An Effective Trust-aware Authentication Framework for Cloud Computing Environment

Saboutnagaraju1*, S.K.V. Jayakumar2

1Dept. of Computer Science, Pondicherry University Community College, Lawspet, Pondicherry-605 008, India
2Dept. of Computer Science, School of Engineering & Technology, Pondicherry University, Kalapet, Pondicherry-605 014, India

*Corresponding Author: saboutnagaraju1983@gmail.com, Tel.: +91-87787-01151

Available online at: www.ijcseonline.org

Accepted: 06/Dec/2018, Published: 31/Dec/2018

Abstract—Although cloud computing has become one of the basic utility in ICT era with several benefits like rapid elasticity, resource pooling broad network access, and on-demand self-service, it introduces dozens of dirty security threats too. An effective authentication protocol is the basis, topmost prioritized and emergence one for the secure cloud communications. As a result, in this article an effective trust-aware authentication framework is proposed based on n-party multi-linear key pairing functions, trust and reputation aggregation functions and time-based dynamic nonce generation. In addition to formulating an effective authentication protocol, we have analyzed the mutual authentication and formal security strength by using cryptographic GNY belief logic which will prove proposed protocol not only meets intended mutual authentication, but also justifies the security strength against the impersonation and ephemeral secret leakage attacks.

Keywords—Mutual Authentication, Single Sign-On, Elliptic-Curve, Cloud Service Provider, Identity Provider, Trustee

I. INTRODUCTION

The recent development in Internet-of-Things, big data, mobile and social networks require cloud computing to provide economical data storage and high-speed computing capabilities. However, these imperatives are rapidly emerging as pillars for the smarter daily life and official works [1]. Although cloud computing has become one of the basic utility in ICT era with several benefits like rapid elasticity, resource pooling, broad network access, and on-demand self-service, it introduces dozens of dirty security threats too [2]. As per Cloud Security Alliance (CSA) research report 2018 [3], the top ten cloud specific security threats are data breaches, insufficient identity, credential and access management, web-based impersonations, insecure interfaces and APIs, system vulnerabilities, account hijacking, malicious insiders, advanced persistent threats, data loss, insufficient due diligence and denial of service. In [2], Bob Violino reported cloud authentication specific threats, among which the top five are data breaches, web-based impersonations, identity theft, account hijacking and malicious insiders. To preserve authenticity, authorization and key management properties in the cloud is given higher priority as these helps to protect the client’s sensitive information from malicious users [4].

It is observed that each and every traditional web or mobile application usually itself authenticates and authorizes the users and stores all the credentials and authorization information required. In this traditional scenario, each user may have multiple accounts for different applications with the same or similar credentials. Traditional authentication and authorization methods have been working successfully well for a long time. However, for the security reasons it is better for the user to use different passwords for each application and need to change all of them regularly. It is a tough work to do for the user. Even today, in a few leading web applications user credentials are actually stored in unencrypted form [5, 6].

It would be easier for the user, if all the applications have a common user credentials database. Here, users can access all the applications using one set of login credentials. This type of authentication is called OpenID or single sign-on (SSO). OpenID is an authentication protocol developed by non-profit OpenID foundation with a centralized credentials database server [7, 8, 9, 10]. The centralized database can be managed by a trusted third party identity provider. Here, end users are allowed to access different web applications using same set of login credentials. The key advantage of this OpenID mechanism is to eliminate the need of webmasters to provide their own ad hoc login systems. The major problem in this protocol is a user is allowed to confirm his/her identity
to a web application. This approach requires a trusted identity provider and it could become a bottleneck for the user authentication.

To overcome the problems of the OpenID mechanism, Chris Messina presented an OAuth 2.0 protocol [11, 12] not only for authentication but also for authorization of the users. In this protocol, a user is agrees to share his/her limited profile data from an OAuth identity provider. Here, a user can choose an OAuth identity provider like Microsoft, Google, Facebook, twitter etc. To choose or accept an Identity Providers (IdP) for the real-time applications should get recommendation from the OpenID Foundation. In [13, 14, 15, 16], security researcher reported that huge vulnerabilities are present in OAuth 2.0 libraries and OpenID approach and also demonstrated several security flaws in OAuth2.0 OpenSSL encryption process. OpenSSL has several security flaws due to that users access tokens and credentials are exposed to man in the middle attack and buffer overflow attacks. If OAuth2.0 is not configured correctly, it doesn’t even look at the access token, it just checks the User-ID has come from the correct source. Hence, there is a chance of impersonation attack. Importantly, popular practical cloud-based authentication mechanisms [17-24] are designed based on OAuth2.0 protocol.

An existing OpenID based authentication schemes [27, 28] provides security and convenience for mobile users to access multiple mobile cloud computing services from multiple Cloud Service Providers (CSPs) using only a single private key. The authors have taken effort to supports mutual authentication, key exchange, user anonymity, and user untraceability in the cloud. In these investigations, user password and finger print details are never shared with the CSPs. However the mechanisms are insecure against the service provider impersonation attack and the adversary can able to extract the user identity. These schemes also not secure against the Ephemeral Secret Leakage (ESL) attack and malicious insiders.

The followings are the key problems identified from existing cloud-based authentication investigations [7-28]:

- In the cloud computing environment, very critical and barely explored issue that should be taken into consideration is impersonation attack that impersonates the cloud communication with the false responses. Due to insufficient identity credentials and nonintellectual access key management controls, impersonation attackers can steal user credentials and gain the control over outsourced data and applications.

- There is a chance of impersonation attack in the configurations of the OAuth2.0-based cloud authentications.

- Passwords are not enough and maintenance of numerous passwords increases security risks.
- To reduce number of complex operations involved in the authentication.
- For some financial/personal gain, dishonest cloud staff/rogue system administrator may leak the user identities and access management details [29].
- Existing scheme is unrealistic in storing key credentials in the host device memory for identity verification.
- The development of an effective collaborative multi-factor authentication is critical.

In this article, we have presented a trust-aware authentication protocol to bring an appropriate solution for the above problems.

Rest of the paper is organized as follows, Section I presents literature reviews , Section II illustrates system-level model and assumptions, Section III provides system preliminaries, Section IV describes our investigation, section V discusses completeness of the proposed authentication protocol, Section VI reports the security and performance evaluation, Section VII concludes research work with future directions.

II. RELATED WORK

Developing an efficient, robust and more convenient mutual authentication mechanism for the distributed cloud computing environment is a challenging research work. This section presents the existing cloud-based authentication approaches that can meet stakeholder’s requirements at some extend.

A. Risk-based Multi-factor Authentications

In [17], Merritt Maxim reported that Gigya Customer Identity Management (CIM) platform v6.5 is a market-leading secure identity and access management solution for public cloud SaaS applications that facilitates to the stakeholders to safeguard their cloud assets. The solution provides a Risk-based Multi-factor Authentication (RBMFA) using risk factors and One-time Password (OTP). Here, the first factor is risk parameters and the second factor is OTP via mobile or email. The risk parameters can be registered customer device and current location. When a consumer tries to access the cloud service accounts by using a new device, it authenticates the user via text SMS or voice call. Finally, this approach blacklists the consumer after a specified number of access attempts fail. This solution also facilitates user authentication through social accounts registration and logins or third-party plug-ins. The major strength of this solution is to provide password-less authentication and cross-network registration and login analytics. Compared to other cloud authentication controls, this approach provides the best execution, administration, analytics, partner ecosystem and
reporting. This solution also has larger market presence and global presence of vendors. The major drawback of this approach is lacking in supporting TRUSTe security standards and certifications.

LoginRadius Identity and Access Management (LRIAM) mechanism [22] provides customizable identity solutions to securely access multitenant SaaS offering in the Microsoft Azure. LoginRadius platform supports a Risk-based Multi-factor Authentication (RBMFA). Here, the first factor is risk parameters and the second factor is OTP via mobile or email. The risk parameters can be registered customer device, network address and current location. This platform also facilitates the user authentication through social login, anonymous login, phone SSO login, federation SSO and two-factor Authentication (2FA). The major strength of this solution is to provide password-less authentication and federation SSO-based registration and logins. The major drawback of this approach is lacking in supporting TRUSTe security standards and certifications. The solutions also lack in providing appropriate partner ecosystem and secure customer data management.

Ping Identity and Access Management (PIAM) mechanism [21] provides market-leading authentication solutions to securely access public cloud SaaS applications. The solution platform easily integrates third-party identity providers’ servers with the cloud service providers’ servers. The solution provides a Risk-based Multi-factor Authentication (RBMFA) using risk factors and One-time Password (OTP). Here, the first factor is risk parameters and the second factor is OTP via mobile or email. The risk parameters can be registered customer device, network address and current location. This solution also facilitates the user authentication through social login linking or adaptive authentication policies. The major strength of this solution is to provide password-less authentication and cross-network registration and login analytics. Compared to other cloud authentication controls, this approach provides better execution, administration and partner ecosystem. The major drawback of this approach is lacking in supporting TRUSTe security standards and certifications. The solutions also lack in providing simplicity in the adaption customizations.

B. Single Sign-On Authentications
Janrain Identity Cloud [18] is another market-leading secure identity and access management solution for next generation cloud-based technologies such as IoT and big data networks. The solution ensures safe and seamless identity generation, establishment and management. It gives a set of options for the authentication based on user requirement, such as corporate login, mobile authentication, single sign-on, universal ID, social login, adaptive MFA authentication, etc.,. The major strengths of the solution are to manage hierarchical groups to access each individual critical resources and uses Single Sign-On (SSO) to delivers one login across multiple applications and domains. Compared to other cloud authentication controls, this approach is better in overall performance, compliance management, threats and risks management and administration. This solution also has larger market presence, geographical presence of vendors and supports HIPAA, ISO and SOC2 security certifications and privacy compliance. The major drawback of this approach is lacking in protection of data breaches and transparent policy management.

Salesforce Identity and Access Management (SIAM) mechanism [19] is designed to securely authenticate various multitenant SaaS applications. In this approach, Salesforce SSO login system is implemented by using OAuth2.0 protocol strategy across multiple organizations. Salesforce SSO allows the consumers to authenticate into their multiple registered cloud services without having separate accounts for each service. Instead, a user can access all the applications using one set of login credentials. SIAM provides various administrative tools to monitor, report and maintain user authentication and authorization access tokens. Compared to other cloud authentication controls, this approach performs better in availability, scalability, security and privacy compliance management, customer data management, analytics and reporting. The major drawback of this approach is it has smallest CIAM installed base and performs immature user authentication. If OAuth2.0 is not configured correctly, it does not even look at the access token, it just checks the User-ID whether it comes from the correct source.

Azure Active Directory Business-to-Customer (AADB2C) solution [23] provides a seamless fully customizable identity and access management solution to securely access multitenant SaaS offering in the Microsoft Azure. Azure-AD also offers easy to use, consumer-centric, affordable and flexible CIAM solutions to the stakeholders to access multiple cloud applications by using single set of credentials. This solution delivers the user authentication through self-service password management, device registration, social and on-premise login, employees and business partners SSO login, two-factor Authentication (2FA) and federation SSO. Compared to other cloud authentication controls, Azure-AD performs better in scalability and performance. The major drawback of this approach is it has smallest CIAM installed base and performs immature authentication data analytics and reporting. The solutions also lack in providing appropriate partner ecosystem, content management solutions and geographical presence of vendors.

C. Multi-factor Authentications
ForgeRock Identity Platform (FRIP) [20] is a unified platform for secure user identity and access management
services in the private cloud or on-premises applications. The solution provides a Time-based or HMAC-based Multi-factor Authentication (TBMFA or HBMFA) using user ID, password and One-time Password (OTP). Here, the first factor is user ID and password and the second factor is OTP via mobile or email or registered hard device. In this approach, OTP can be generated from the registered hardware device or apps and used in the user authentication. In another way OTP will be generated using hash functions or time interval specified. Specifically, the ForgeRock IAM services are suitable for data sharing and user consent. This approach preserves better privacy controls for data privacy protection compared to other mechanisms. This solution does not work for the user authentication in multitenant SaaS applications. The major drawback of this approach is it lacks in supporting TRUSTe security standards and certifications. This mechanism also performs immature user authentication in analytics and reporting.

Figure 1. Typical Architecture of Proposed Scheme for cloud computing environment

In [24-26, 32-41] authors described various authentication frameworks for cloud computing environment to facilitate mutual authentication and secure key management for cloud users. Observing limitations, these schemes perform computational overhead and cannot secure against the Ephemeral Secret Leakage (ESL) attack. In [27], Jia-Lun Tsai et al. described a scheme to provide security and convenience for the mobile users to access multiple mobile cloud computing services from multiple service providers using only a single private key. The authors have taken effort to supports mutual authentication, key exchange, user anonymity, and user untraceability in the cloud. In this scheme user password and finger print details are never shared with CSPs and SCG. However the mechanism is insecure against the service provider impersonation attack and the adversary can able to extract the user identity. Debiao He et al [28] presented a privacy-aware authentication solution to address the impersonation problem exist in Jia-Lun Tsai et al. scheme. This scheme also not secure against the Ephemeral Secret Leakage (ESL) attack and malicious insiders.

III. SYSTEM-LEVEL FRAMEWORK AND ASSUMPTIONS

In this section a system-level framework is presented for distributed cloud computing environment which consists of cloud service providers, identity provider, distributed trustee and users as shown in Figure 1. The personal and sensitive information of a data owner or enterprise will be managed in the geographically distributed cloud data centers. The cloud service providers outsource cheap, flexible and on-demand storage space and computing capabilities to the data owner to make this information available any time to the legitimated users. Trustee is a set of distributed servers that are managed by an organization or board of eminent security researchers. It is a separation from the identity provider and cloud resource applications and will run on a separate trusted
lockdowns security platforms. It collects and validates authentication codes and access tokens generated by the identity provider, if they are valid, then user will be directed to access cloud application. Trustee protects authentication codes and access tokens and are never been processed in the service providers platforms. And also audits and records SLA and PLA parameters. Trustee services are distributed geographically with shared and highly secured databases. Identity provider is an authentication and authorization server which computes session key material and generates identity and access codes for the user authentication. In our proposed framework, identity providers, legitimate users and CSPs must rely on distributed trustee. The notations and their meanings we used for describing our framework are listed in Table 1.

Table 1. List of Abbreviations

<table>
<thead>
<tr>
<th>Notations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_i&amp;CSP_j</td>
<td>User i and Cloud Service Provider j</td>
</tr>
<tr>
<td>E_{PB}()</td>
<td>A public-key encryption function</td>
</tr>
<tr>
<td>D_{PB}()</td>
<td>A decryption function’s corresponding to E_{PB}()</td>
</tr>
<tr>
<td>e()</td>
<td>A symmetric encryption’s function</td>
</tr>
<tr>
<td>d()</td>
<td>A symmetric decryption’s function corresponding to e()</td>
</tr>
<tr>
<td>e^()</td>
<td>Exponential function</td>
</tr>
<tr>
<td>UID^*</td>
<td>The ID which U_i inputs in authentication phase</td>
</tr>
<tr>
<td>PWD^*</td>
<td>The password which U_i inputs in authentication phase</td>
</tr>
<tr>
<td>MAC_Ui^*</td>
<td>The MAC address which U_i submits in authentication phase</td>
</tr>
<tr>
<td>UID</td>
<td>The ID which U_i inputs in registration phase</td>
</tr>
<tr>
<td>PWD</td>
<td>The password which U_i inputs in registration phase</td>
</tr>
<tr>
<td>MAC_Ui</td>
<td>The MAC address which U_i submits in registration phase</td>
</tr>
<tr>
<td>C_i</td>
<td>The cipher text sent by various communication entities.</td>
</tr>
<tr>
<td>T_{bp}</td>
<td>Bilinear pairing operation time</td>
</tr>
<tr>
<td>T_{m}</td>
<td>Multiplication operation time</td>
</tr>
<tr>
<td>T_{d&amp;T_c}</td>
<td>Division operation time and Concatenation operation time</td>
</tr>
<tr>
<td>T_{x&amp;T_h}</td>
<td>Ex-OR operation time and One-way Hash operation time</td>
</tr>
<tr>
<td>T_{dp&amp;T_m}</td>
<td>Trust on data processing and Trust on data privacy</td>
</tr>
<tr>
<td>R_{val} &amp; GT_{val}</td>
<td>Reputation value and Global Trust value</td>
</tr>
<tr>
<td>a, b and γ</td>
<td>Importance given to service cost, trust and reputation values</td>
</tr>
<tr>
<td>T_{dp} &amp; ST_{val}</td>
<td>Trust on data transmission and Service Trust value</td>
</tr>
<tr>
<td>K &amp; K_i</td>
<td>Shared session key and Shared session key of communication entity i</td>
</tr>
<tr>
<td>ST &amp; DT</td>
<td>Cloud Service Type and User Data Type</td>
</tr>
<tr>
<td>SS &amp; DS</td>
<td>Cloud Storage Size and User Data Size</td>
</tr>
<tr>
<td>PS &amp; RS</td>
<td>Cloud Processing Speed and User Requested Speed</td>
</tr>
<tr>
<td>SC &amp; SP</td>
<td>Cloud Service Cost and User Service Pay</td>
</tr>
</tbody>
</table>

IV. SYSTEM PRELIMINARIES

The n-party bilinear key pairing preliminaries we used in our proposed authentication protocol are described in this section. Let \( G_1, G_2, G_3 \) be three cyclic additively-written groups and let \( G_T \) be a cyclic multiplicative group of an exponential base \( g \) with a large prime number order \( p \).

Definition 4.1. Let a mapping \( \hat{e}=G_1 x G_2 \rightarrow G_T \) be a bilinear pairing that has characteristics as follow:
1. Bilinearity: \( \forall a, b, \in F_q^*, \forall g \in G_1, G_2, G_T, \hat{e}(g^a, g^b, g^c) = \hat{e}(g, g, g)^{abc} \).
2. Computability: Bilinear groups and bilinear mapping are computed efficiently.
3. If \( \hat{e}(g, g, g)=I \), then bilinear pairing preserves non-degeneracy property.

Definition 4.2. Let \( \hat{e} \) be a bilinear pairing on \( (G_1, G_2, G_T) \). The bilinear Elliptic Curve Diffie-Hellman key pairing for \( \forall a, b, \in F_q^*, \forall g \in G_1, G_2, G_T \) can be computed as \( \hat{e}(g^a, g^b, g^c) = \hat{e}(g, g, g)^{abc} \).

The above definitions and properties are used in our authentication process for establishing and generating shared session keys among the users, identity providers, cloud service providers and trustee. In key generation process there are up-flow and down-flow stages. In up-flow stage, each entity computes intermediate secret values and in the down-flow, intermediate results will be sent to the communication entity group to generate shared session keys. The
communication entities involved in authentication process are denoted as \(E_1, E_2, \ldots, E_n\). Trustee chooses an exponential base \(\alpha\) and a large prime number \(p\) as an order and secretly shares these values to the authorized users and CSPs.

During up-flow, each communication entity \(E_i\) performs a single exponent and concatenates resultant value to the received intermediate values as given in equation (1) and then sends it to \(E_{i+1}\).

\[
E_i \alpha^{[(N_i)K_i[\epsilon][i,j] \bmod [1, t] \bmod E_{i+1}]}
\]

\(E_{i+1}\) receives the up-flow as formulated in equation (2).

\[
E_i \left(\alpha^{N_1}, \alpha^{N_2}, \ldots, \alpha^{N_{i+1}}\right) \rightarrow E_{i+1}
\]

Upon receipt of the resultant flow, \(E_n\) computes the shared session key \(K\) as given in equation (3) by exponentiation of secrete value \(N_n\) chosen by \(E_n\).

\[
K = K_n = (\alpha^{N_1}, \alpha^{N_2}, \ldots, \alpha^{N_{i+1} \ldots N_n})^{N_n}
\]

The up-flow process ends and the down-flow process starts when \(E_i = E_n\). Once the shared session key \(K\) is computed, \(E_n\) starts the down-flow with \(n-1\) intermediate values as formulated in equation (4)

\[
(\alpha^{N_1 N_n}, \alpha^{N_2 N_n}, \ldots, \alpha^{N_{i+1} \ldots N_n})
\]

Upon receipt of \(n-1\) intermediate values, each entity \(E_i\) computes the shared session key as given in equation (5)

\[
K = K_i = (\alpha^{N_1 N_n \ldots N_{i+1} \ldots N_n})^{N_i}
\]

The down-flow ends when \(E_i = E_1\).

V. TRUSTED AUTHENTICATION PROTOCOL

In this section we describe a trust-aware mutual authentication protocol. In this protocol, authentication parameters’ matching will be performed in the identity provider servers. The authentication codes and access tokens generated by the identity provider will be validated in the distributed trustee servers. In our approach, user authentication credentials never shared with the cloud service providers. The control flow of the proposed authentication protocol is represented in Figure.2. This approach helps the users to protect identity and access management tokens from the malicious insiders and unauthorized external adversaries.

The protocol has three phases as follow.

**Initialization phase.** First, trustee chooses a random number as private key \((Pr_t)\) and computes \(Pb_t = h_I(Pr_t)\) as its corresponding public key, where \(h_I\) is a one-way hashing function. Next trustee selects various bilinear pairing function parameters \((p, a, b, G, h_2, h_3, n)\). Finally, trustee publishes \(Pb_n\) and \((p, a, b, G, n, h_2)\) to \(h_3\) as public parameters. Likewise, identity provider \((IdP)\) chooses a random number as private key \((Pr_{IdP})\) and computes \(Pb_{IdP} = h_I(Pr_{IdP})\) as its corresponding public key, where \(h_2\) is a one-way hashing function and publishes \(Pb_{IdP}\) as a public parameter. Similarly, CSP chooses a random number as private key \((Pr_{CSP})\) and computes \(Pb_{CSP} = h_I(Pr_{CSP})\) as its corresponding public key, where \(h_I\) is a one-way hashing function and publishes \(Pb_{CSP}\) and its service attributes.
The registration phase. Each User (U) filters CSPs based on service attributes like STQD, SSQD, PSQD, RSQD, and then sends a request to trustee to provide trust and reputation values of desired CSP. User (U) sends his/her chosen CSP, user-id (ID_{U}), password (pwd_{U}) and device MAC address (MAC_{U}) to IdP for registration. Where, each user selects desired cloud service provider (CSP_{IDP}) based on the global trust evaluation algorithm which is described in [26]. Identity provider computes h_{i}(PWD+salt)=HWD, h_{i}(RN\oplus MAC) = h_{i} (RN\oplus MAC) = HMAC and e_{MAC_{U}} (RN) = ERN, where h_{i}(-) and h_{i}(-) are the one-way hashing functions, e_{MAC_{U}}(-) is the symmetric encryption function using MAC_{U} and stores these values in distributed and highly secured databases. IdP sends ID_{U} and mutual operation on nonce to U and trustee through secure channel.

Assumption: Similarly, trustee and cloud service provider with IdP.

The authentication phase performs the following steps to validate remote user (U) log credentials.

1) User U inputs ID_{U}, chooses a random secret number x (x<p) and nonce n_{1} and then computes intermediate secrets as X_{i} = x^{g} mod p. U performs public key encryption on concatenation of ID_{U}, X_{i} and n_{1} and computes cipher text as CI=EP_{Pr} (ID_{U}, X_{i}) mod n, and then sends C_{1} as service request to Trustee.

2) Trustee k obtains U message details such as ID_{U}, CSP, and n_{1} by decrypting C_{1} using private key Pr_{k}. If UID_{k}==UID, then Trustee k selects a random secret number y (y<p) and calculates X_{y} = X_{i}^{y} mod p, X_{x} = e(X_{x}, X_{y}) mod p, n_{2} = n_{1} + 1 mod n and then derives C_{2} = E_{PB_{IDP}} (ID_{U}, X_{i}) mod n, and sends it as identity provider. If ID_{U} or CSP_{IDP} is not found or invalid, then the user request will be rejected.

3) Identity provider obtains Trustee k message details such as ID_{U}, X_{i}, and n_{1} by decrypting C_{2} using private key Pr_{IDP}. If ID_{U} == ID_{U} and ID_{IDP} == ID_{k} & valid? then chooses a secret z (z<p) and computes X_{z} = G^{z} mod p, session key k = X_{z} = e(X_{z}, X_{i}) mod p, and also performs the mutual operation on n_{z} = n_{1} + 1 mod n. Finally, computes C_{3} = E_{k} (ERN || X_{i} || X_{y} || X_{z} || X_{y} || X_{z} and sends C_{3} as identity provider. If UID_{k} or SPID_{k} is not found with trustee, then the authentication request will be rejected.

4) From C_{3}, U computes session key k = X_{z} = e(X_{z}, X_{i}) mod p, obtains ERN||n_{z} and then checks for mutual authentication i.e., n_{z} = n_{1} + 1 mod n, if it matches, then User is allowed to enter pwd_{U} and MAC_{U} and then obtains RN by decryption of message as d_{pwd_{U}}(RN) = RN. Finally, U computes h_{i}(RN\oplus MAC_{U}) = HMAC and C_{4} = E_{k} (HMAC || n_{1}) and sends C_{4} to IdP. If n_{z} = n_{1} + 1 mod n, then the authentication process will be terminated.

5) From C_{4}, IdP obtains HMAC_{*} from D_{k} (HMAC_{*} || n_{d}) and checks for HMAC_{*} == HMAC and n_{d} = n_{1} + 1 mod n? if matches, then computes authentication and access token T_{k}(Token_{k} || n_{d}), then checks for Token_{k} = E_{OTP} (ID_{U} || NA_{U} || OTP) || n_{d} and sends C_{5} to U. If n_{d} = n_{1} + 1, then the authentication process will be terminated.

6) From C_{5}, U obtains Token_{k} || n_{d} by using secret key k and then checks for n_{d} = n_{1} + 1 mod n? if matches then user is allowed to enter OTP and forms C_{k} = E_{k} (Token_{k} || NA_{U} || OTP) || n_{d} and sends C_{k} to Trustee, if n_{d} = n_{1} + 1, then the authentication process will be terminated.

7) Trustee_k computes k = e(X_{z}, X_{y}) mod p and obtains Token_{k} || NA_{U} || OTP || n_{d} and then obtains ID_{U} || N_{U} || OTP || n_{d} using Pr_{k} and then checks for ID_{U} == ID_{U} and NA_{U} || OTP = OTP & n_{d} = n_{1} + 1 mod n? if matches, then redirects to CSP applications with C_{k} and then sends C_{k} to U. If n_{d} = n_{1} + 1, then the authentication process will be terminated.

8) U checks for mutual authentication value as n_{d} = n_{1} + 1 mod n, if it matches, then user is allowed to access the cloud services. Otherwise, the request will be rejected.

The proposed mutual authentication protocol is described in Algorithm 1.

Algorithm 1: Authentication phase

Input: User-ID, password, MAC address and random nonce. Output: Accept or Reject remote user.

1) U_i inputs ID_{U}, selects x (x<p) and n_{1} Computes X_{i} = x^{g} mod p, C_{1} = E_{PB_{IDP}} (ID_{U} || X_{i}) mod n, then sends C_{1} as authentication request to Trustee.

2) Trustee_k computes D_{k} = E_{PB_{IDP}} (ID_{U} || CSP) || n_{1} = (ID_{U} || CSP) || n_{1} if ID_{k} && CSP_{IDP} = F and valid? then chooses a secret z (z<p) and computes X_{z} = G^{z} mod p, session key k = X_{z} = e(X_{z}, X_{i}) mod p, X_{y} = e(X_{y}, X_{z}) mod p and also performs the mutual operation on n_{z} = n_{1} + 1 mod n. Finally, computes C_{3} = E_{k} (ERN || X_{y} || X_{z} || X_{y} || X_{z} = E_{k} (ERT || X_{i} || X_{y} || X_{z} || X_{y} || X_{z} and sends C_{3} as identity provider. If UID_{k} or SPID_{k} is not found with trustee, then the authentication request will be rejected.

3) From C_{3}, U computes session key k = X_{z} = e(X_{z}, X_{i}) mod p, obtains ERT||n_{z} and then checks for mutual authentication value i.e., n_{z} = n_{1} + 1 mod n, if it matches, then User is allowed to enter pwd_{U} and MAC_{U} and then obtains RN by decryption of message as d_{pwd_{U}}(RN) = RN. Finally, U computes h_{i}(RN\oplus MAC_{U}) = HMAC and C_{4} = E_{k} (HMAC || n_{1}) and sends C_{4} to IdP. If n_{z} = n_{1} + 1 mod n, then the authentication process will be terminated.

4) From C_{4}, IdP obtains HMAC_{*} from D_{k} (HMAC_{*} || n_{d}) and checks for HMAC_{*} = HMAC and n_{d} = n_{1} + 1 mod n? if matches, then computes authentication and access token T_{k}(Token_{k} || n_{d}), then checks for Token_{k} = E_{OTP} (ID_{U} || NA_{U} || OTP) || n_{d} and sends C_{5} to U. If n_{d} = n_{1} + 1, then the authentication process will be terminated.
\[ IdP \rightarrow U_i \]

Otherwise, the authentication request will be rejected.

4) \( U_i \) Computes session key \( k = x_{ui} \cdot e(x_i, X_i) \mod p \)

obtains \( ERN \)[(\( n_i \)] and then checks for mutual authentication value \( i.e., n_i = n_i \times 1 \mod n \), if it matches, then user is allowed to enter \( pwd \) and \( MAC \) and then obtains \( RN \) by \( d_{pwd}(e(pwd,RN))=RN \), finally computes \( h_d(RN \oplus MAC) \cdot HMAC^* \) and forms \( C_4 = E_k(\text{HMAC}[^*]n_d) \).

\[ C_4 \rightarrow IdP \]

If \( n_i \neq n_i \times 1 \mod n \), then the authentication process will be terminated.

5) From \( C_4 \), \( IdP \) obtains \( HMAC^* \) from \( D_k(\text{HMAC}[^*]n_d) \) and checks for \( \text{HMAC}[^*]=\text{HMAC} \& \& n_i = n_i \times 1 \mod n \)? if matches, then computes authentication and access token \( E_k(\text{Token}_1)[n_i] \), where \( \text{Token}_1 = E_{\otimes}\text{OTP}(\text{ID}_i \mid \text{Na}_i \mid \text{OTP} \mid n_i) \) and \( n_i = n_i \times 1 \mod n \) and computes \( C_5 = E_k(\text{Token}_1[n_i]) \)

\[ \text{Trustee} \rightarrow U_i \text{and redirects to the Trustee, server} \]

If \( n_i \neq n_i \times 1 \), then the authentication process will be terminated.

6) From \( C_5 \), \( U_i \) obtains \( \text{Token}_1[n_i] \) by using secrete key \( k \) and then checks for \( n_i = n_i \times 1 \mod n \)?, if matches then user is allowed to enter \( OTP \) and forms \( C_6 = E_k(\text{Token}_1[n_i] \mid \text{Na}_i \mid \text{OTP}) \mid n_d \mid X_c \).

\[ U_i \rightarrow \text{Trustee} \]

Otherwise, the authentication request will be rejected.

7) \( \text{Trustee} \) computes \( k = e(X_i, x) \mod p \) and obtains \( \text{Token}_2[n_i] \)

\( \text{Na}_i \mid \text{OTP} \mid n_i \) using \( k \) and then obtains \( \text{ID}_i \mid \text{Na}_i \mid \text{OTP} \mid n_i \) using \( \text{Pr}_1 \) and then checks for \( \text{ID}_i = \text{ID}_i \& \& \text{Na}_i \& \& \text{OTP} \& \& n_i = n_i \times 1 \mod n ? \), if matches, then redirects to \( \text{CSP} \) applications with \( C_7 = E_k(n_i) \).

\[ \text{Trustee} \rightarrow U_i \]

Otherwise, the authentication request will be rejected.

8) \( U_i \) checks for mutual authentication value as \( n_i = n_i \times 1 \mod n \), if it matches, then user is allowed to access the cloud services. Otherwise, the request will be rejected.

**VI. COMPLETENESS OF THE PROPOSED PROTOCOL**

This section formally analyses the mutual authentication and security strength of the proposed protocol using standard GNY cryptographic logic. The analysis proved that the proposed protocol not only meets intended mutual authentication functionality, but also ensures the security strength against the service provider impersonation and other replay attacks. We used cryptographic GNY belief logic to formally analyze the working nature of our trusted authentication mechanism and to verify whether our mechanism meets its goals. GNY belief logic is the substantial extension of BAN logic. First, we present the basic terminologies and statements, protocol transformation, goals and assumption list we used. Next, we describe the logical postulates adoption.

1) **Basic Terminologies and Statements**

Let \( CP_i \) be the credential parameter message and the following basic terminologies are introduced on \( CP_i \):

- \( h(CP_i) \): hash operation on \( CP_i \).
- \( \{CP_i\} \cdot K \): \( CP_i \) is encrypted with \( +K \) and decrypted with \( -K \).
- \( \{CP_i\}, \{CP_i\} \cdot K \): \( CP_i \) is encrypted and decrypted with secrete key \( K \).

**Statements:** Let \( E_i \) and \( E_j \) be two communication entities and the following statements are formed on \( E_i \) and \( E_j \).

1) \( E_i < E_j \); \( E_i \) holds \( E_i \)

2) \( E_i \equiv CP_i \); \( E_i \) possesses credential parameter message \( CP_i \)

3) \( E_i \equiv CP_i \); \( E_i \) once conveyed \( CP_i \)

4) \( E_i \equiv \#CP_i \); \( E_i \) believes that \( CP_i \) is fresh

5) \( E_i \equiv (CP_i) \cdot E_i \); \( E_i \) is recognized

6) \( E_i \equiv E_i \cdot \#E_i \); \( E_i \) believes that \( S \) is a suitable secrete for \( E_i \) and \( E_j \)

7) \( E_i \equiv +K \cdot E_i \); \( E_i \) believes that public key \( +K \) is suitable for \( E_j \)

8) \( E_i \equiv X \); \( E_i \) has jurisdiction over \( X \)

9) \( E_i \equiv ^*X \); \( E_i \) is told that he/she didn’t convey \( X \) previously in the current session.

2) **Protocol Transformation**

Our proposed authentication protocol is mapped into the form of \( E_i \rightarrow E_j \cdot CP_i \)

1) \( U_i \rightarrow \text{Trustee} \cdot \{\text{ID}_i \mid \text{Na}_i \} \mid \text{CSP} \cdot \text{pwd} \mid X_c \}

2) \( \text{Trustee} \rightarrow \text{IdP} \cdot \{\text{ID}_i \mid \text{TR}\} \mid n_d \cdot k \mid X_i \}

3) \( \text{IdP} \rightarrow U_i \cdot \{\text{ERN} \mid n_j \mid X_i \mid X_c \mid X_{ij} \}

4) \( U_i \rightarrow \text{IdP} \cdot \{\text{HMAC} \mid n_d \}

5) \( IdP \rightarrow U_i \cdot \{\text{Token}_1 \mid n_d \}

6) \( U_i \rightarrow \text{Trustee} \cdot \{\text{Na}_i \} \mid \text{OTP} \mid n_d \cdot x \}

7) \( \text{Trustee} \rightarrow U_i \cdot \{n_j \cdot k \}

**Parsing of the authentication protocol into \( E_i \cdot \cdot CP_i \) and \( E_j \cdot ^*X \) is given below.**

1) \( \text{Trustee} \cdot \text{IdP} \cdot \{\text{Id}_j \mid \text{Na}_j \} \rightarrow U_i \equiv U_i \)

2) \( \text{IdP} \cdot \{\text{ERN} \mid n_j \cdot k \} \rightarrow \text{Trustee} \cdot \{\text{Tr} \}

3) \( U_i \cdot \{\text{Na}_j \} \rightarrow \text{IdP} \equiv \text{IdP} \rightarrow U_i \)

4) \( \text{IdP} \cdot \{\text{HMAC} \mid n_d \cdot x \} \rightarrow U_i \equiv U_i \cdot \{\text{Tr} \}

5) \( U_i \cdot \{\text{Na}_j \} \rightarrow \text{IdP} \equiv \text{IdP} \rightarrow U_i \)

6) \( \text{Trustee} \cdot \{\text{Trustee} \cdot \{\text{Na}_j \} \mid \text{OTP} \mid n_d \cdot x \}

7) \( U_i \cdot \{\text{Tr} \}

A. **Goals**
The followings are the goals which describe the basic functionalities of the proposed protocol.

1) Authentication on message content
In the proposed protocol, Trustee_k believes that user login request contents are recognizable and valid
\[ Trustee_k \equiv \phi \{ID_{ui}^*, [CSP_{idr}^*]|n_j, k\} |X_j| \]

In the second flow, \( IdP_j \) believes that Trustee_k message contents are recognizable and valid
\[ IdP_j \equiv \phi \{ID_{ui}^*, [ID_{rtn}^*]|n_j, k\} |X_j| |X_{cj}| \]

In the third flow, \( U_i \) believes that \( IdP_j \) message contents are recognizable and valid
\[ U_i \equiv \phi \{ERN||n_{j,k}, |X_j||X_j||X_{cj}|X_{cj}| \]

In fourth flow, \( IdP_j \) believes that \( U_i \) response contents are recognizable and valid
\[ IdP_j \equiv \phi \{[Token_{rtn}]|NA_{ui}^*||OTP||n_j, k\} |X_{c2}| \]

In fifth flow, \( U_i \) believes that \( IdP_j \) reply message contents are recognizable and valid
\[ U_i |\equiv \phi \{[Token_{rtn}]|n_{j,k}\} , X_j \]

In sixth flow, Trustee_k believes that \( U_i \) message contents are recognizable and valid
\[ Trustee_k |\equiv \phi \{[Token_{rtn}]|NA_{ui}^*||OTP||n_j, k\} |X_{c3}| \]

In seventh flow, \( U_i \) believes that Trustee_k reply message contents are recognizable and valid
\[ U_i \equiv \phi \{n_{j,k}\} \]

2) Authentication on message origin
From the login request, Trustee_k believes that \( U_i \) is originated the following message:
\[ Trustee_k |\equiv U_i \sim \{ID_{ui}^*|CSP_{idr}^*|n_j, k\} |X_j| \]

In the second flow, \( IdP_j \) believes that Trustee_k redirected the user with following message:
\[ IdP_j |\equiv Trustee_k \sim \{ID_{ui}^*, [ID_{rtn}^*]|n_j, k\} |X_j| |X_{c3}| \]

In the third flow, \( U_i \) believes \( IdP_j \) is replied
\[ U_i |\equiv IdP_j |\sim \{ERN||n_{j,k}, |X_j||X_j||X_{c2}|X_{c2}| \]

In the fourth flow, \( IdP_j \) believes \( U_i \) is replied
\[ IdP_j \equiv U_i \sim \{[HMAC^*||n_{j,k}\} \]

In the fifth flow, \( U_i \) believes and validates \( IdP_j \) response
\[ U_i \equiv IdP_j \sim \{[Token_{rtn}]|n_{j,k}\} \]

In the sixth flow, Trustee_k believes that \( U_i \) is replied
\[ Trustee_k |\equiv U_i \sim \{[Token_{rtn}]|NA_{ui}^*||OTP||n_j, k\} |X_{c3}| \]

In the seventh flow, \( U_i \) believes and validates Trustee_k response
\[ U_i \equiv Trustee_k \sim \{[Token_{rtn}]|n_{j,k}\} \]

3) Mutual Identity Verification
From the first flow, Trustee_k believes and verifies \( ID_{ui}^* \) and \( CSP_{idr}^* \) if identities are valid and matched then Trustee_k sends \( ID_{ui}^*, ID_{rtn}^* \) and \( n_j \) to \( IdP_j \) otherwise user request will be terminated
\[ Trustee_k |\equiv U_i \exists (ID_{ui}^*) \]

From the second flow, \( IdP_j \) believes and verifies \( ID_{ui}^* \) and \( ID_{rtn}^* \) if identities are valid and found, then \( IdP_j \) sends the intermediate secrets and encrypted random number (ERN) and \( n_j \) to \( U_i \), otherwise authentication request will be terminated
\[ IdP_j |\equiv U_i \exists (ID_{ui}^*) ) \& & Trustee_k \exists (ID_{rtn}^*) \]

From the third flow, \( U_i \) verifies \( ERM \) and \( n_j \) if \( n_j = n_{j+1} > 1 \text{ mod } n \), then user believes that the response received is genuine, otherwise authentication process will be stopped
\[ U_i \equiv Trustee_k \equiv CSP \exists (n_{j+1}) \]

From the fourth flow, \( IdP_j \) verifies \( U_i \) incremented nonce value and \( HMAC^* \), if \( n_{j+1} = n_{j+1} > 1 \text{ mod } n \) and MAC address is matched, then \( IdP_j \) believes that the response received from \( U_i \) is genuine, otherwise the authentication process will be terminated
\[ IdP_j \equiv U_i \exists (HMAC^*, n_{j+1}) \]

From the fifth flow, \( U_i \) verifies \( IdP_j \) incremented nonce \( n_{j+1} \), if \( n_{j+1} = n_{j+1} > 1 \text{ mod } n \), then \( U_i \) believes that the response received from \( IdP_j \) is genuine and user is allowed to enter OTP and forms \( C_2 = E_k (Token_{rtn})|NA_{ui}^*||OTP||n_j||X_j \); otherwise the authentication process will be terminated
\[ U_i \equiv IdP_j \exists (Token_{rtn}, n_{j+1}) \]

From sixth flow, Trustee_k believes that \( U_i \) message contents are recognizable and valid
\[ Trustee_k |\equiv U_i \exists (Token_{rtn}, NA_{ui}^*, OTP \text{ and } n_{j+1}) \]

In seventh flow, \( U_i \) believes that Trustee_k reply message contents are recognizable and valid, then \( U_i \) checks for mutual authentication value as \( n_j = n_{j+1} > 1 \text{ mod } n \), if it matches, then user is allowed to access the cloud services. Otherwise, the request will be rejected
\[ U_i \equiv Trustee_k (Token_{rtn}, NA_{ui}^*, OTP \text{ and } n_{j+1}) \]

B. Session Key Material Establishment
\( U_j, IdP_j \) and Trustee_k believes each other that \( X_o, X_j \) and \( X_k \) are their intermediate secrete values for generating shared session key
\[ \equiv \exists U_i \equiv IdP_j |\equiv \{U_i, IdP_j, Trustee_k \exists \{X_o, X_j, X_k \} \]

\( U_j, IdP_j \) and Trustee_k believes that \( K \) is a shared one-time secrete key for the current session
\[ \equiv IdP_j |\equiv Trustee_k |\equiv \{U_j \leftarrow \exists IdP_j \rightarrow IdP_j \leftarrow \exists Trustee_k \rightarrow U_i \leftarrow \exists Trustee_k \} \]

C. Assumption List
We consider the following assumptions in our authentication protocol.

- Trustee_k chooses a random values as private key \( -K \), computes corresponding public key \(+K \) and prepares a one-time intermediate secrete value \( X_i \) for generating shared session key
  \[ Trustee_k \equiv +K, Trustee_k \equiv K, Trustee_k \equiv X_i \]

- Trustee_k publishes a public key \(+K \) for the users and identity providers to encrypt their communication messages and also believes that \(+K \) is suitable for \( IdP_j \) and \( U_i \)

\[ Trustee_k \equiv +K, \equiv +K \]

© 2018, IJCSE All Rights Reserved
• $IdP_j$ chooses a random value as private key $\rightarrow K$, computes corresponding public key $+K$ and prepares a one-time intermediate secret value $X_j$ for generating shared session key

$$IdP_j \equiv K, IdP_j \equiv +K, IdP_j \equiv X_j.$$

• $IdP_i$ publishes a public key $+K$ for the users to encrypt their communication parameters and believes that $+K$ is suitable for $U_i$.

$$IdP_i \equiv +K, \{U_i\}.$$

• $U_i$ chooses a random value ‘$x$’ and prepares one-time intermediate secret value $X_i$. $U_i$ believes that $X_i$ is fresh and it will be used by $Trustee_k$ and $IdP_j$ to compute shared secret keys

$$U_i \equiv X_i, U_i = \#(X_i).$$

• $IdP_j$ chooses a random value ‘$y$’ and prepares one-time intermediate secret value $X_i$. $IdP_j$ believes that $X_i$ is fresh and it will be used by $Trustee_k$ and $U_i$ to compute shared secret keys

$$IdP_j \equiv X_j, IdP_j \equiv \#(X_i).$$

• $Trustee_k$ chooses a random value ‘$z$’ and prepares one-time intermediate secret value $X_k$. $Trustee_k$ believes that $X_k$ is fresh and it will be used by $IdP_j$ and $U_i$ to calculate shared secret keys

$$Trustee_k \equiv X_k, Trustee_k = \#(X_k).$$

VII. PERFORMANCE EVALUATION

In this section we establish the tested simulation using Microsoft Azure Compute and Storage Emulator. Using simulation platform we determine an effectiveness of the proposed protocol in terms of number of cryptographic operations are required, Resistance to various possible attacks, communication and computation costs. In first subsection, we present security comparisons. Next, we analyse the computational efficiency of our scheme with an existing schemes.

Setup: We have implemented our proposed investigation on a computer which has windows 7 operating system with 4GB RAM and 2.0GHz Intel Core i7 processor. C#.NET framework was installed on this computer which contains Visual Studio community 2013 as a frontend, SQL Server 2012 R2 SP1 as a backend and a Windows Azure Emulator as software platform.

A. Security Comparisons

In this subsection, first we compare proposed authentication protocol with the existing mechanisms [17]-[22], [27]-[28] in terms of mutual authentication, resistant to various reply and impersonation attacks, and trust and reputations management attacks. As presented in Table II, the existing mechanisms [17]-[22] and [27]-[28] are effortlessly to protect trust and reputation management attacks such as white-wash attack, collusion attack, bad mouth attack and good mouthing attack.

The mechanisms described in [17]-[22] are not suitable for collaborative cloud service providers. Existing mechanisms presented in [20]-[27] are not resistance to reply and impersonation attacks. The schemes presented in [17] and [20]-[22] are unable support mutual authentication. However, the mechanisms described in [27]-[28] are unable support provision of user anonymity. Therefore, our investigation meets all the design goals and is immune to various reply and impersonation attacks.

<table>
<thead>
<tr>
<th>Table 2. Comparisons with Existing Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>CP1</td>
</tr>
<tr>
<td>CP2</td>
</tr>
<tr>
<td>CP3</td>
</tr>
<tr>
<td>CP4</td>
</tr>
<tr>
<td>CP5</td>
</tr>
<tr>
<td>CP6</td>
</tr>
<tr>
<td>CP7</td>
</tr>
<tr>
<td>CP8</td>
</tr>
<tr>
<td>Merritt Max. [17]</td>
</tr>
<tr>
<td>Jannain. [18]</td>
</tr>
<tr>
<td>SIAM [19]</td>
</tr>
<tr>
<td>David G.[20]</td>
</tr>
<tr>
<td>John T.[21]</td>
</tr>
<tr>
<td>LoginRadius [22]</td>
</tr>
<tr>
<td>Jia-Lun T et al.[27]</td>
</tr>
<tr>
<td>Debiao H et al.[28]</td>
</tr>
<tr>
<td>Our’s</td>
</tr>
</tbody>
</table>


Next, we compare the computation costs of our authentication protocol with an existing scheme [28]. Let $T_{bp}$ be the bilinear pairing operation time, $T_b$ is one-way hash operation time, $T_c$ and $T_i$ are concatenation and Exclusive-OR operation times and, $T_i$ and $T_m$ are inverse and additive multiplication operation times respectively. The comparison of the computation cost of our scheme with an existing mechanism [28] is listed in Table III. In general, concatenation and bitwise Exclusive-OR operations are much faster and will consume constant timings, so that these two operations time can be neglected in calculating computation cost. Therefore, for registration process, our scheme requires two hash and one Exclusive-OR operations (i.e., $2T_b + T_c$). On the other hand, Debiao He et al. [28] scheme consumes two bilinear pairing, two hash, two multiplication and two inverse operations (i.e.,$2T_{bp} + 2T_b + 2T_m + 2T_i$).

For authentication process, proposed protocol consumes $3T_{bp} + T_b$ and Debiao He et al. scheme requires $7T_{bp} + 6T_b + 4T_m + 2T_i$. So, the total computation cost of our authentication mechanism is $O(3T_{bp} + 2T_b + 3T_m + 9T_i + 2T_c)$ and Jia-Lun T et al. scheme consumes $O(7T_{bp} + 6T_b + 4T_m + 2T_i + 5T_c + 2T_i)$.

| Table 3. Computation Cost Comparisons with Existing Scheme |
|--------------|-----------------|-----------------|
| Phase       | Existing Scheme | Our scheme |
| Registration | $T_{bp} + T_b + T_m + T_i$ | Nil |

© 2018, IJCSE All Rights Reserved
The registration and authentication phase running time of our scheme is recorded in Table IV for different elliptic curves over prime fields. Here, we have considered 300 and 41442 records in registration and authentication phases respectively. The proposed scheme consumes less running time for Elliptic Curves Diffie-Hellman P521. The overall computation cost comparison of our scheme with Jia-Lun T et al. [27] and Debiao H et al. [28] is recorded in Table V for different elliptic curves over prime fields. The proposed scheme consumes less computation cost compare with Jia-Lun T et al.[27] and Debiao H et al. [28] schemes. The overall communication cost comparison of our scheme with Jia-Lun T et al.[27] and Debiao H et al. [28] is listed in Table VI for different elliptic curves over prime fields. The proposed scheme consumes less communication cost compare with Jia-Lun T et al.[27] and Debiao H et al. [28] schemes. Therefore, we can conclude that our proposed authentication scheme is computationally efficient and robust towards various reply and impersonation attacks than the existing schemes.

VIII. CONCLUSION AND FUTURE DIRECTION

In this article, we developed a robust and an efficient mutual authentication model for verifying genuine communication entities in the cloud using n-party Diffie-Hellman bilinear key distribution and random nonce. User credentials and access keys are never revealed to the malicious users. Stakeholders can gain the control over the cloud environment. Experimental results and performance analysis shows that the proposed work is computationally efficient for mutual authentication and robust against the impersonation and ephemeral secret leakage attacks. However, this investigation can be further extended to reduce trustee participation overhead using tokenization techniques.

Declarations

Competing Interests

The authors Mr. Sabour Nagaraju and Dr. S.K.V. Jayakumar declare that they have no competing interests.

Authors’ Contributions

Mr. Sabour Nagaraju has made substantial contributions to conception, design, implementation, acquisition of test data, and performed experimental evaluation. Dr. S.K.V. Jayakumar has involved in revising it critically for important intellectual content, supervision of the research work and has given final approval of the version to be published. Both authors read and approved the final manuscript.

Acknowledgements

Thanks to the Pondicherry University administration for providing required hardware and software resources to carry out this work successfully.

REFERENCES

Authors' information

Sabout Nagaraju is currently working as assistant professor in Pondicherry University. He is graduated from G.Pulla Reddy Engineering College, Kurnool and did his post graduation at National Institute of Technology, Calicut. He is pursuing his Ph.D. from Pondicherry University, India. His professional experience spans over 11 years in various Engineering colleges and software industry. His areas of interest include Cloud Computing, Internet of Things and Big Data. He has published eight papers in international journals and seven papers in national/ international conferences.

Dr. S.K.V. Jayakumar is currently working as assistant professor in Pondicherry University. He obtained his B.E in Electrical and Electronics Engineering from Madurai Kamaraj University and did his M.E. in Computer Science and Engineering from Madras University. He has obtained his PhD from Pondicherry University. His teaching experience spans over 19 years and his research interest includes Web Services computing and cloud computing. He has published 29 research papers in International Journals and National and International Conferences.