot SK_i = H([y_i'] || r_i') \), \( L_i = H(ID_i || SK_i || A_i') \) and sends a reply message \((Q_i, L_i)\) to \( U_i \) via a insecure channel. \( SP \) accepts \( SK_i \) as a common and secret session key.

After getting the reply message \((Q_i, L_i)\) from \( SP \), the mobile of \( U_i \) performs the following step to authenticate the reply message of \( SP \):

1. The mobile calculates \( y_i' = A_i' \oplus Q_i \cdot SK_i = H([y_i'] || r_i') \), \( L_i' = H(ID_i || SK_i || A_i') \) and checks \( L_i' = L_i \). If the equality is satisfied, \( U_i \) concurs upon the common secret key \( SK_i = SK_i \); otherwise, it refuses the reply message.

Fig. 4 depicts a pictorial view of login and authentication with key agreement phases.

5.5 Password Update Phase

If a user \( U_i \) likes to alter his/her password, this phase is invoked. \( U_i \) opens the application from mobile and provides his/her biometric information \( B_i' \) via sensor, identity \( ID_i \), and password \( PW_i \) to the mobile. The mobile then executes the following steps:

\[
\begin{align*}
\end{align*}
\]

Table 1 Nomenclature used in the paper

<table>
<thead>
<tr>
<th>Term</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g )</td>
<td>A multiplicative group of prime order ( q )</td>
</tr>
<tr>
<td>( q )</td>
<td>Generator of group ( g )</td>
</tr>
<tr>
<td>( H(\cdot) )</td>
<td>Cryptographic hash function</td>
</tr>
<tr>
<td>( U_i )</td>
<td>ith User</td>
</tr>
<tr>
<td>( SP )</td>
<td>Service provider</td>
</tr>
<tr>
<td>( s )</td>
<td>Secret key of ( SP )</td>
</tr>
<tr>
<td>( PK )</td>
<td>Public key of ( SP )</td>
</tr>
<tr>
<td>( PW_i )</td>
<td>Password at ( U_i )</td>
</tr>
<tr>
<td>( ID_i )</td>
<td>Identity of ( U_i )</td>
</tr>
<tr>
<td>( B_i )</td>
<td>Biometric information of ( U_i )</td>
</tr>
<tr>
<td>( r_i )</td>
<td>Random number picked up by mobile</td>
</tr>
<tr>
<td>( y_i )</td>
<td>Random number picked up by ( SP )</td>
</tr>
<tr>
<td>( des(\cdot) )</td>
<td>Distance evaluation function</td>
</tr>
<tr>
<td>( \delta_i )</td>
<td>Estimated difference</td>
</tr>
<tr>
<td>( Y' )</td>
<td>Parameter ( Y ) calculated or obtained by mobile</td>
</tr>
<tr>
<td>( SK_i )</td>
<td>Common and secret session key between ( U_i ) and ( SP )</td>
</tr>
<tr>
<td>( \oplus )</td>
<td>Exclusive-OR operation</td>
</tr>
<tr>
<td>( \parallel )</td>
<td>Concatenation/append operation</td>
</tr>
</tbody>
</table>

IET Research Journals, pp. 1–11 © The Institution of Engineering and Technology 2015
1. The mobile calculates $\delta_i = \theta_i \oplus H(ID_i \| PW_i)$ and verifies $des(\delta_i, \delta_i') \leq 6d$. For not satisfying the condition, $U_i$ will be rejected; otherwise, it computes the next step.

2. The mobile calculates $\theta_i' = \theta_i \oplus H(ID_i \| PW_i)$. If equality does not hold, $U_i$ will be rejected; otherwise, the mobile device is allowed to enter a new password.

3. $U_i$ picks a new password $PW_{i}^{\text{new}}$ and inputs it to the mobile. The mobile then executes $PW_{i}^{\text{new}} = H(PW_{i}^{\text{new}} \| \theta_i')$ and $\delta_i^{\text{new}} = \theta_i' \oplus H(ID_i \| PW_{i}^{\text{new}})$, respectively.

Theorem 1. Under the assumption that a fuzzy extractor $FE$ and cryptographic hash function $H(\cdot)$ represent the random oracles, SAS-Cloud is provably secure against an attacker $A$ for acquiring the password $PW_i$, biometric $B_i$ and identity $ID_i$ of a user $U_i$, even if $A$ obtains the parameters that are preserved into the mobile of $U_i$ and captures the communication messages between $U_i$ and the service provider $SP$.

### Security Analysis of SAS-Cloud

This study has done the formal security analysis of SAS-Cloud under the random oracle model. We define the random oracles for the formal security analysis of SAS-Cloud as follows:

- A random oracle $OracleF$ keeps a tuple $(u, v)$ such that $v = H(u)$. It supplies $u$ from $v$ upon getting a query $(q, H(v))$ if $(u, v)$ is exist in the tuple; otherwise, produces a number $r_1$ randomly. Then it reserves $(r_1, v)$ into its tuple as a new entry.

- A random oracle $OracleH$ keeps a tuple $(u, v)$ such that $v = H(u)$ if $(u, v)$ is exist in the tuple; otherwise, produces a number $r_1$ randomly. Then it reserves $(r_1, v)$ into its tuple as a new entry.

**Fig. 3:** Enrollment phase of SAS-Cloud

**Fig. 4:** Login and authentication with key agreement phases of SAS-Cloud
Algorithm 1 EXP_{oracle, SAS−Cloud}

Input: C_i, D_i, \tilde{B}_i, \tilde{\theta}_i, DID_i, G_i, F_i, l, PK, g, q

Output: 0 or 1; 0: Fail and 1: Win

1: Asks Oracle_{\mathcal{H}} on the input D_i to obtain the information A_i \leftarrow H(s||D_i) as \langle A_i^* \rangle \leftarrow \text{Oracle}_{\mathcal{H}}(D_i)
2: Asks Oracle_{\mathcal{H}} on the input F_i to get the information A_i, r_i, and E_i as \langle r_i^* || E_i^* || A_i^* \rangle \leftarrow \text{Oracle}_{\mathcal{H}}(F_i)
3: Asks Oracle_{\mathcal{H}} on the input L_i to get the information SK_i, \Gamma_i, D_i, and A_i as \langle \Gamma_i^* || SK_i^* || A_i^* \rangle \leftarrow \text{Oracle}_{\mathcal{H}}(L_i)
4: Computes E_i^* = PK\cdot m_{\mathcal{Q}} \mod q \text{ and } G_i = \tilde{g}^* \mod q
5: if \langle A_i^* || \tilde{A}_i^* || A_i \rangle \& \langle E_i^* || \tilde{E}_i^* \rangle \& \langle G_i || \tilde{G}_i \rangle \text{ then}
6: Computes \langle ID_i^* || r_i^* \rangle = DID_i \cdot (E_i^*)^{-1} \mod q
7: Asks Oracle_{\mathcal{H}} on the input \tilde{A}_i^* to get the information s and \langle ID_i \rangle as \langle s^* || ID_i^* \rangle \leftarrow \text{Oracle}_{\mathcal{H}}(\tilde{A}_i^*)
8: if \langle s^* || r_i^* \rangle \& \langle ID_i^* = ID_i^* \rangle \text{ then}
9: Computes \langle PW_i || PW_i^* \rangle = C_i \oplus A_i
10: \text{repeat}
11: \text{Choose a password } PW_i^{\{\text{password}\}}
12: \text{Computes } B_i^* = \tilde{B}_i \oplus H(ID_i^* || PW_i^{\{\text{password}\}}) \text{ and } \tilde{\theta}_i^* = \tilde{\theta}_i \oplus H(ID_i^* || PW_i^{\{\text{password}\}})
13: Asks Oracle_{\mathcal{FE}, SP} on the input \tilde{B}_i^* and \tilde{\theta}_i^* to get the information \phi_i as \langle \phi_i^* \rangle \leftarrow \text{Oracle}_{\mathcal{FE}, SP}(\tilde{B}_i^*, \tilde{\theta}_i^*)
14: Calculates \langle PW_i^{\{\text{password}\}} || PW_i^{\{\text{password}\}} \rangle = H(PW_i^{\{\text{password}\}} || \phi_i^*)
15: \text{until } \langle PW_i^{\{\text{password}\}} \rangle = PW_i^*
16: if \langle PW_i^{\{\text{password}\}} \rangle = PW_i^* \text{ then}
17: Accepts \langle PW_i || PW_i^* \rangle, ID_i^* \text{ and } B_i^* \text{ as correct password, identity and biometric}
18: Return 1
19: else
20: Return 0
21: end if
22: else
23: Return 0
24: end if
25: else
26: Return 0
27: end if

Proof: This study constructs an attacker \mathcal{A} who has the ability to obtain the password PW_i, identity ID_i and biometric information B_i of U_i. In this regards, this work assumes that the mobile device of a user U_i is lost or stolen. Therefore, \mathcal{A} can obtain the stored information \langle C_i, D_i, \tilde{B}_i, \tilde{\theta}_i \rangle from the memory of mobile of U_i by calculating power consumption [42]. The attacker \mathcal{A} also captures the login request message \langle DID_i, G_i, F_i \rangle and a reply message \langle Q_i, L_i \rangle. The adversary \mathcal{A} executes the experiment, EXP_{oracle, SAS−Cloud} for our secure authentication scheme (SAS-Cloud) to get the password PW_i, identity ID_i and biometric parameter B_i of the user U_i, as discussed in Algorithm 1. This study defines the success probability for EXP_{oracle, SAS−Cloud} as Succ_{oracle, SAS−Cloud} = \frac{2Pr[\text{Exp_{oracle, SAS−Cloud}} = 1]}{2} + 1. Then the advantage of EXP_{oracle, SAS−Cloud} is given by Adv_{oracle, SAS−Cloud}(t, qH, qFE) = \max_{\mathcal{A}} \left\{ \text{Succ}_{oracle, SAS−Cloud} \right\}, where the maximum is considered over all \mathcal{A} with the implementation time t, qH and qFE are the # of queries submitted to Oracle_\mathcal{H} and Oracle_\mathcal{FE} oracles, respectively. It can be said that SAS-Cloud is provably secure against the attacker \mathcal{A} for obtaining the password PW_i, identity ID_i and biometric information B_i of U_i, if Adv_{oracle, SAS−Cloud}(t, qH, qFE) \leq \eta, for any negligible \eta > 0. EXP_{oracle, SAS−Cloud} (see Algorithm 1) shows that, if \mathcal{A} earns success to execute the reverse of cryptographic hash function H() and can explore the hardness of fuzzy extractor, then \mathcal{A} will correctly obtain the password PW_i, identity ID_i and biometric parameter B_i of U_i by employing random oracles Oracle_\mathcal{H} and Oracle_\mathcal{FE}, respectively, and secures the win in this game. However, according to Definitions 1 and 2, we can write that Adv_{oracle, H}(t) \leq \eta_i, for any negligible \eta_i > 0 and Adv_{oracle, FE}(t) \leq \eta_i, for any negligible \eta_i > 0. Therefore, we get Adv_{oracle, SAS−Cloud}(t, qH, qFE) \leq \eta_i for any negligible \eta_i > 0 as the SAS-Cloud depends on both Adv_{oracle, H}(t) and Adv_{oracle, FE}(t). Therefore, SAS-Cloud provides the security against the attacker \mathcal{A} for obtaining the password PW_i, identity ID_i and biometric information B_i of U_i.

Theorem 2. Under the assumption that DLP and cryptographic hash function H() represent the random oracles, SAS-Cloud is provably secure against an attacker \mathcal{A} for getting the private key s of service provider SP even if \mathcal{A} knows the parameters that are reserved into U_i’s mobile and captures the communication messages between U_i and SP.

Proof: This work constructs an attacker \mathcal{A} who has the ability to get the private key s of the service provider SP. However, we consider the same suppositions discussed in Theorem 1. The attacker \mathcal{A} executes the experiment, EXP_{oracle, SAS−Cloud} for the secure authentication scheme (SAS-Cloud) to get the private key s of the service provider SP as provided in Algorithm 2. We define the success probability for EXP_{oracle, SAS−Cloud} as Succ_{oracle, SAS−Cloud} = \frac{2Pr[\text{Exp_{oracle, SAS−Cloud}} = 1]}{2} + 1. Then the advantage of EXP_{oracle, SAS−Cloud} is given by Adv_{oracle, SAS−Cloud}(t, qH, qD) = \max_{\mathcal{A}} \left\{ \text{Succ}_{oracle, SAS−Cloud} \right\}, where we have considered the maximum over all \mathcal{A} with the implementation time t, # of queries qH, qD submitted to Oracle_\mathcal{H} and Oracle_\mathcal{D} oracles, respectively. The proposed SAS-Cloud is provably secure against the attacker \mathcal{A} for obtaining the secret key s of the service provider SP, if Adv_{oracle, SAS−Cloud}(t, qH, qD) \leq \eta, for any negligible \eta > 0.
Algorithm 2 \[\text{EXP}_A^{oracle},\text{SAS-Cloud}\]

**Input:** \(D_i, D_{ID}, G_i, F_i, L_i, PK, q, q_i, Q_i\)

**Output:** \(0 \text{ or } 1; 0: \text{Fail}, 1: \text{Win}\)

1. Asks \(O_{oracle}\) on the input \(D_i\) to obtain the information \(A_i = (H(s_i|TD_i))\) as \((A_i^*) \leftarrow O_{oracle}(D_i)\)
2. Asks \(O_{oracle}\) on the input \(F_i\) to get the information \(A_i, r_i, E_i\) as \((r_i^*, E_{i}\) mod \(q) \leftarrow O_{oracle}(F_i)\)
3. Asks \(O_{oracle}\) on the input \(PK = (s^*\mod q)\) to obtain the information \(s^* = O_{oracle}(PK)\)
4. Asks \(O_{oracle}\) on the input \(L_i\) to get the information \(SK_i, TD_i\) and \(A_i\) as \((TD_i, SK_i^*, A_i^*) \leftarrow O_{oracle}(L_i)\)
5. Asks \(O_{oracle}\) on the input \(G_i = (s^*\mod q)\) and \(q\) to get the information \(r_i, (r_i^*, E_i^* \leftarrow O_{oracle}(G_i, q)\)

6. If \((r_i^* = r_i^*^*)\) & \((A_i^* = A_i^* = A_i^*)\) then
   7. Computes \(E_{i}^* = PK\) mod \(q\)
   8. If \((E_i^* = E_i^*)\) then
      9. Computes \((TD_i^* \mod q^*) = D_{ID, (E_i^*\mod q)}\)
   10. Asks \(O_{oracle}\) on the input \(A_i\) to get the information \(s\) and \(TD_i\) as \((s^*\mod |TD_i|^*) \leftarrow O_{oracle}(A_i)\)
   11. If \((r_i^{*^*} = r_i^*)\) & \((TD_i^* = TD_i^* \mod q^*)\) & \((s^* = s^*)\) then
      12. Accepts \(s^*\) as private key of \(SP\)
      13. Return 1
   14. else
      15. Return 0
   16. end if
   17. else
      18. Return 0
   19. end if
   20. else
      21. Return 0
   22. end if

Theorem 3. Under the assumption that the DLP and cryptographic hash function \(H(x)\) represent the random oracles, the SAS-Cloud is provably secure against an attacker \(A\) for obtaining the common secret session key \(SK_i\) between \(UI_i\) and \(SP\) even if \(A\) knows the parameters that are reserved into \(UI_i\)'s mobile device and captures the communication messages between \(UI_i\) and \(SP\).

**Proof:** This work constructs an attacker \(A\) who has ability to obtain the session key \(SK_i\) between a user and the service provider \(SP\). However, we consider the same suppositions discussed in Theorem 1. The attacker \(A\) executes the experiment, \(\text{EXP}_A^{oracle},\text{SAS-Cloud}\) for the secure authentication scheme (SAS-Cloud) to obtain the session key \(SK_i\) between \(UI_i\) and \(SP\) as given in Algorithm 3.

We define the success probability of \(\text{EXP}_A^{oracle},\text{SAS-Cloud}\) as \(\text{Success}_A^{oracle},\text{SAS-Cloud} = |2\text{Pr}[\text{EXP}_A^{oracle},\text{SAS-Cloud} = 1] - 1|.\) Then the advantage of \(\text{EXP}_A^{oracle},\text{SAS-Cloud}\) is given by \(\text{Adv}_A^{oracle},\text{SAS-Cloud}(t, qH, qD) = \max_x(\text{Success}_A^{oracle},\text{SAS-Cloud})\), where we have taken the maximum over all \(A\) with the implementation time \(t, qH\) and \(qD\) as the # of queries asked to \(O_{oracle}\) and \(O_{oracle}\) oracles, respectively. We can say SAS-Cloud is provably secure against the attacker \(A\) for obtaining the session key \(SK_i\) between \(UI_i\) and the service provider \(SP\). If \(\text{Adv}_A^{oracle},\text{SAS-Cloud}(t, qH) \leq \eta_1\), for any negligible \(\eta_1 > 0\) and \(\text{Adv}_A^{oracle},\text{SAS-Cloud}(t, qD) \leq \eta_2\), for any negligible \(\eta_2 > 0\). SAS-Cloud (discussed in Algorithm 3) shows, if the attacker \(A\) is able to calculate reverse of the cryptographic hash function \(H(x)\) and also cracks DLP, then \(A\) can get success to derive the session key \(SK_i\) by employing the OracleH and OracleD random oracles, and gets victory in the game. Nevertheless, after observing Definition 1 and Definition 3, we can write \(\text{Adv}_A^{oracle}(t, qH) \leq \eta_1\), for any negligible \(\eta_1 > 0\) and \(\text{Adv}_A^{oracle}(t, qD) \leq \eta_2\), for any negligible \(\eta_2 > 0\). SAS-Cloud depends on \(\text{Adv}_A^{oracle}(t)\) as well as \(\text{Adv}_A^{oracle}(t)\). Thus, the proposed SAS-Cloud is providing security against the attacker \(A\) for obtaining \(s\) of the service provider \(SP\).

Theorem 4. A registered user \(UI_i\) as an attacker cannot extract the private key \(s\) of the service provider \(SP\) even if he/she has stored parameters into his/her mobile device.

**Proof:** A registered user say, \(UI_i\) as an attacker \(\hat{A}\) may try to login into SAS-Cloud as another valid user say, \(UI_j\). To do so, \(\hat{A}\) must knows the private key \(s\) of the service provider \(SP\). Since, \(\hat{A}\) is a legal user, \(\hat{A}\) knows his/her identity \(TD_i\), password \(PW_i\) and biometric parameter \(\phi_i\). Therefore, \(\hat{A}\) is able to derived \(H(s|TD_i)\) by executing \(C_i \subseteq H(PW_i|\phi_i)\), where \(C_i\) is the stored information into his/her mobile device. However, from \(H(s|TD_i)\), \(\hat{A}\) unable to derive \(s\) due to hardness of the reverse of cryptographic hash function. In addition, Theorem 2 exhibits that \(s\) cannot be derived from familiar parameters. As a result, \(\hat{A}\) cannot produce any security attacks on SAS-Cloud.

\(\text{IET Research Journals, pp. 1–11} \quad \text{© The Institution of Engineering and Technology 2015}\)
Algorithm 3 $EXP_{oracle}$

Input: $D$, $DID_i$, $G_i$, $L_i$, $PK$, $g$, $q$, $Q_i$
Output: $0$ or $1$; $0$: Fail, $1$: Win

1: Asks Oracle $H$ on the input $D_i$ to obtain the information $A_i = (H(s(\mathbb{ID}))$ as ($A_i^*$) $\leftarrow$ Oracle($D_i$)
2: Asks Oracle $H$ on the input $F_i$ to get the information $A_i$, $r_i$ and $E_i$ as ($E_i^*$) $\leftarrow$ Oracle($F_i$)
3: Asks Oracle $PK$ on the input $PK = (g^s \mod q)$ to retrieve the information $s$ as ($s^*$) $\leftarrow$ Oracle($PK$)
4: Asks Oracle $D_i$ on the input $G_i$ to get the information $SK_i$, $TD_i$, and $A_i$ as ($TD_i^*$ $\leftarrow$ Oracle($G_i$))
5: Asks Oracle $D_i$ on the input $G_i$ $= (g^s \mod q)$ to retrieve the information $r_i$ as ($r_i^*$) $\leftarrow$ Oracle($G_i$)
6: if ($A_i^* = A_i^*$) & ($r_i^* = r_i^*$) then
7: Computes $E_i^* = PK^{-1} \mod q$ and $y_i^* = Q_i \oplus A_i^*$
8: if ($E_i^* = E_i^*$) then
9: Computes ($TD_i^*$ $\leftarrow$ $E_i^*$) $= D_i$ $\cdot$ ($E_i^*$) $^{-1} \mod q$
10: if ($s^* = s^* \) & ($TD_i^* = TD_i^*$) then
11: Executes $SK_i^* = H(y_i^* \cdot r_i^*)$
12: if ($SK_i^* = SK_i^*$) then
13: $SK_i^*$ is accepted as the common session key
14: Return 1
15: else
16: Return 0
17: end if
18: else
19: Return 0
20: end if
21: else
22: Return 0
23: end if
24: else
25: Return 0
26: end if

6.1 Remarks on Proposed Theorems

Theorem 1 demonstrates that SAS-Cloud is providing security against the off-line password guessing attack. In SAS-Cloud, $A$ cannot produce the forgery attack without knowing the $PW_i$ and biometric information $B_i$ of a user $U_i$, and the private key $s$ of the service provider $SP$. Theorem 1 and Theorem 2 show that the confidential parameters of the service provider and the user are well protected from the attacker. As a result, there is no feasibility to produce the forgery attack on SAS-Cloud.

Furthermore, Theorem 3 shows that SAS-Cloud can protect the session key obtaining attack because, without any knowledge of random nonce(s) $r_i$ and $y_i$, $A$ cannot derive the session key $SK_i^*$.

In SAS-Cloud, the communicating messages are computed using random numbers. Hence, the messages are assuring to be non-identical for each session. As a result, $A$ cannot create the replay attack on SAS-Cloud. In addition, Denial of Service (DoS) attack is easily identified in SAS-Cloud (See Section 8).

7 Performance Evaluation

Here, the performances of SAS-Cloud are compared with the competing existing authentication schemes namely, Tan’s scheme [13], Yan et al.’s scheme [14], Mishra et al.’s scheme [15], Chuang and Chen’s scheme [16], Hwang and Li’s scheme [23], Shen et al.’s scheme [32], Yoon et al.’s scheme [34], Ramasamy and Muniyandi’s scheme [36] and Lee et al.’s scheme [37]. The compared schemes in [13–16, 23, 32, 34, 36, 37] are not usable for practical scenarios because, these schemes are not resisting the security attacks (See Table 2). In the related work section of this work, we have described that most of the proposed schemes are insecure against security attacks. Furthermore, analysis of security of SAS-Cloud (see Section 6) shows that it can protect all the possible attacks. Therefore, SAS-Cloud is more secure than other schemes.

Table 3 is given to show the storage cost, computational cost and communication overhead comparison of the schemes in [13–16, 23, 32, 34, 36, 37] with the proposed SAS-Cloud. Here, only login and authentication phases have been considered due to rapid and maximum usage during online cloud services. On the other hand, registration of users is done offline and this phase is used only one time in the authentication systems. Therefore, communication and computational costs of registration phase can be neglected with respect to login and authentication phases. $T_{EXP}$, $T_H$, $T_M$, $T_{ENC}$ and $T_{DEC}$ are the times required for exponentiation operation, hash operation, multiplication operation, symmetric key encryption and decryption respectively. However, it is well known that exponentiation operation takes more time than other operations and order of execution time can be expressed as: $T_{EXP} \gg T_H \approx T_{ENC} / T_{DEC} > T_M$ [17]. SAS-Cloud takes time for three exponentiation operation in two phases, which is the lower among ElGamal-based schemes in [23, 32, 34, 36, 37]. However, according to MIRACL C/C++ Library with the specifications of system (i.e., processor: Intel(R) Core(TM) (5–210U CPU @ 1.70GHz, 2.40 GHz; RAM: 8 GB; 64-bit Windows 10; Visual C++ 2008 software), the time complexity of the different cryptographic operations is roughly calculated as follows:

1. $T_{ENC} / T_{DEC}$: For private key en/decryption, the time complexity is $\approx 0.1303$ ms.
2. $T_H$: For cryptographic hash function, the time complexity is $\approx 0.0004$ ms.
3. $T_{EXP}$: Time complexity for exponentiation is $\approx 1.8269$ ms.
4. $T_M$: Time to execute multiplication operation is $\approx 0.0147$ ms.

According to the aforementioned information, a comparison graph (see Fig. 5) has been given as an evidence to show that SAS-Cloud takes less time to execute than related ElGamal cryptosystem based schemes. Here, we assume that the length of $TD_i$ and $PW_i$ are 64 bits each. Cryptographic hash function $H(\cdot)$, threshold value $\delta$, symmetric key encryp-
SAS-Cloud inputs a faulty password as well as faulty identity in login phase.

Table 2 Security attacks and functionality comparison of SAS-Cloud with related competing schemes

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Cost of Storage (in bits)</th>
<th>Cost of Communication (in bits)</th>
<th>Cost of Computation (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Login + Authentication</td>
<td>Login Authentication</td>
<td></td>
</tr>
<tr>
<td>Tan [13]</td>
<td>384</td>
<td>576</td>
<td>7T_{N} + 7T_{SEC}</td>
</tr>
<tr>
<td>Yan et al. [14]</td>
<td>640</td>
<td>960</td>
<td>10T_{N} + 4T_{SEC}</td>
</tr>
<tr>
<td>Mishra et al. [15]</td>
<td>800</td>
<td>1120</td>
<td>4T_{N} + 7T_{SEC}</td>
</tr>
<tr>
<td>Chuang and Chen [16]</td>
<td>576</td>
<td>960</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>Huang and Li [23]</td>
<td>1152</td>
<td>2280</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>Shen et al. [32]</td>
<td>2176</td>
<td>2280</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>Yoon et al. [34]</td>
<td>4544</td>
<td>2496</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>Ramasamy and Muniyandi [36]</td>
<td>4544</td>
<td>2496</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>Lee et al. [37]</td>
<td>384</td>
<td>3456</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
<tr>
<td>SAS-Cloud</td>
<td>640</td>
<td>2432</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
</tr>
</tbody>
</table>

Fig. 5: Execution time in login and authentication phases of various related schemes (ElGamal cryptosystem-based): a comparison with SAS-Cloud.

Table 3 Communication, storage and computation costs comparison of SAS-Cloud with competing existing schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Login + Authentication</td>
<td>Login Authentication</td>
<td>Login</td>
<td>Authentication</td>
</tr>
<tr>
<td>Tan [13]</td>
<td>384</td>
<td>576</td>
<td>7T_{N} + 7T_{SEC}</td>
<td></td>
</tr>
<tr>
<td>Yan et al. [14]</td>
<td>640</td>
<td>960</td>
<td>10T_{N} + 4T_{SEC}</td>
<td></td>
</tr>
<tr>
<td>Mishra et al. [15]</td>
<td>800</td>
<td>1120</td>
<td>4T_{N} + 7T_{SEC}</td>
<td></td>
</tr>
<tr>
<td>Chuang and Chen [16]</td>
<td>576</td>
<td>960</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>Huang and Li [23]</td>
<td>1152</td>
<td>2280</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>Shen et al. [32]</td>
<td>2176</td>
<td>2280</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>Yoon et al. [34]</td>
<td>4544</td>
<td>2496</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>Ramasamy and Muniyandi [36]</td>
<td>4544</td>
<td>2496</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>Lee et al. [37]</td>
<td>384</td>
<td>3456</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
<tr>
<td>SAS-Cloud</td>
<td>640</td>
<td>2432</td>
<td>2T_{X} + 2T_{M} + 3T_{N}</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5: Execution time in login and authentication phases of various related schemes (ElGamal cryptosystem-based): a comparison with SAS-Cloud.

8 Satisfaction to Use of SAS-Cloud

1. Efficient login system: By some unwanted mistakes, if a user U_{i} inputs a faulty password as well as faulty identity in login phase of SAS-Cloud, the mobile device can identify the incorrect inputs before going to create a login message. This is because, the mobile calculates B_{i} = B_{i} \oplus H(id_{U_{i}} || PW_{i}) and verifies d_{ex}(B_{i}', B_{i}) \leq d. For the incorrectness, the mobile discards U_{i}; otherwise, it computes B_{i}' = B_{i} \oplus H(id_{U_{i}} || PW_{i}), \phi_{i} \leftarrow \text{Replay}(B_{i}', \phi_{i}'), PW_{i}' = H(PW_{i} || \phi_{i}'), A_{i}' = C_{i} \oplus PW_{i}', \mathcal{D}_{i}' = \text{H}(A_{i}') and compares \mathcal{D}_{i}' \neq \mathcal{D}_{i}. For the dissimilar result, the mobile discards U_{i}; otherwise, considers PW_{i}' and id_{U_{i}} as correct inputs. Therefore, in SAS-Cloud for the wrong inputs, no login message will be generated which reduces extra communication overhead.

2. Efficient password update system: By some unwanted mistakes, if a user U_{i} inputs a faulty password as well as faulty identity in login phase of SAS-Cloud, the mobile device can identify the incorrect inputs before giving licence to the users to select their new password. This is because, the mobile follows the same steps as mentioned above to verify the correctness of entered inputs. Only for the correct inputs, the mobile gives licence to U_{i} to select new password. On the other hand, to update the password of a user, there is no need any communication between the mobile device and the
service provider. Hence, overhead for the communications is also decreased in SAS-Cloud.

3. Identity of Denial of Service (DoS) attack: Before going to create a login message, mobile device checks the password, biometric and identity of a user $U_i$ in SAS-Cloud. That means, for any wrong input from user, mobile device does not generate login message. Now, if a login request message of $U_i$ is discarded by service provider $SP$, then it can be said that the login message has been tampered with another party (i.e., attacker) or corrupted by some reasons. Therefore, the user may take necessary action to stop DoS attack by informing service provider.

4. Satisfaction of mutual authentication: In SAS-Cloud, the service provider $SP$ calculates and agrees on a secret session key $SK_i$ after checking the authenticity of the user $U_i$ through login message and after that, $SP$ transmits a reply message to $U_i$. Similarly, $U_i$ goes for the same secret session key $SK_i$ with $SP$ after checking the authenticity of $SP$ through reply message. Hence, two-way verification has been done in SAS-Cloud. Beside this, SAS-Cloud can protect all the possible security attacks (see, Section 6). Hence, SAS-Cloud satisfies mutual authentication.

5. Untraceability of user: In SAS-Cloud, identity of a user $U_i$ is dynamic for every session. This is because, $DIID_i$ is computed as $(TD_i, r_i) = E_{mod q}$, where $E_{mod}$ is a random number, which will be non-identical for each session in random oracle model. Thus for different sessions, $DIID_i$ will be changed and as security analysis of SAS-Cloud (see Theorem 1) shows that $TD_i$ cannot be extracted from known parameter for an adversary. Therefore, it can be claimed that the adversary cannot trace the user, which means the adversary is unable to locate the valid user’s existence.

9 Conclusion

This work observed that most of the authentication protocols using hash function and ElGamal cryptosystem for cloud based applications are affected by security attacks and are unable to hide the actual identities of the end users during login session. Therefore, this work has introduced a secure ElGamal-based authentication scheme called SAS-Cloud. Analysis of security of SAS-Cloud using random oracle model shows that it is secure from all possible attacks. Performance comparison of SAS-Cloud with competing schemes has shown that the proposed scheme is more efficient than these competing schemes. In addition, as biometric features like finger print, iris scan, retina scan, and hand geometry are used with password in SAS-Cloud, therefore they can improve the security label of password-based authentication scheme. In future, this work will be extended to provide secure authentication among cloud server and Internet enabled devices so that a complete security framework can be built for IoT applications.

10 References

17. ElGamal, T.: ‘In a public key cryptosystem and a signature based on discrete logarithms’, (Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, 10–18
