Received 30 April 2022, accepted 17 June 2022, date of publication 27 June 2022, date of current version 7 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186683

A Secure User Authentication Protocol for Heterogeneous Mobile Environments

ALZUBAIR HASSAN[1,2], RAFIK HAMZA[3], FAGEN LI[4], (Member, IEEE), AWAD ALI[5], MOHAMMED BAKRI BASHIR[6,7], SAMAR M. ALQHTANI[8], TAWFEEG MOHMMED TAWFEEG[9], AND ADIL YOUSIF[5]

[1]Department of Computer Science, School of Computer Science and Informatics, University College Dublin, Dublin, D04 V1W8 Ireland
[2]Lero-The Irish Software Research Centre, University of Limerick, Limerick, V94 NYD3 Ireland
[3]Institute for International Strategy, Tokyo International University, Saitama 350-1197, Japan
[4]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[5]Department of Computer Science, College of Science and Arts-Sharourah, Najran University, Sharourah 68341, Saudi Arabia
[6]Department of Math, Turubah University College, Taif University, Taif 26571, Saudi Arabia
[7]Department of Computer Science, Faculty of Computer Science and Information Technology, Shendi University, Shendi 41601, Sudan
[8]Department of Information Systems, College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia
[9]Department of Computer Science, Faculty of Computer Science and Information Technology, University of Science and Technology, Khartoum 14411, Sudan

Corresponding authors: Alzubair Hassan (alzubair.mohamedtahir@ucd.ie) and Adil Yousif (ayalfaki@nu.edu.sa)

This work was supported by the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia under the Institutional Funding Committee at Najran University, Saudi Arabia, under Project NU/IFC/ENT/01/013.

ABSTRACT Mobile devices have become very important for our daily needs. The user authentication protocols with the key agreement are required to deal with the security issues that arise from the use of mobile devices through Internet applications. However, existing user authentication protocols are only suitable if the client and the server use a similar cryptographic approach. Therefore, it is important to develop an authentication protocol for mobile environments with heterogeneous cryptographic approaches. In this paper, an efficient user authentication and key agreement protocol is proposed for a heterogeneous client-server mobile environment. The security of the proposed scheme is formally proved under the q-strong Diffie-Hellman problem (q-SDH), the q-bilinear Diffie-Hellman inversion problem (q-BDHI), and the modified bilinear Diffie-Hellman inversion problem (mBDHI), respectively. Our scheme has reasonable processing costs and communication costs on the client and server sides. Moreover, our scheme is suitable for applications that use different cryptographic approaches. In particular, the proposed protocol can work when the client applies the identity-based cryptosystem and the server applies the certificateless cryptosystem.

INDEX TERMS User authentication, key agreement, random oracle model, heterogeneous environment.

I. INTRODUCTION

Previously, larger devices such as laptops and PCs were preferred over smaller devices such as cell phones and tablets. Today, however, people prefer mobile devices because they can be used for a variety of applications such as e-commerce, e-banking, e-healthcare, and e-government [1]–[3]. Mobile applications can work in different architectures, for example, in a client-server environment and in a multi-server environment [4], [5]. In a client-server architecture, the mobile device represents the client for accessing the services offered by the server. Separately, mobile device applications operate in an open network by using the Internet, which raises security issues such as authentication, key agreement, and confidentiality [6]–[8]. The client and server must authorize each other. In addition, the two parties must agree on a session key for subsequent connections [9].

To overcome the drawbacks of authentication, key agreement, and confidentiality, various public-key cryptosystems have been proposed, such as the public-key infrastructure cryptosystem (PKI) [10], the identity-based cryptosystem (IBC) [11], and the certificateless cryptosystem (CLC) [12]. However, PKI is very computationally intensive due to certificate management and is therefore not suitable for mobile

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Arshad.

environments. IBC appears to reduce the computational overhead of PKI and is more suitable for mobile environments, but lacks efficiency in key escrow. The CLC, on the other hand, eliminates both the need for the certificate in PKI and the key escrow problem in IBC [13].

Certain user authentication schemes with key agreement have been proposed to solve the user authentication and key agreement issues [14]–[16]. However, all protocols are homogeneous, i.e., the client and the server all belong to a similar cryptographic approach. For example, the clients get their keys from the server, which mainly uses IBC or CLC. With the rapid growth of mobile applications and the daily needs of users, it is important to have heterogeneous architectures that can work with different cryptographic approaches. For example, clients get their keys from the server using IBC and servers get their keys fromthe server using CLC. In this context, mobile applications using the Internet with different cryptographic environments may need to communicate with each other to access services.

Recently, Li *et al.* [17] proposed a heterogeneous user authentication scheme that takes into account when the client and the server use different cryptosystems. In their protocol, the client uses the IBC environment and the server is from PKI. However, their scheme has the disadvantage of causing certificate management overhead, which is unsuitable for mobile devices with limited memory and power. Therefore, existing systems need to be vastly improved or new ones created. For example, some applications require the client to request an IBC while the server requests a CLC. In this case, it is highly desirable to use a user authentication protocol that meets the requirements of this environment. In this paper, we propose a secure user authentication protocol for heterogeneous mobile environments.

II. RELATED WORKS

To tackle the authentication issues in mobile environments, several identity-based user authentication schemes have been suggested without mutual authentication and key agreement [18], [19].

To address authentication issues in mobile environments, several identity-based user authentication schemes without mutual authentication and key agreement have been proposed [18], [19]. Similarly, many identity-based user authentication protocols with the key agreement have been introduced to overcome the weaknesses of the previous schemes in terms of computational cost, communication cost, and security vulnerabilities. Wu and Tseng [14] have presented a new user authentication protocol based on IBC and provided a formal proof of their protocol. In addition, their protocol provides key agreement and mutual authentication. In terms of computational cost improvement, He [20] proposed another user authentication protocol with formal proof that has the same security properties as Wu and Tseng's scheme [14], resulting in a computationally effective and efficient scheme compared to that of Wu and Tseng's [14]. Chou *et al.* [21] proposed a two-party scheme AKA for the mobile application environment. This scheme is based on identity with ECC.

In addition, they extended their work to design another three-party scheme AKA for initiating the session key between trusted service providers and users. They claimed that their proposed scheme resists common attacks on user authentication schemes, such as perfect forward secrecy, known key security, and key identity compromise impersonation [22]. Later, Farash and Attari [23] found that Chou's schemes *et al.* [21]are not as resistant to the above attacks as they claimed. They introduced an improved identity-based AKA scheme to overcome the drawbacks of Chou *et al.*'s scheme [21].

Tsai and Lo [24] considered the anonymous property of the client identity which led them to design anonymous user authentication protocol for mobile device applications. Furthermore, their scheme has less processing cost on the client side than He's scheme. Tseng *et al.* [25] also considered ephemeral secret leakage (ESL) attacks which resulted in designing a user authentication protocol which resists these attacks. In all the schemes above, the client and the server belong to the IBC. Because the private keys in IBC are generated by trust party called the private key generator (PKG), these schemes suffer from the key escrow problem. Consequently, if the PKG is compromised by an adversary, then all clients' private keys are jeopardized.

Since the certificateless cryptosystem (CLC) [26] is proposed to overcome the key escrow problem in the IBC, Hou *et al.* [27] suggested the scheme of user authentication protocol without achieving mutual authentication which used CLC. The proposed protocols should be secure against the adversary type I and adversary type II as discussed in [26]. Aftermath, Hassan *et al.* [16] presented a new protocol based on CLC that offers key agreement and mutual authentication. They claimed that their scheme is secured from adversary types I and II respectively. Nevertheless, Hassan *et al.* [28] found that the protocol in [16] does not resist adversary type II. Then, they proposed another user authenticated key agreement protocol with mutual authentication which is indeed secure against adversary types I and II. In all the schemes above, the client and the server remains on the CLC.

After looking at all the previous user authentication protocols, we found that the client and the server belong to the same cryptosystem environment. For example, the client and the server belong to the IBC or the CLC. However, if an application needs to work in different cryptosystem environments (eg., the client belongs to the PKI and the server belongs to CLC), it doesn't requires such user authentication protocol. Recently, Li *et al.* [17] proposed a heterogeneous user authentication protocol of mobile client-server. In their scheme, the client belongs to the IBC and the server belongs to the PKI. In this paper, user authentication with the key agreement protocol is presented for the heterogeneous mobile client-server environment. In our proposed scheme, the client applies to the IBC, while the server applies to the CLC.

A. ORGANIZATION

Our paper is presented as follows: The preliminaries are introduced in Section III while our protocol is proposed in Section IV. The security of the presented protocol is displayed in Section V and the analysis of the protocol's performance is demonstrated in Section VI with the conclusions in Section VII.

III. PRELIMINARIES

A. BILINEAR PAIRINGS

Let $\mathbb{G}_1$ be a cyclic additive group and $\mathbb{G}_2$ be a cyclic multiplicative group with a large prime order q. Let P be a generator of $\mathbb{G}_1$. Therefore, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ realizes the following properties [29], [30]:

1) **Bilinearity :** $\forall x, y \in \mathbb{Z}_q^*$ and $\forall Q, R \in \mathbb{G}_1$, $e(xQ, yR) = e(Q, R)^{xy}$.
2) **Non-degeneracy:** Let P be a generator of $\mathbb{G}_1$, $e(P, P) \neq 1_{\mathbb{G}_2}$. Where $1_{\mathbb{G}_2}$ is identity element of $\mathbb{G}_2$.
3) **Computability:** $\forall Q, R \in \mathbb{G}_1$, the $e(Q, R)$ is processed efficiently.

1) q-STRONG DIFFIE-HELLMAN (q-SDH) PROBLEM

Given $(P, \alpha P, \alpha^2\ P, \ldots, \alpha^q P)$, the q-strong Diffie-Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is to produce a pair $(w, \frac{1}{\alpha+w})$ where $w, \alpha \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$.

2) q-BILINEAR DIFFIE-HELLMAN INVERSION PROBLEM

Given $(P, \alpha P, \alpha^2\ P, \ldots, \alpha^q P)$ the q-bilinear Diffie-Hellman inversion problem (q-BDHIP) in $(\mathbb{G}_1, \mathbb{G}_2)$ is to produce $e(P, P)^{\frac{1}{\alpha}}$ as hard assumption.

3) MODIFIED BILINEAR DIFFIE-HELLMAN INVERSION PROBLEM

Given $(P, \alpha P, \gamma)$ the modified bilinear Diffie-Hellman inversion problem (mBDHIP) in $(\mathbb{G}_1, \mathbb{G}_2)$ is to produce $e(P, P)^{\frac{1}{\alpha+\gamma}}$ as hard assumption.

B. NETWORK ARCHITECTURE

Figure 1 depicts our scheme's network architecture. The architecture comprises the client, a service provider (SP), and an Internet server. The client employs the IBC to communicate with the server, while the server employs the CLC. In IBC, the private key generator (PKG) generates the private keys of the clients, while the clients' partial private keys are generated by the key generator center (KGC). Here, the SP plays the role of PKG for the clients and the role of the KGC for the server. Our protocol is capable of working in this environment. It is assumed that the SP is trusted and cannot be attacked [17].

C. SECURITY MODEL

In this subsection, the security authentication and key agreement as well as the capabilities of the adversary $\mathcal{A}$ are presented. The symbol Π_u^λ denotes as an instance λ of a

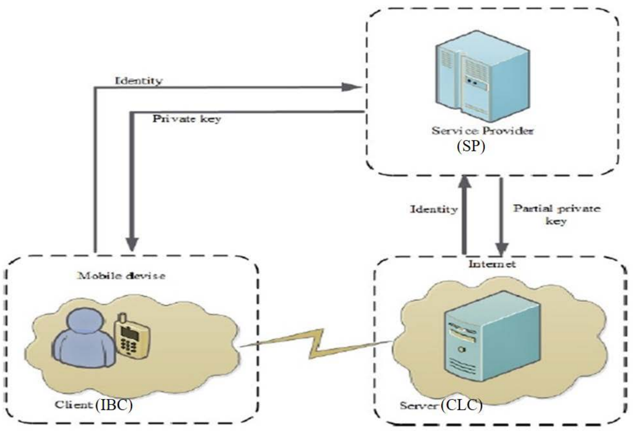

FIGURE 1. Network architecture.

member u. In the following, we show how the challenger $\mathcal{C}$ reacts with the adversary $\mathcal{A}$:

1) *Setup*(1^λ): Taking as input a security parameter λ. $\mathcal{C}$ runs the *Setup* algorithm to create the master secret key $x \in \mathbb{Z}_q^*$, the master public key P_{pub} as well as the system parameters *params*. Afterwards, $\mathcal{C}$ sends *params* to $\mathcal{A}$ and remains x secret.
2) *Probing*: At any time $\mathcal{A}$ can request the following queries in an adaptive manner:
    a) *Extract partial private key* queries: For any identity ID, $\mathcal{A}$ can request the partial private key. The corresponding partial private key D_{ID} is calculated by $\mathcal{C}$ and is reverted to $\mathcal{A}$ consequently.
    b) *Extract private key* queries: $\mathcal{A}$ allows to request the private key for any ID. The corresponding private key is computed by $\mathcal{C}$ and is returned to $\mathcal{A}$ consequently.
    c) *Request public key* queries: Upon $\mathcal{A}$ requests the public key for any ID, $\mathcal{C}$ calculates public key PK_{ID} associated with ID and returns PK_{ID} to $\mathcal{A}$.
    d) *Replace public key* queries: a new secret value v' for any ID, can be chosen by $\mathcal{A}$. Then, using v' a new public key is computed by $\mathcal{A}$ consequently. Thus, the PK_{ID} can be replaced by $\mathcal{A}$ with PK'_{ID}.
    e) *Send*(Π_u^λ, m) queries: while a plaintext m is submitted depending on the introduced protocol from $\mathcal{A}$ to $\mathcal{C}$, the estimation and the response to $\mathcal{A}$ are made by $\mathcal{C}$.
    f) *Reveal* (Π_u^λ) queries: $\mathcal{A}$ may access to the session key *sk* from $\mathcal{C}$, when $\mathcal{C}$ accepts. If $\mathcal{C}$ does not accept $\mathcal{A}$'s session key, $\mathcal{C}$ returns a null.
    g) *Corrupt*(u) queries: When a member's u private key is remitting, $\mathcal{A}$ makes a *Corrupt* query to $\mathcal{C}$.
    h) *Test* (Π_u^λ) queries: After $\mathcal{A}$ sends one *Test* query, $\mathcal{C}$ throws a coin b. If $b = 1$, $\mathcal{A}$ obtains the session key. Otherwise, an arbitrary string is returned. This query gives the semantic security of the session key.

TABLE 1. Notation.

Symbol	Description
λ	A security parameter
$\mathbb{G}_1$	A cyclic additive group
$\mathbb{G}_2$	A cyclic multiplicative group
q	A prime order of group $\mathbb{G}_1$ and $\mathbb{G}_2$
P	A generator of $\mathbb{G}_1$
e	A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
H_i	One way function, $\forall i \in \{1, 2, 3\}$
x	The SP's master secret key
P_{pub}	The SP's master public key
ID_c	The client's identity
ID_s	The server's identity
SK_c	The client's private key
D_{ID_s}	The server's partial private key
SK_s	The server's full private key
PK_s	The server's public key
ID_t	A challenged identity

Afterwards, $\mathcal{A}$ generates b' as estimation for b. Mathematically, the advantage of $\mathcal{A}$ is represented by Adv($\mathcal{A}$). Therefore, Adv($\mathcal{A}$) $= |Pr[b' = b] - 1/2|$, where $Pr[b' = b]$ represents the probability of $b' = b$.

In our security proof, *Extract private key* and *Send* queries are used to show that our scheme has client-to-server authentication. For the key agreement, we show that our scheme is secure against adversary type I and adversary type II due to the server belongs to the CLC. *Extract partial private key*, *Extract private key*, *Request public key*, *Replace public key*, *Send*, *Reveal*, *Corrupt*, and *Test* queries are used for adversary type I. *Extract private key*, *Request public key*, *Send*, *Reveal*, *Corrupt*, and *Test* queries are used for adversary type II.

IV. PROPOSED PROTOCOL

The symbols used in our paper are displayed in Table 1. Our scheme is illustrated by the phases: Setup, Key Extraction, as well as User Authentication with Key agreement. The proposed scheme lets the client and the server register by their identity in the SP. Note that the client applies the IBC and the server applies the CLC. To be sure that our scheme provides the user authentication and key agreement requirement, the proposed scheme follows the following steps. First, in the Setup phase, the SP prepares the public parameters that will be used in the communication between the client and the server. Second, the SP runs the extract phase to generate the corresponding keys for the client and the server respectively. Third, the client computes a value and sends this value with his identity to the server. Then, the server checks this value which is correct or not. Fourth, after verify the value sent by the client, the server computes other values and sends it to the client. By this step, the server can know whether is communicating with the right server or not. Fifth, the client receives the server's values and used them to compute the session key and generate a signature and send it to the server. Finally, the server receives the client's signature and checks if it's correct will accept the client otherwise reject the client.

A. SETUP PHASE

The service provider (SP) executes this phase by the following steps :

1) Take λ as input and calculate the *params*.
2) Pick two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$,where $\mathbb{G}_1$ is additive group and $\mathbb{G}_2$ is multiplicative group with the same prime order q and the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Let P be a generator of $\mathbb{G}_1$.
3) Pick the master secret key $x \in \mathbb{Z}_q^*$ randomly and let $P_{pub} = xP$, and pick secure hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ as well as $H_3 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$.
4) Issue the *params* $= \{\mathbb{G}_1, \mathbb{G}_2, q, e, P, P_{pub}, g, H_1, H_2, H_3\}$, where $g = e(P, P)$, as the public parameters and remain x secret.

B. KEY EXTRACTION

This algorithm is carried out by the SP as follows:

- IBC-KG: A client sends his identity ID_c to the SP. Then, the SP responds with $SK_c = (1/H_1(ID_c) + x)P$ as a client's private key. The client uses his identity ID_c as public key.
- CLC-KG: A server sends his identity ID_s to the SP. Then, the SP responds with $D_{ID_s} = (1/H_1(ID_s) + x)P$ as a server's partial private key. Then, the server with identity ID_s chooses randomly $v \in \mathbb{Z}_q^*$ as secret value to compute his public key as follows $PK_s = v(H_1(ID_s)P + P_{pub})$. When the D_{ID_s} and the v are given, the server computes his full private key as follows $SK_{ID_s} = \frac{1}{v+H_2(PK_s)} D_{ID_s}$.

C. USER AUTHENTICATION WITH KEY AGREEMENT

Figure 2 depicts the collaboration among the client and the server to do the verification. Here, this phase gives the ability to the client who applies the IBC to authenticate from himself/herself in the server, while the server applies the CLC. Indeed, this step illustrates the authentication in heterogeneous cryptographic approaches. This phase can be completed by the following steps:

- While a client private key SK_c, a client public key ID_c, and the server public key PK_s are given, the client responds as follows:
  1) Select φ randomly from $\mathbb{Z}_q^*$.
  2) Compute $r = g^{\varphi}$.
  3) Compute $T = \varphi(PK_s + H_2(PK_s)(H_1(ID_s)P + Ppub))$ and return ID_c and T to the server.

  Note that the values of r and T are precomputed off-line by the client.
- After ID_c and T are received, the server responds by the following :
  1) Calculate $r = e(T, SK_s)$.
  2) Choose $\beta \in \mathbb{Z}_q^*$.
  3) Compute $h = H_3(ID_c, ID_s, PK_s, r, \beta)$.
  4) Return β and h to the client.

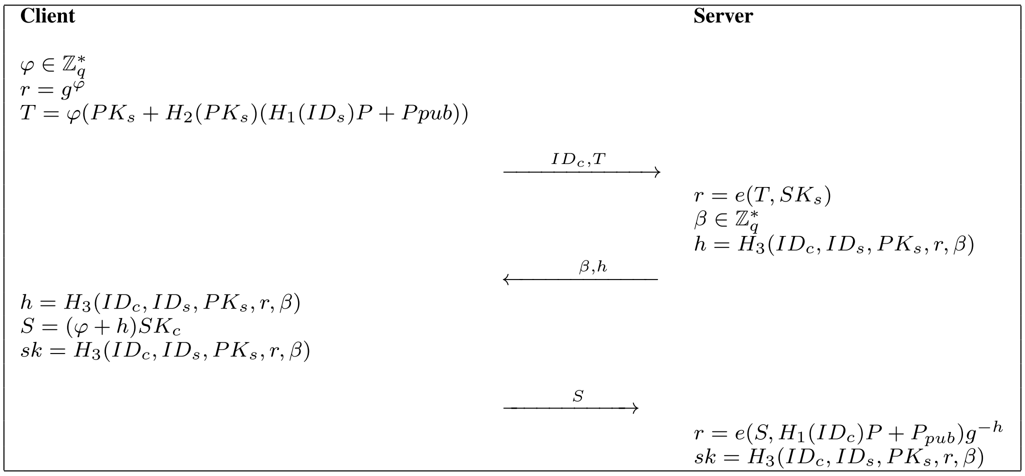
Client
Server
$\varphi \in \mathbb{Z}_q^*$
$r = g^{\varphi}$
$T = \varphi(PK_s + H_2(PK_s)(H_1(ID_s)P + Ppub))$
$\xrightarrow{ID_c, T}$
$r = e(T, SK_s)$
$\beta \in \mathbb{Z}_q^*$
$h = H_3(ID_c, ID_s, PK_s, r, \beta)$
$\xleftarrow{\beta, h}$
$h = H_3(ID_c, ID_s, PK_s, r, \beta)$
$S = (\varphi + h)SK_c$
$sk = H_3(ID_c, ID_s, PK_s, r, \beta)$
$\xrightarrow{S}$
$r = e(S, H_1(ID_c)P + P_{pub})g^{-h}$
$sk = H_3(ID_c, ID_s, PK_s, r, \beta)$

FIGURE 2. User authentication with key agreement.

- Since β and h are received, the client responds as follows:
    1) Verify if h is similar to the computed $H_3(ID_c, ID_s, PK_s, \beta, r)$ by the client.
    2) Compute $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$ as the session key that will be used between the client and the server for the future communication.
    3) Compute $S = (\varphi + h)SK_c$ and return it to the server.
- Since S is received, the server verifies from S using the following equation $r = e(S, H_1(ID_c)P + P_{pub})g^{-h}$. Then, the session key $sk = H_2(ID_c, ID_s, PK_s, \beta, r)$ is computed.

To insure that the proposed scheme is correct, first we have

$$\begin{aligned}
r &= e(T, SK_s) = e(\varphi PK_s, \frac{1}{v + H_2(PK_s)} D_{ID_s}) \\
&= e(\varphi(PK_s + H_2(PK_s)(H_1(ID_s)P + Ppub)), \\
&\quad \frac{1}{v + H_2(PK_s)} \cdot \frac{1}{H_1(ID_s) + x} P) \\
&= e(\varphi(v(H_1(ID_s)P + P_{pub}) + H_2(PK_s)(H_1(ID_s)P + Ppub)), \\
&\quad \frac{1}{v + H_2(PK_s)} \cdot \frac{1}{H_1(ID_s) + x} P) \\
&= e((\varphi(v + H_2(PK_s))(H_1(ID_s)P + Ppub)), \\
&\quad \frac{1}{v + H_2(PK_s)} \cdot \frac{1}{H_1(ID_s) + x} P) \\
&= e((\varphi(v + H_2(PK_s))(H_1(ID_s) + x)P), \\
&\quad \frac{1}{v + H_2(PK_s)} \cdot \frac{1}{H_1(ID_s) + x} P) \\
&= e(\varphi P, P) \\
&= e(P, P)^{\varphi} \\
&= g^{\varphi} \\
&= r
\end{aligned}$$

Then, we verify the correctness of the following equation

$$\begin{aligned}
r &= e(S, H_1(ID_c)P + P_{pub})g^{-h} \\
&= e((\varphi + h)SK_c, H_1(ID_c)P + P_{pub})g^{-h} \\
&= e((\varphi + h)\frac{1}{H_1(ID_c) + x}P, H_1(ID_c)P + P_{pub})g^{-h} \\
&= e((\varphi + h)\frac{1}{H_1(ID_c) + x}P, (H_1(ID_c) + x)P)g^{-h} \\
&= e((\varphi + h)P, P)g^{-h} \\
&= e(P, P)^{(\varphi + h)}g^{-h} \\
&= g^{(\varphi + h)}g^{-h} \\
&= g^{\varphi} \\
&= r
\end{aligned}$$

V. SECURITY ANALYSIS

This section depicts that our protocol can achieve the client-to-server authentication (CSA), key agreement, and server-to-client authentication (SCA) by using the random oracle model [31]. We employed the logic in [14] to do the security analysis.

A. ANALYSIS OF CLIENT-TO-SERVER AUTHENTICATION

In theorem 1, $\mathcal{A}$ unable to use the client to complete the communication with the server. The $\mathcal{A}$ can not solve the q-SDH problem which is known as hard.

Theorem 1: Supposing that $\mathcal{A}$ takes a non-negligible advantage ε for breaking the CSA security. Also, at most q_S queries to the server's oracle Π_S^j, q_C queries to the client's oracle Π_C^i, Extract private key queries and q_{H_i} queries on H_i oracle $\forall i \in \{1, 2, 3\}$ are made by a challenger $\mathcal{C}$. Therefore $\mathcal{C}$ solves the q-SDH problem by having a non-negligible probability.

Proof: Figure 3 depicts the proof structure of this theorem. As the proof in [30], the challenger $\mathcal{C}$ takes as input $(P, \alpha P, \alpha^2 P, \ldots \alpha^p P)$ and attempts to extract $(w_i, \frac{1}{w_i + \alpha} P)$ from its communication with $\mathcal{A}$, where $w_i, \alpha \in \mathbb{Z}_q^*$.

Initialization: In IBC-KG preparation phase, $\mathcal{C}$ chooses randomly $w_1, w_2, \ldots, w_{p-1} \in \mathbb{Z}_q^*$. As in the proof technique of [30], $\mathcal{C}$ sets up a generator $Q \in \mathbb{G}_1$ and the public key $Q_{pub} = \alpha P \in \mathbb{G}_1$, such that it knows $p - 1$ pairs $(w_i, k_i = \frac{1}{w_i + \alpha} Q)$ for $i \in \{1, 2, 3, \ldots, p - 1\}$. To do so,

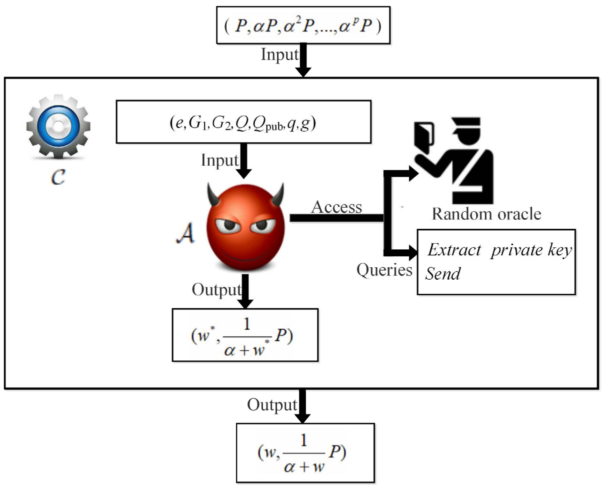
$(P,\alpha P,\alpha^2 P,...,\alpha^p P)$
Input
$(e,G_1,G_2,Q,Q_{pub},q,g)$
Input
$\mathcal{C}$
$\mathcal{A}$
Access
Random oracle
Queries
Extract private key
Send
Output
$(w^*,\frac{1}{\alpha+w^*}P)$
Output
$(w,\frac{1}{\alpha+w}P)$

FIGURE 3. The proof structure of Theorem 1.

the polynomial $f(z) = \Pi_{i=1}^{p-1}(z + w_i)$ is expanded by $\mathcal{C}$ to acquire the coefficients of $f(z)$, then $f(z) = \sum_{i=0}^{p-1} l_i z^i$ where $\{l_0, l_1, l_3, \ldots, l_{p-1}\} \in \mathbb{Z}_q^*$. After that, Q and Q_{pub} are computed by $\mathcal{C}$ using the following equations:

$$Q = \sum_{i=0}^{p-1} l_i(\alpha^i P) = f(\alpha)P \in \mathbb{G}_1$$

and

$$Q_{pub} = \sum_{i=1}^{p} l_{i-1}(\alpha^i P) = \alpha f(\alpha)P = \alpha Q$$

The pair (w_i, k_i) can be acquired similar to the proof in [30]. The following equations $f_i(z)$ are expanded by the $\mathcal{C}$ to get $f_i(z) = \frac{f(z)}{z+w_i} = \sum_{i=0}^{p-2} d_i z^i$ and k_i is taken as

$$k_i = \sum_{i=0}^{p-2} d_i(\alpha P) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha + w_i}P = \frac{1}{\alpha + w_i}Q$$

Q_{pub} and α are set as the master public key and master secret key of the SP, respectively. Then Q, Q_{pub}, and $g = e(Q, Q)$ are sent to the $\mathcal{A}$ by the $\mathcal{C}$. The public key and private key of the server PK_s, SK_s are computed by the $\mathcal{C}$ using the CLC-KG algorithm and are sent to $\mathcal{A}$. A challenged identity ID_t is chosen and is given to the $\mathcal{A}$ by the $\mathcal{C}$.

Attack: The algorithm $\mathcal{C}$ generates the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, e, Q, g, Q_{pub}, H_1, H_2, H_3\}$ and forwards to $\mathcal{A}$. $\mathcal{C}$ selects ID_t randomly. $\mathcal{C}$ simulates the random oracles of H_1, H_2 and H_3 with lists regarding to avoid collision and consistency. For oracles queries and responses, $\mathcal{C}$ prepares four lists L_{H_1}, L_{H_2}, L_{H_3} and L_K to keep the public keys. We assume $H_1(ID)$ query is completed first then the other queries are created.

1) H_1 queries: When $\mathcal{A}$ forwards $H_1(ID)$ query, $\mathcal{C}$ selects $w_i \in_R \mathbb{Z}_q^*$ as well as sends it to $\mathcal{A}$. However, if $ID = ID_t$, $\mathcal{C}$ sends w_t to $\mathcal{A}$. Then, $\mathcal{C}$ updates L_{H_1} with (ID, w).
2) H_2 queries: When $\mathcal{A}$ submits this query on PK_s, First of all $\mathcal{C}$ verifies if the value of H_2 was precedently defined for PK_s. If it was defined, $\mathcal{C}$ returns the value that was defined previously. Otherwise, $\mathcal{C}$ randomly chooses $h_{2,i} \in \mathbb{Z}_q^*$. Then, $\mathcal{C}$ returns $h_{2,i}$ to $\mathcal{A}$ and updates L_{H_2} with $(PK_s, h_{2,i})$.
3) H_3 queries: When $\mathcal{A}$ sends this query on $(ID_c, ID_s, PK_i, r_i, \beta_i)$, $\mathcal{C}$ firstly looks at L_{H_3} to verify that the value of H_3 was precedently assigned to the $(ID_c, ID_s, PK_i, r_i, \beta_i)$. If yes, the precedent value is returned by $\mathcal{C}$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly chooses $h_{3,i} \in \mathbb{Z}_q^*$. Then, $\mathcal{C}$ returns $h_{3,i}$ to $\mathcal{A}$ and updates L_{H_3} with $(ID_c, ID_s, PK_i, r_i, \beta_i, h_{3,i})$.
4) *Extract private key* queries: When $\mathcal{A}$ forwards this query to get the private key of ID, $\mathcal{C}$ does as follows:
   a) If $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation.
   b) If $ID \neq ID_t$, the private key of the client is known and is computed by the $\mathcal{C}$ with $k_i = 1/(\alpha + w_i)Q$.
5) *Send* queries:
   a) When $\mathcal{A}$ submits *Send* $(\Pi_C^i$, "start") query with the client identity ID_c, If $ID \neq ID_t$, $\mu \in \mathbb{Z}_q^*$ is chosen randomly by $\mathcal{C}$. Then, $r = g^{\mu}$ and $T = \mu(w_i Q + Q_{pub}) - h(PK_s + h_{2,i}(w_i Q + Q_{pub}))$ are computed. Finally, $\mathcal{C}$ returns ID_c and T to $\mathcal{A}$. If $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation.
   b) Whenever $\mathcal{A}$ forwards *Send* $(\Pi_S^j, (ID_c, T))$ query with (ID_c, T) to the server, if $ID \neq ID_t$, $\mathcal{C}$ computes $r = e(T, SK_s)$. Then, $\beta \in \mathbb{Z}_q^*$ is chosen randomly and $h = H_3(ID_c, ID_s, PK_s, r, \beta)$ is computed by $\mathcal{C}$. Finally, β and h are returned to $\mathcal{A}$. Otherwise, $\mathcal{C}$ fails and stops the simulation.
   c) When $\mathcal{A}$ submits *Send* $(\Pi_C^i, (\beta, h))$ query to the client, If $ID \neq ID_t$, $H_2(ID_c, ID_s, PK_s, r, \beta)$ is computed again by $\mathcal{C}$ to check whether it is equal to the h that is received from the server or not.Then, $S = \mu SK_s$ and $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$ are computed by $\mathcal{C}$. Finally, S is returned to $\mathcal{A}$. If $ID = ID_t$, $\mathcal{C}$ fails due to the $\mathcal{A}$ can not get the correct value of h'.
   d) Whenever $\mathcal{A}$ forwards *Send* (Π_S^j, S) query to the server, if $ID \neq ID_t$, $e(S, w_i Q + Q_{pub})g^{-h}$ is computed by $\mathcal{C}$ to check whether is equal to r. If the condition is satisfied, $\mathcal{C}$ accepts and computes $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$. Otherwise, the simulation is stopped by $\mathcal{C}$.

Analysis: Turing machine $\mathcal{A}'$ can be constructed if $\mathcal{A}$ is a forger in the above simulation, according to the forking lemma [32]. Then, $\mathcal{A}'$ is ables to sign messages (ID^*, h, S) and (ID^*, h^*, S^*) with $h \neq h^*$ and sed them to the challenger. A machine $\mathcal{C}$ is constructed to solve the q-SDHP by using $\mathcal{A}'$ as follows:

- $\mathcal{C}$ finds two various signatures (ID^*, h, S) and (ID^*, h^*, S^*) by running $\mathcal{A}'$.
- $k^* = (S - S^*)(h - h^*)^{-1} = 1/(\alpha + w^*)Q = \frac{f(\alpha)}{\alpha+w^*}P$ is computed by $\mathcal{C}$.

- The long division is used and the polynomial f is computed as $f(z) = \Psi(z) = (z+w^*)+\Psi-1$ by $\mathcal{C}$ for several $\Psi(z) = \sum_{i=0}^{p-2} \Psi_i z^i$ and some $\Psi - 1 \in \mathbb{Z}_q^*$. Then $\frac{f(z)}{z+w^*}$ can be computed as

$$\frac{f(z)}{z+w^*} = \Psi(z) + \frac{\Psi-1}{z+w^*} = \sum_{i=0}^{p-2} \Psi_i z^i + \frac{\Psi-1}{z+w^*}$$

Then, $\frac{1}{z+w^*}P$ is computed by $\mathcal{C}$ as follows

$$\frac{1}{z+w^*}P = \frac{1}{\Psi}(k^* - \sum_{i=0}^{p-2} \Psi_i(x^i P))$$

- The pair $(w^*, \frac{1}{\alpha+w^*}P)$ is outputted by $\mathcal{C}$ as answer of q-SDHP.

By the forking lemma [32], if $\mathcal{A}$ is succeeded in a time t with probability $\varepsilon \geqslant \frac{10(q_s+1)(q_s+q_{H_2})}{2^k}$. Then, the q-SDHP is solved by $\mathcal{C}$ with expected time

$$t' \le 120686\, q_{H_1} q_{H_2} \frac{t + O(q_s t_p)}{\varepsilon(1-1/2^k)(1-q/2^k)} + O(q^2 t_m).$$

Indeed, our scheme provides a client-to-server authentication under the q-SDHP.

B. ANALYSIS OF KEY AGREEMENT

Theorem 2: Suppose that the Test query coin can be predicted by $\mathcal{A}_I$ and $\mathcal{A}_{II}$ with a non-negligible advantage ε and at most q_S queries to the server's oracle Π_S^j, q_C queries to the client's oracle Π_C^i, Extract partial private key queries, Extract private key queries, Request public key queries, Replace public key queries, Send queries, Corrupt queries, Reveal queries, Test queries and q_{H_i} queries on H_i oracle $\forall i \in \{1, 2, 3\}$ are made by $\mathcal{C}$. Then, $\mathcal{C}$ solves the q-BDHI Problem and mBDHI problem with a non-negligible probability.

Proof:

Lemma 1: Our protocol resists to the adversary type I $\mathcal{A}_I$ under q-BDHI problem in the random oracle model.

Proof: Figure 4 depicts the proof structure of this theorem. When a *Test* query is submitted, adversary can guess the value of coin b correctly with probability not less than 1/2. Lets suppose that the adversary is able to guess the coin with ε. As result, adversary with advantage $\Pr[Esk] \ge \varepsilon/2$ enables to acquire the correct session key. Here, some symbols are employed in our proof. We use the symbol *Esk* to show the event of acquiring the correct session key by the adversary. In addition, these symbols $Test(C^i)$ and $Test(S^j)$ are used to explain the correct test queries that are made to the client and to the server separately. The symbol E^{C2S} represents the event of breaking the CSA security. By supposing that the *Test* query can be submitted to the client as well as to the server by adversary, subsequently this probability, into some i and j, is:

$$\Pr[Esk \wedge Test(\Pi_S^j) \wedge E^{C2S}] + \Pr[Esk \wedge Test(\Pi_S^j) \wedge \neg E^{C2S}] + \Pr[Esk \wedge Test(\Pi_C^i)] \ge \frac{\varepsilon}{2}$$

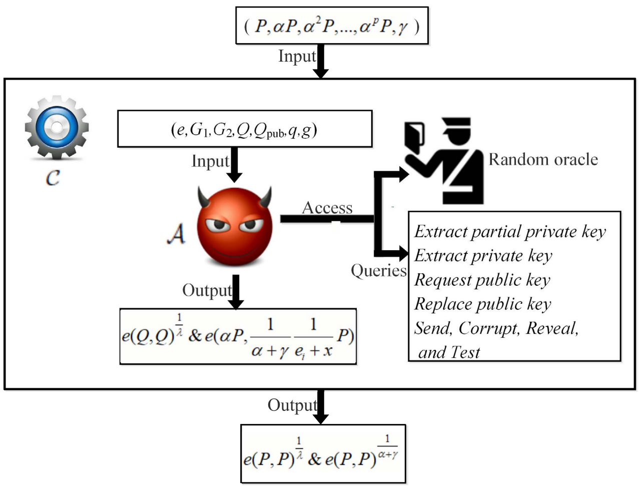
$(P, \alpha P, \alpha^2 P, ..., \alpha^p P, \gamma)$
Input
$\mathcal{C}$
$(e, G_1, G_2, Q, Q_{pub}, q, g)$
Input
$\mathcal{A}$
Access
Random oracle
Queries
Extract partial private key
Extract private key
Request public key
Replace public key
Send, Corrupt, Reveal, and Test
Output
$e(Q,Q)^{\frac{1}{\lambda}}$ & $e(\alpha P, \frac{1}{\alpha+\gamma}\frac{1}{e_i+x}P)$
Output
$e(P,P)^{\frac{1}{\lambda}}$ & $e(P,P)^{\frac{1}{\alpha+\gamma}}$

FIGURE 4. The proof structure of Theorem 2.

Assuming the possibility of breaking the CSA security is denoted by $\Pr_{C2S}$. Then, the following probability is computed into some i and j

$$\Pr[Esk \wedge Test(\Pi_C^i) \wedge \neg E^{C2S}] + \Pr[Esk \wedge Test(\Pi_C^i)] \ge \frac{\varepsilon}{2} - \Pr_{C2S}$$

As the proof in [30], the challenger $\mathcal{C}$ takes as input $(P, \alpha P, \alpha^2 P, ...\alpha^p P)$ and attempts to extract $(w_i, \frac{1}{w_i+\alpha}P)$ from its communication with $\mathcal{A}_I$. Where $w_i, \alpha \in \mathbb{Z}_q^*$

Initialization: In preparation phase, $\mathcal{C}$ chooses randomly $e_\gamma \in \mathbb{Z}_q^*$ and $w_1, w_2, \ldots, w_{\gamma-1}, w_{\gamma-2}, w_p \in \mathbb{Z}_q^*$ where $\gamma \in \{1, 2, 3, \ldots, q_{H_1}\}$.Then, $e_i = e_\gamma - w_i$ is computed by $\mathcal{C}$ for $i \in \{1, 2, 3 \ldots, \gamma-1, \gamma-2, p\}$. As in the proof technique of [30], $\mathcal{C}$ sets up a generator $Q \in \mathbb{G}_1$ and $Y = \alpha Q \in \mathbb{G}_1$, such that it knows $p-1$ pairs $(w_i, k_i = \frac{1}{w_i+\alpha}Q)$ for $i \in \{1, 2, 3, \ldots, p\}$ except γ. To do so, the polynomial $f(z) = \Pi_{i=1}^{p-1}(z + w_i)$ is expanded by $\mathcal{C}$ to acquire the coefficients of $f(z)$, then $f(z) = \sum_{i=0}^{p-1} l_i z^i$ where $\{l_0, l_1, l_3, \ldots, l_{p-1}\} \in \mathbb{Z}_q^*$. Then, Q and Y are computed by $\mathcal{C}$ by the following equations:

$$Q = \sum_{i=0}^{p-1} l_i(\alpha^i P) = f(\alpha)P \in \mathbb{G}_1$$

and

$$Y = \sum_{i=1}^{p} l_{i-1}(\alpha^i P) = \alpha f(\alpha)P = \alpha Q$$

The pair (w_i, k_i) can be acquired similar to the proof in [30]. The following equations $f_i(z)$ are expanded by the $\mathcal{C}$ to get $f_i(z) = \frac{f(z)}{z+w_i} = \sum_{i=0}^{p-2} d_i z^i$ and k_i is taken as

$$k_i = \sum_{i=0}^{p-2} d_i(\alpha^i P) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha+w_i}P = \frac{1}{\alpha+w_i}Q$$

Q_{pub} and $x = (-\alpha - e_\gamma)$ are chosen as the master public key and master secret key of the SP respectively.

$Q = (-\alpha - e_\gamma)$, $Q_{pub} = (-Y - e_\gamma Q)$ and $g = e(Q, Q)$ are sent to the $\mathcal{A}_I$ by the $\mathcal{C}$. $\forall i \in \{1, 2, 3, \ldots, p\}$ except γ, then we have $(e_i, -k_i) = (e_i, \frac{1}{e_i+x}Q)$. The public key and private key of the server PK_s, SK_s are computed by the $\mathcal{C}$ using the CLC-KG algorithm and are sent to $\mathcal{A}_I$.

The algorithm $\mathcal{C}$ generates the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, g, P_{pub}, H_1, H_2, H_3\}$ and sends them with the public key of the server $PK_s = v(H_1(ID_s)P + P_{pub})$ to $\mathcal{A}_I$. $\mathcal{C}$ selects ID_t randomly. $\mathcal{C}$ simulates the random oracles of H_1, H_2 and H_3 with lists regarding to avoid collision and consistency. For oracles queries and responses, $\mathcal{C}$ prepares four lists L_{H_1}, L_{H_2}, L_{H_3} and L_K to keep the public keys. We assume $H_1(ID)$ query is completed first then the other queries are created.

1) H_1 queries: When $\mathcal{A}_I$ sends this query on ID, $\mathcal{C}$ chooses randomly $e_i \in \mathbb{Z}_q^*$ as well as sends it to $\mathcal{A}_I$. However, if $ID = ID_t$, $\mathcal{C}$ returns e_i to $\mathcal{A}_I$. Then, $\mathcal{C}$ updates L_{H_1} with $(ID, h_{1,i})$.
2) H_2 queries: When $\mathcal{A}_I$ submits this query on PK_s, First of all $\mathcal{C}$ verifies if the value of H_2 was precedently defined for PK_s. If it was defined, $\mathcal{C}$ returns the value that was defined previously. Otherwise, $\mathcal{C}$ randomly chooses $h_{2,i} \in \mathbb{Z}_q^*$. Then, $\mathcal{C}$ returns $h_{2,i}$ to $\mathcal{A}_I$ and updates L_{H_2} with $(PK_s, h_{2,i})$.
3) H_3 queries: When $\mathcal{A}_I$ submits this query on $(ID_c, ID_s, PK_i, r_i, \beta_i)$, $\mathcal{C}$ firstly looks at L_{H_2} to verify that the value of H_3 was precedently assigned to the $(ID_c, ID_s, PK_i, r_i, \beta_i)$. If yes, the precedent value is returned by $\mathcal{C}$ to $\mathcal{A}_I$. Otherwise, $\mathcal{C}$ chooses randomly $h_{3,i} \in \mathbb{Z}_q^*$, $\zeta^* = r_i.e(Q, Q)^{h_{3,i}}$ and returns it to $\mathcal{A}_I$. $\mathcal{C}$ updates L_{H_2} with $(ID_c, ID_s, PK_i, r_i, \beta_i, h_{3,i}, \zeta^*)$.
4) *Extract partial private key* queries: This query can be asked by $\mathcal{A}_I$ on identity ID. If $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation. Else, the value of $H_1(ID) = e_i$ is known by $\mathcal{C}$ that can be used to compute the partial private key $-k_i = \frac{1}{e_i+x}P$ and is returned to $\mathcal{A}_I$.
5) *Extract private key* queries: $\mathcal{A}_I$ can submit this query on ID_i requesting for the private key. $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation. If $ID \neq ID_t$, the partial private key is known by $\mathcal{C}$. Then, $\mathcal{C}$ looks up at the L_k for the inputs (ID_i, PK_i, v_i) (if these inputs do not already exist in L_k, a new user key information will be generated by $\mathcal{C}$) and $SK_i = -(1/v)k_i$ is returned by $\mathcal{C}$ to $\mathcal{A}_I$.
6) *Request public key* queries: An identity ID_i can be selected by $\mathcal{A}_I$ and is sent to $\mathcal{C}$. Then, $\mathcal{C}$ checks whether L_k has these tuples (ID_i, PK_i, v_i) if yes, the PK_i is returned to $\mathcal{A}$. If it has not, a random $v \in \mathbb{Z}_q^*$ is selected and $PK_i = v_i(e_iQ + Q_{pub})$ is set. Finally, $\mathcal{C}$ updates L_k with (ID_i, PK_i, v_i) and returns PK_i to $\mathcal{A}_I$.
7) *Replace public key* queries: The public key PK_i may be replaced with chosen value by $\mathcal{A}_I$. For this query on (ID_i, PK_i) the L_k is updated by $\mathcal{C}$ with $(ID_i, PK_i, \perp)$ where $\perp$ means that the value is unknown.
8) *Send* queries :
   a) When $\mathcal{A}_I$ submits *Send* $(\Pi_C^i$, "start ") query with the client identity ID_c, If $ID_c \neq ID_t$, $\mu \in \mathbb{Z}_q^*$ is chosen randomly by $\mathcal{C}$. Then, $r = g^\mu$ and $T = \mu(w_iQ + Q_{pub}) - h(PK_s + h_{2,i}(w_iQ + Q_{pub}))$ are computed. Finally, $\mathcal{C}$ returns ID_c and T to $\mathcal{A}_I$. If $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation.
   b) When $\mathcal{A}_I$ submits *Send* $(\Pi_S^j, (ID_c, T))$ query to the server with (ID_c, T), if $ID \neq ID_t$, $\mathcal{C}$ computes $r = e(T, SK_s)$. Then, $\beta \in \mathbb{Z}_q^*$ is chosen randomly and $h = H_3(ID_c, ID_s, PK_s, r, \beta)$ is computed by $\mathcal{C}$. Finally, β and h are returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}$ fails and stops the simulation.
   c) When $\mathcal{A}_I$ submits *Send* $(\Pi_C^i, (\beta, h))$ query to the client, If $ID \neq ID_t$, $H_2(ID_c, ID_s, PK_s, r, \beta)$ is computed again by $\mathcal{C}$ to check whether it is equal to the h that is received from the server or not. Then, $S = \mu SK_s$ and $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$ are computed by $\mathcal{C}$. Finally, S is returned to $\mathcal{A}$. If $ID = ID_t$, $\mathcal{C}$ fails due to the $\mathcal{A}_I$ can not get the correct value of h'.
   d) When $\mathcal{A}_I$ submits *Send* (Π_S^j, S) query to the server, If $ID \neq ID_t$, $e(S, e_iQ + Q_{pub})g^{-h}$ is computed by $\mathcal{C}$ to check whether is equal to r. If the condition is satisfied, $\mathcal{C}$ accepts and computes $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$. Otherwise, the simulation is stopped by $\mathcal{C}$.
9) *Corrupt* queries: Whenever this query on ID_c is sent by $\mathcal{A}_I$, $-k_i$ is returned by $\mathcal{C}$.
10) *Reveal* queries: Whenever this query on ID_c is submitted by $\mathcal{A}_I$, If $ID_s \neq ID_t$, $\mathcal{C}$ fails and stops the simulation. Otherwise, $\lambda \in \mathbb{Z}_q^*$ is chosen and $T^* = -\lambda Q$ is computed by $\mathcal{C}$. T^* is returned to $\mathcal{A}_I$ to get the session key. Assume that $\Gamma = \frac{\lambda}{\alpha}$ and we know that $x = (-\alpha - e_i)$, so then we can get
$$\begin{aligned} T^* &= -\lambda v_s Q - \lambda h(2, s)Q \\ &= -\Gamma\alpha v_s Q - \Gamma\alpha h(2, s)Q \\ &= (e_i + x)\Gamma v_s Q + (e_i + x)\Gamma h(2, s)Q \\ &= \Gamma v_s e_i Q + \Gamma v_s Q_{pub} + (e_i + x)\Gamma h(2, s)Q \\ &= \Gamma v_s(e_iQ + Q_{pub}) + e_i\Gamma h_{2,s}Q + \Gamma h_{2,s}Qpub \\ &= \Gamma v_s(e_iQ + Q_{pub}) + \Gamma h_{2,s}(e_iQ + Qpub) \\ &= \Gamma PK_s + \Gamma h_{2,s}(e_iQ + Qpub) \\ &= \Gamma(PK_s + h_{2,s}(e_iQ + Qpub)) \end{aligned}$$
Then, $\mathcal{A}_I$ is unable to get the correct session key unless it makes H_3 query on $e(P, P)^\Gamma$.
11) *Test* queries: Whenever this query is sent by $\mathcal{A}_I$, a fair coin b is generated by $\mathcal{C}$. A random input $(ID_c, ID_s, PK_i, r_i, \beta_i, h_{2,i}, \varsigma_i)$ is selected from L_{H_3} by $\mathcal{C}$ that contains no more than q_{H_3} inputs. If $b = 1$ the session key is returned to $\mathcal{A}_I$ by $\mathcal{C}$. If $b = 0$, $\mathcal{C}$ sends $\perp$ to $\mathcal{A}_I$ where $\perp$ denotes a random string.

On the other hand, it is observed by $\mathcal{C}$ that these two events $\exists i, Esk \wedge Test(\Pi_C^j)$ and $\exists j, Esk \wedge Test(\Pi_S^j) \wedge \neg E^{C2S}$ are equivalent. As result, we know this probability $\Pr[Esk \wedge Test(\Pi_C^i)] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$. In addition, by simulating the queries that are submitted to the client, we can get this probability

$$\Pr\left[sk = H_3(ID_c, ID_s, PK_s, r, \beta)|_{r\leftarrow\mathbb{G}_2}^{\beta\in\mathbb{Z}_q^*}\right] \geq \frac{\varepsilon}{2} - Pr_{c2s}$$

It is known that $\Pr_{C2S}$ is negligible from the proof in Theorem 1. Furthermore, whether ε is non-negligible, then we know that $\frac{\varepsilon}{2} - \Pr_{C2S}$ is a non-negligible. Hence, to obtain the session key, the adversary $\mathcal{A}_I$ needs to select inputs containing the correct element of $r_i = e(P,P)^{\Gamma}$ with probability $\frac{1}{q_{H_3}}$. $\mathcal{A}_I$ allows to make the previous queries with the following limitations:

1) $\mathcal{A}_I$ can not submit *Extract partial private key* query on ID_t when the public key of the ID_t has been replaced.
2) $\mathcal{A}_I$ can not submit *Extract private key* query on ID_t.

Indeed, as the proof in [33] the value of $(e(Q,Q))^{\lambda^{-1}}$ is computed by $\mathcal{C}$ as a solution for q-BDHI problem when $\zeta^* = e(P,P)^{\lambda^{-1}}$ with the following equation

$$e(Q,Q)^{1/\alpha} = \zeta^{*(l_0^2)} e(\sum_{j=0}^{p-2} l_{j+1}(\alpha^j P), l_0 P)$$
$$e(Q, \sum_{j=0}^{p-2} l_{j+1}(\alpha^j P))$$

Hence, our scheme achieves a key agreement protocol under q-BDHI problem against adversary type I.

Lemma 2: Our scheme is resists to $\mathcal{A}_{II}$ under mBDHI problem in the random oracle model.

Proof: When a *Test* query is submitted, the adversary can guess the value of coin b correctly with probability not less than $1/2$. Lets suppose adversary is able to guess the coin with ε. As result, the adversary with advantage $\Pr[Esk] \geq \varepsilon/2$ enables to acquire the correct session key. Here, some symbols are employed in our proof. We use the symbol *Esk* to show the event of acquiring the correct session key by adversary. In addition, these symbols $Test(C^i)$ and $Test(S^j)$ are used to explain the correct test queries that are made to the client and to the server separately. The symbol E^{C2S} represents the event of breaking the CSA security. By supposing that the *Test* query can be submitted to the client as well as to the server by adversary, then this probability, into some i and j, is :

$$\Pr[Esk \wedge Test(\Pi_S^j) \wedge E^{C2S}] + \Pr[Esk \wedge Test(\Pi_S^j) \wedge \neg E^{C2S}] + \Pr[Esk \wedge Test(\Pi_C^i)] \geq \frac{\varepsilon}{2}$$

Assuming the possibility of breaking the CSA security is denoted by $\Pr_{C2S}$. Then, the following probability is computed into some i and j

$$\Pr[Esk \wedge Test(\Pi_C^i) \wedge \neg E^{C2S}] + \Pr[Esk \wedge Test(\Pi_C^i)] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$$

As the proof in [30], the challenger $\mathcal{C}$ takes as input $(P, \alpha P, \alpha^2 P, \ldots \alpha^p P)$ and attempts to extract $(w_i, \frac{1}{w_i+\alpha}P)$ from its communication with $\mathcal{A}_{II}$, where $w_i, \alpha \in \mathbb{Z}_q^*$

Initialization In preparation phase, $\mathcal{C}$ chooses randomly $x \in \mathbb{Z}_q^*$ and $P_{pub} = xP$ is computed. The algorithm $\mathcal{C}$ generates the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, g, P_{pub}, H_1, H_2, H_3\}$ and forwards to $\mathcal{A}_{II}$. $\mathcal{C}$ selects ID_t randomly. $\mathcal{C}$ simulates the random oracles of H_1, H_2 and H_3 with lists regarding to avoid collision and consistency. For oracles queries and responses, $\mathcal{C}$ prepares four lists $L_{H_1}, L_{H_2}, L_{H_3}$ and L_K to keep the public keys. We assume $H_1(ID)$ query is completed first then the other queries are created.

1) H_1 queries: When $\mathcal{A}_{II}$ submits this query on ID, $\mathcal{C}$ checks whether L_{H_1} has these inputs $(ID_i, h_{1,i})$. If it has them, e_i is returned to $\mathcal{A}_{II}$ and L_{H_1} is updated with (ID, e_i). However, if $ID = ID_t$, $\mathcal{C}$ returns $H_1(ID) = e_i$ to $\mathcal{A}$ when $\mathcal{A}_{II}$ submits this query on ID_t .
2) H_2 queries: When $\mathcal{A}_{II}$ submits this query on PK_s, First of all $\mathcal{C}$ verifies if the value of H_2 was precedently defined for PK_s. If it was defined, $\mathcal{C}$ returns the value that was defined previously. Otherwise, $\mathcal{C}$ verifies if $PK_i = e_i\alpha P + x\alpha P$. If it is satisfied, $\mathcal{C}$ returns $h_{2,i} = \gamma$ and updates L_{H_2} with (PK_s, γ). If it is not satisfied, $\mathcal{C}$ randomly chooses $h_{2,i} \in \mathbb{Z}_q^*$. Then, $\mathcal{C}$ returns $h_{2,i}$ to $\mathcal{A}_{II}$ and updates L_{H_2} with $(PK_s, h_{2,i})$.
3) H_3 queries: When $\mathcal{A}_{II}$ submits this H_3 query on $(ID_c, ID_s, PK_i, r_i, \beta_i)$, $\mathcal{C}$ firstly looks at L_{H_3} to verify that the value of H_3 was precedently assigned to the $(ID_c, ID_s, PK_i, r_i, \beta_i)$. If yes, the precedent value is returned by $\mathcal{C}$ to $\mathcal{A}_{II}$. Otherwise, $\mathcal{C}$ chooses randomly $h_{3,i} \in \mathbb{Z}_q^*$, $\zeta^* = r_i.e(P,P)^{h_{3,i}}$ is computed and is returned to $\mathcal{A}_{II}$. $\mathcal{C}$ updates L_{H_3} with $(ID_c, ID_s, PK_i, r_i, \beta_i, h_{3,i}, \zeta^*)$.
4) *Extract private key* queries: $\mathcal{A}_{II}$ can submit this query on ID_i requesting for the private key. $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation. If $ID \neq ID_t$, $\mathcal{C}$ looks up at the L_k for the inputs (ID_i, PK_i, v_i) (if these inputs do not already exist in L_k a new user key information will be generated by $\mathcal{C}$) and $SK_i = \frac{1}{v+h_{2,i}}.\frac{1}{e_i+x}P$ is computed and returned by $\mathcal{C}$ to $\mathcal{A}_{II}$.
5) *Request public key* queries: An identity ID can be selected by $\mathcal{A}_{II}$ and is sent to $\mathcal{C}$. If $ID \neq ID_t$, $v \in \mathbb{Z}_q^*$ is selected randomly by $\mathcal{C}$ and $PK_i = v(e_iP + P_{pub})$ is set as public key. Then, $\mathcal{C}$ updates L_k with (ID_i, PK_i, v_i) and PK_i is returned to $\mathcal{A}_{II}$. If $ID = ID_t$, the public key $PK_t = e_t\alpha P + x\alpha P$ is set and is returned to $\mathcal{A}_{II}$. Finally, $\mathcal{C}$ updates L_k with $(ID_i, PK_t, \perp)$.
6) *Send* queries:
    a) When $\mathcal{A}_{II}$ submits *Send* $(\Pi_C^i$, "start ") query with the client identity ID_c, If $ID_c \neq ID_t$, $\mu \in \mathbb{Z}_q^*$ is chosen randomly by $\mathcal{C}$. Then, $r = g^{\mu}$ and $T = \mu(e_iP + P_{pub}) - h(PK_s + h_{2,i}(e_iP + P_{pub}))$ are computed. Finally, $\mathcal{C}$ returns ID_c and T to $\mathcal{A}_{II}$. If $ID = ID_t$, $\mathcal{C}$ fails and stops the simulation.
    b) When $\mathcal{A}_{II}$ submits *Send* $(\Pi_S^j, (ID_c, T))$ query to the server with (ID_c, T), if $ID \neq ID_t$, $\mathcal{C}$ computes $r = e(T, SK_s)$. Then, $\beta \in \mathbb{Z}_q^*$ is chosen randomly and $h = H_3(ID_c, ID_s, PK_s, r, \beta)$ is computed by

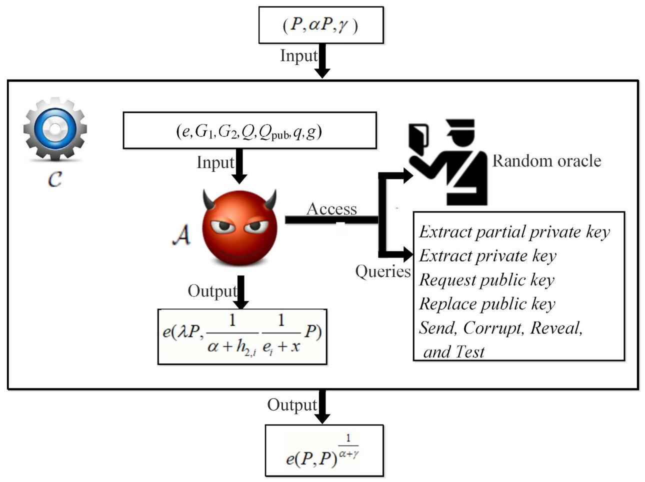
$(P, \alpha P, \gamma)$
Input
$(e, G_1, G_2, Q, Q_{pub}, q, g)$
Input
$\mathcal{C}$
$\mathcal{A}$
Access
Random oracle
Queries
Extract partial private key
Extract private key
Request public key
Replace public key
Send, Corrupt, Reveal, and Test
Output
$e(\lambda P, \frac{1}{\alpha + h_{2,i}} \frac{1}{e_i + x} P)$
Output
$e(P,P)^{\frac{1}{\alpha+\gamma}}$

FIGURE 5. The proof structure of Theorem 3.

$\mathcal{C}$. Finally, β and h are returned to $\mathcal{A}_{II}$. Otherwise, $\mathcal{C}$ fails and stops the simulation.

c) When $\mathcal{A}_{II}$ submits *Send* $(\Pi^i_C, (\beta, h))$ query to the client, If $ID \neq ID_t$, $H_3(ID_c, ID_s, PK_s, r, \beta)$ is computed again by $\mathcal{C}$ to check whether it is equal to the h that is received from the server or not. Then, $S = \mu SK_s$ and $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$ are computed by $\mathcal{C}$. Finally, S is returned to $\mathcal{A}_{II}$. If $ID = ID_t$, $\mathcal{C}$ fails due to the $\mathcal{A}_{II}$ can not get the correct value of h'.

d) When $\mathcal{A}_{II}$ submits *Send* (Π^j_S, S) query to the server, If $ID \neq ID_t$, $e(S, e_iP + P_{pub})g^{-h}$ is computed by $\mathcal{C}$ to check whether it is equal to r. If the condition is satisfied, $\mathcal{C}$ accepts and computes $sk = H_3(ID_c, ID_s, PK_s, r, \beta)$. Otherwise, the simulation is stopped by $\mathcal{C}$.

7) *Corrupt* queries: Whenever *Corrupt* query on ID_c is sent by $\mathcal{A}_{II}$, D_{ID_c} is returned by $\mathcal{C}$.

8) *Reveal* queries: Whenever *Reveal* query on ID_c is sent by $\mathcal{A}_{II}$, If $ID \neq ID_t$, $\mathcal{C}$ is fails and stops the simulation. Otherwise, $\alpha \in \mathbb{Z}^*_q$ is chosen and $T^* = \alpha P$ is processed by $\mathcal{C}$. The T^* is returned to $\mathcal{A}_{II}$ to get the session key. Then, $\mathcal{A}_{II}$ is unable to get the correct r_i unless it makes H_3 query on $e(T^*, SK_s)$ to get the session key.

9) *Test* queries: Whenever this query is sent by $\mathcal{A}_{II}$, a fair coin b is generated by $\mathcal{C}$. A random input $(ID_c, ID_s, PK_i, r_i, \beta_i, h_{2,i}, \varsigma_i)$ is selected from L_{H_3} by $\mathcal{C}$ that contains no more than q_{H_3} inputs. If $b = 1$ the session key is returned to $\mathcal{A}_{II}$ by $\mathcal{C}$. If $b = 0$, $\mathcal{C}$ sends $\perp$ to $\mathcal{A}_{II}$ where $\perp$ denotes a random string.

On the other hand, it is observed by $\mathcal{C}$ that these two events $\exists i, Esk \wedge Test(\Pi^i_C)$ and $\exists j, Esk \wedge Test(\Pi^j_S) \wedge \neg E^{C2S}$ are equivalent. As a result, we know this probability $\Pr[Esk \wedge Test(\Pi^i_C)] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$. In addition, by simulating the queries that are submitted to the client we can get this

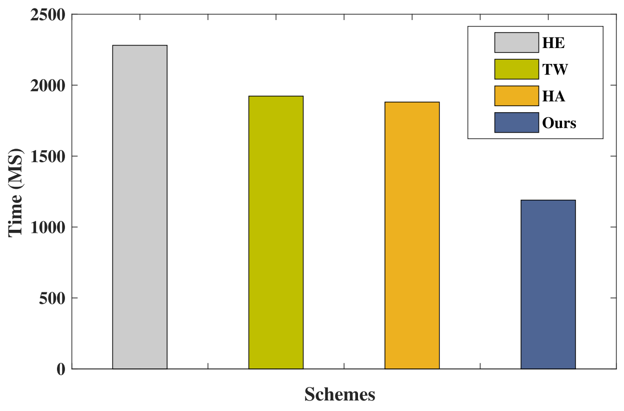

FIGURE 6. Client time.

probability

$$\Pr\left[sk = H_3(ID_c, ID_s, PK_s, r, \beta)|^{\beta \in \mathbb{Z}^*_q}_{r \leftarrow \mathbb{G}_2}\right] \geq \frac{\varepsilon}{2} - Pr_{c2s}$$

It is known that $\Pr_{C2S}$ is negligible from the proof in Theorem 1. Furthermore, whether ε is non-negligible, then we know that $\frac{\varepsilon}{2} - \Pr_{C2S}$ is a non-negligible. Hence, to have the key agreement the adversary should select inputs containing the correct element of $r_i = e(T^*, SK_s)$ with probability $\frac{1}{q_{H_3}}$. $\mathcal{A}_{II}$ allows to make the previous queries, which can not submit *Extract private key* query on ID_t. Indeed, the mBDHI problem can be delivered by the following equation

$$r_i = e(T^*, SK_s) = e(\lambda P, \frac{1}{\alpha + \gamma} \frac{1}{e_i + x} P)$$

then we have

$$e(P, P)^{\frac{1}{\alpha+\gamma}} = r_i^{\frac{e_i+x}{\lambda}}$$

Indeed, our scheme achieves a key agreement protocol under mBDHI problem against adversary type II.

C. ANALYSIS OF SERVER-TO-CLIENT AUTHENTICATION

The server can not be compromised by the adversary to interact with the client by Theorem 3 under q-BDHI problem.

Theorem 3: Supposing that $\mathcal{A}$ takes a non-negligible advantage ε for breaking the SCA security. Also, at most q_S queries to the server's oracle Π^j_S, q_C queries to the client's oracle Π^i_C, and q_{H_i} queries on H_i oracle $\forall i \in \{1, 2, 3\}$ are made by $\mathcal{A}$. Therefore a challenger $\mathcal{C}$ solves the q-BDHI problem by having a non-negligible probability.

Proof: Figure 5 depicts the proof structure of this theorem. As it is mentioned in Theorem 2, the simulation is working correctly since the E^{C2S} happens. We denote the event of breaking the server-to-client authentication with this symbol E^{S2C}. The event E^{S2C} happens during the simulation when (ID_c, T) are sent by the client to the server. In addition, (β, h) are received by the client while are not issued from the correct server. The previous situation occurs in one of the following conditions :

TABLE 2. Security comparisons.

	User authentication	Mutual authentication	Key agreement	Resistance to forgery attack	Perfect forward-secrecy	Provable secrecy	Environment
HE [20]	Y	Y	Y	Y	Y	Y	IBC→IBC
TW [25]	Y	Y	Y	Y	Y	Y	IBC→ IBC
HA [28]	Y	Y	Y	Y	Y	Y	CLC→ CLC
Our scheme	Y	Y	Y	Y	Y	Y	IBC → CLC

TABLE 3. Computational costs.

	HE [20]	TC [25]	HA [28]	Ours
Client-Time	$3T_m + T_{inv}$	$4T_m + 3T_{ad}$	$5T_m + T_{ad}$	$T_m + T_{ad}$
Server-Time	$T_p + 2T_m + 2T_{ad}$	$4T_m + 3T_{ad}$	$2T_p + 4T_m + 2T_{ad}$	$2T_p + T_m + T_{ad}$

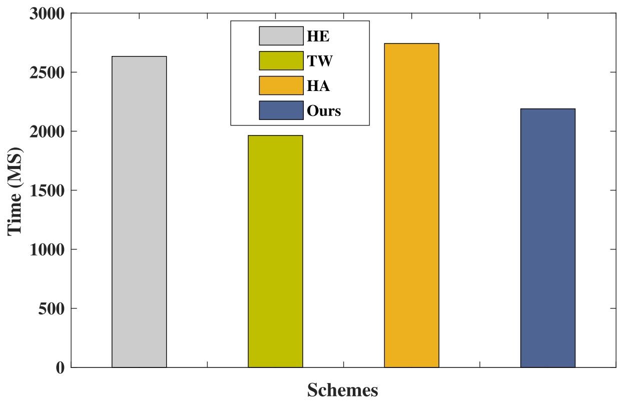

FIGURE 7. Server time.

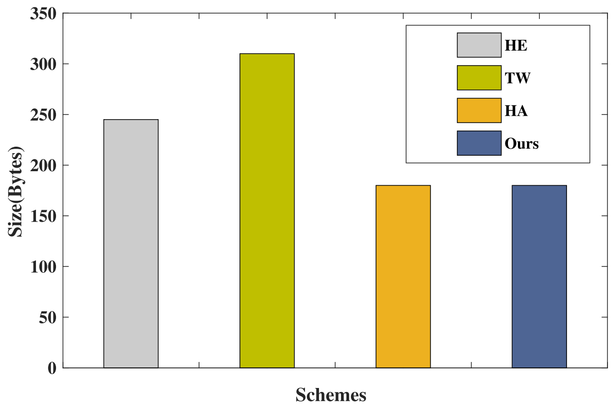

FIGURE 8. Communication costs.

1) The value of h is guessed by adversary $\mathcal{A}$ with probability less than $q_C/2^k$.
2) The value T is happened in a further session with a probability $q_C/q \times (q_C - 1)$ less than q_C^2/q.
3) $H_1(ID_t)$ is asked by adversary $\mathcal{A}$ with a probability $\Pr[(ID_c, ID_s, PK_s, r, \beta)|$ $\beta \in_R \mathbb{Z}_q^*, r = e(T, SK_s)]$.
   Then, we have
   $$\Pr[E^{S2C}|\neg E^{C2S}] \leq \Pr[ID_c, ID_s, PK_s, r, \beta]|\beta \in_R \mathbb{Z}_q^*, r = e(T, SK_s)] + \frac{q_C}{2^k} + \frac{q_C^2}{q}$$

The algorithm $\mathcal{C}$ generates the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, g, P_{pub}, H_1, H_2, H_3\}$ and sends them with the public key of the server $PK_s = v(H_1(ID_s)P + P_{pub})$ to $\mathcal{A}$. $\mathcal{C}$ selects ID_t randomly. $\mathcal{C}$ simulates the random oracles of H_1, H_2 and H_3 with lists regarding to avoid collision and consistency. For oracles queries and responses, $\mathcal{C}$ prepares four lists L_{H_1}, L_{H_2}, L_{H_3} and L_K to keep the public keys.

Hence, to break our server-to-client authentication, the adversary needs to select inputs containing the correct element of $r_i = e(T^*, SK_s)$ with probability $\frac{1}{q_{H_2}}$. Indeed, The q-BDHI problem can be delivered by the following equation

$$r_i = e(T^*, SK_s) = e(\lambda P, \frac{1}{\alpha + h_{2,i}} \frac{1}{e_i + x} P)$$

subsequently we get

$$e(P, P)^{\frac{1}{\alpha + h_{2,i}}} = r_i^{\frac{e_i + x}{\lambda}}$$

Indeed, our scheme achieves mutual authentication under q-BDHI problem .

VI. PERFORMANCE

This section illustrates the advantages of the proposed protocol compared to existing protocols. To this regard, the evaluation with consideration to security and the performance is conducted. The performance evaluation shows the the computational cost and the communication overhead in the client and server for the compared protocols. This comparison is conduct with the following protocols He (HE) [20], Tseng *et al.* (TW) [25], and Hassan *et al.* (HA) [28]. For the performance evaluation, we use these notations T_p, T_m, T_{ad}, and T_{inv} to explian the bilinear pairing operation time, multiplication time in $\mathbb{G}_1$, inversion operation time, and addition in $\mathbb{G}_1$ time respectively.

The theoretical analysis is given regarding to the computational cost in Table 3. From Table 2, We use Y in Table 3 to indicate that the protocol satisfies specific security requirement. We find that the proposed scheme can be employed when the client belong to IBC and the server belong to CLC. Hence, the proposed scheme owns the advantage of working with applications using various cryptosystem environments compared with the existing protocols.

To evaluate the real computational costs, four schemes [20], [25] [28] including our proposed scheme are implemented using Java pairing-based cryptography (JPBC)

Library [34]. Our experimental is done on a machine with CPU Intel Core i 7-3537U dual core (2.00 and 2.50) GHz and RAM 4 GB for the server, while Huawei Mate 8 with CPU Hisilicon Kirin 950 and RAM 4.0 GB is used for the client.

The curve $y^2 = x^3 + x$ over the field $\mathbb{F}_p$ is used to construct the Type A pairings for $p = 3 \bmod 4$. The experimental, entailed 80-bit AES key size security level while the size of $p = 1024$ bits and size of $q = 160$ bits [35]. The processing costs of the mobile side and the server side are shown in Figure 6 and Figure 7 respectively. Our scheme has better processing cost in the client side while it is better than HE [20] and HA [28] schemes in the server side. Indeed, the proposed scheme is suitable for the mobile devices that have limited storage and power.

The communication costs of the schemes are displayed in Figure 8. We use the following notations $|id| = \frac{80}{8} = 10$ bytes, the elliptic carve with $q = \frac{160}{8} = 20$ bytes, and the $\mathbb{G}_1$ size is 65 bytes according to the work introduced in [36], to calculate the exact costs for each scheme.

Here, we give the communication cost for the compored schemes HE [20], TW [25], HA [28], and the proposed scheme using the following calculations $|id| + 2|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| = 10 + 2 \times 20 + 3 \times 65 = 245$ bytes, $|id| + 2|\mathbb{Z}_q^*| + 4|\mathbb{G}_1| = 10 + 2 \times 20 + 4 \times 65 = 310$ bytes, $|id| + 2|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| = 10 + 2 \times 20 + 2 \times 65 = 180$ bytes, and $|id| + 2|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| = 10 + 2 \times 20 + 2 \times 65 = 180$ bytes respectively.

Figure 8 exhibits that our scheme has better communication overhead than HE [20] and TW [25], while it has the same communication cost with HA [28].

VII. CONCLUSION

In this paper, we present a scheme for user authentication with a key agreement in heterogeneous client-server systems. The security of our scheme is proved in the random oracle model. We also use the q-strong Diffie-Hellman problem (q-SDH), the q-bilinear Diffie-Hellman inversion problem (q-BDHI), and the modified bilinear Diffie-Hellman inversion problem (mBDHI) as hard assumptions in our security proof. Our protocol can be used in heterogeneous environments when the client relies on IBC and the server relies on CLC. However, exciting protocols cannot be used in this environment, which emphasizes the importance of our scheme. Moreover, our protocol can be used in both client-server architectures and multi-server architectures. In the future, we plan to improve the efficiency of our protocol by using post-quantum lattice-based cryptography. We will also address privacy preservation in the development of user authentication protocols for mobile requirements.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (NU/IFC/ENT/01/013) under the Institutional Funding Committee at Najran University, Saudi Arabia.

REFERENCES

[1] K. Riad, R. Hamza, and H. Yani, "Sensitive and energetic IoT access control for managing cloud electronic health records," *IEEE Access*, vol. 7, pp. 86384–86393, 2019.

[2] Z. Liu, B. Li, Y. Huang, J. Li, Y. Xiang, and W. Pedrycz, "NewMCOS: Towards a practical multi-cloud oblivious storage scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 714–727, Apr. 2020.

[3] K. Nimmy, S. Sankaran, K. Achuthan, and P. Calyam, "Lightweight and privacy-preserving remote user authentication for smart Homes," *IEEE Access*, vol. 10, pp. 176–190, 2022.

[4] L. D. Tsobdjou, S. Pierre, and A. Quintero, "A new mutual authentication and key agreement protocol for mobile client—Server environment," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1275–1286, Jun. 2021.

[5] Y. Najaflou, B. Jedari, F. Xia, L. T. Yang, and M. S. Obaidat, "Safety challenges and solutions in mobile social networks," *IEEE Syst. J.*, vol. 9, no. 3, pp. 834–854, Sep. 2013.

[6] H. Shen, C. Gao, D. He, and L. Wu, "New biometrics-based authentication scheme for multi-server environment in critical systems," *J. Ambient Intell. Humanized Comput.*, vol. 6, no. 6, pp. 825–834, Dec. 2015.

[7] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.

[8] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou, "Searchable symmetric encryption with forward search privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 460–474, Jan. 2021.

[9] Y. Lu, L. Li, H. Peng, and Y. Yang, "Robust anonymous two-factor authenticated key exchange scheme for mobile client-server environment," *Secur. Commun. Netw.*, vol. 9, no. 11, pp. 1331–1339, Jul. 2016.

[10] J. A. Buchmann, E. Karatsiolis, and A. Wiesmaier, *Introduction to Public Key Infrastructures*. Berlin, Germany: Springer, 2013.

[11] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[12] M. Girault, "Self-certified public keys," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1991, pp. 490–497.

[13] W.-B. Hsieh and J.-S. Leu, "An anonymous mobile user authentication protocol using self-certified public keys based on multi-server architectures," *J. Supercomput.*, vol. 70, no. 1, pp. 133–148, Oct. 2014.

[14] T.-Y. Wu and Y.-M. Tseng, "An efficient user authentication and key exchange protocol for mobile client–server environment," *Comput. Netw.*, vol. 54, no. 9, pp. 1520–1530, 2010.

[15] M. Senftleben, A. Barroso, M. Bucicoiu, M. Hollick, S. Katzenbeisser, and E. Tews, "On the privacy and performance of mobile anonymous microblogging," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 7, pp. 1578–1591, Jul. 2016.

[16] A. Hassan, N. Eltayieb, R. Elhabob, and F. Li, "A provably secure certificateless user authentication protocol for mobile client-server environment," in *Proc. Int. Conf. Emerg. Internetworking, Data Web Technol.* Cham, Switzerland: Springer, 2017, pp. 592–602.

[17] F. Li, J. Wang, Y. Zhou, C. Jin, and S. H. Islam, "A heterogeneous user authentication and key establishment for mobile client–server environment," *Wireless Netw.*, vol. 26, pp. 913–924, Sep. 2020.

[18] G. Fang and G. Huang, "Improvement of recently proposed remote user authentication schemes," Cryptol. ePrint Arch., Paper 2006/200, 2006. [Online]. Available: https://eprint.iacr.org/2006/200

[19] Y.-M. Tseng, T.-Y. Wu, and J.-D. Wu, "A pairing-based user authentication scheme for wireless clients with smart cards," *Informatica*, vol. 19, no. 2, pp. 285–302, 2008.

[20] D. He, "An efficient remote user authentication and key agreement protocol for mobile client–server environment from pairings," *Ad Hoc Netw.*, vol. 10, no. 6, pp. 1009–1016, 2012.

[21] C.-H. Chou, K.-Y. Tsai, and C.-F. Lu, "Two ID-based authenticated schemes with key agreement for mobile environments," *J. Supercomput.*, vol. 66, no. 2, pp. 973–988, Nov. 2013.

[22] X. Huang, X. Chen, J. Li, Y. Xiang, and L. Xu, "Further observations on smart-card-based password-authenticated key agreement in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1767–1775, Jul. 2014.

[23] M. S. Farash and M. A. Attari, ‘‘A secure and efficient identity-based authenticated key exchange protocol for mobile client–server networks,’’ *J. Supercomput.*, vol. 69, no. 1, pp. 395–411, 2014.

[24] J.-L. Tsai and N.-W. Lo, ‘‘Provably secure and efficient anonymous ID-based authentication protocol for mobile devices using bilinear pairings,’’ *Wireless Pers. Commun.*, vol. 83, no. 2, pp. 1273–1286, Jul. 2015.

[25] Y.-M. Tseng, S.-S. Huang, and M.-L. You, ‘‘Strongly secure ID-based authenticated key agreement protocol for mobile multi-server environments,’’ *Int. J. Commun. Syst.*, vol. 30, no. 11, p. e3251, Jul. 2017.

[26] S. S. Al-Riyami and K. G. Paterson, ‘‘Certificateless public key cryptography,’’ in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer-Verlag, 2003, pp. 452–473.

[27] M.-B. Hou and Q.-L. Xu, ‘‘Secure certificateless-based authenticated key agreement protocol in the client-server setting,’’ in *Proc. IEEE Int. Symp. IT Med. Educ.*, vol. 1, Aug. 2009, pp. 960–965.

[28] A. Hassan, N. Eltayieb, R. Elhabob, and F. Li, ‘‘An efficient certificateless user authentication and key exchange protocol for client-server environment,’’ *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 6, pp. 1713–1727, 2018.

[29] D. Boneh and M. Franklin, ‘‘Identity-based encryption from the Weil pairing,’’ in *Proc. Annu. Int. Cryptol. Conf.* Springer-Verlag, 2001, pp. 213–229.

[30] D. Boneh and X. Boyen, ‘‘Short signatures without random oracles,’’ in *Proc. Eurocrypt*, vol. 3027. Berlin, Germany: Springer-Verlag, 2004, pp. 56–73.

[31] M. Bellare and P. Rogaway, ‘‘Random oracles are practical: A paradigm for designing efficient protocols,’’ in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, 1993, pp. 62–73.

[32] D. Pointcheval and J. Stern, ‘‘Security arguments for digital signatures and blind signatures,’’ *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.

[33] P. S. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, ‘‘Efficient and provably-secure identity-based signatures and signcryption from bilinear maps,’’ in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2005, pp. 515–532.

[34] A. De Caro and V. Iovino, ‘‘JPBC: Java pairing based cryptography,’’ in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Corfu, Greece, Jun. 2011, pp. 850–855.

[35] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-the Advanced Encryption Standard*. Berlin, Germany: Springer-Verlag, 2002.

[36] K.-A. Shim, Y.-R. Lee, and C.-M. Park, ‘‘EIBAS: An efficient identity-based broadcast authentication scheme in wireless sensor networks,’’ *Ad Hoc Netw.*, vol. 11, no. 1, pp. 182–189, Jan. 2013.

ALZUBAIR HASSAN received the B.Sc. degree in computer science from the University of Kassala, in 2010, the M.Sc. degree in mathematical science from the University of Khartoum, in 2013, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China, in 2018. He is currently an Assistant Professor at the School of Computer Science, University College Dublin. In addition, he was a Postdoctoral Researcher at the School of Computer Science and Cyber Engineering, Guangzhou University. Furthermore, he was a Research Scientist at the School of Computer Science, University College Dublin. He was also a Researcher with the Lero-the Irish Software Research Centre. His research interests include cryptography, network security, privacy-preserving in machine learning, and adaptive security.

RAFIK HAMZA received the M.Sc. and Ph.D. degrees in cryptography and security from Batna 2 University, Algeria, in 2014 and 2017, respectively. In 2018, he joined DC Research and Development Sonatrach, where he worked as a Principal Engineer focused on data reliability and AI-based prediction for industrial systems. In March 2019, he moved to Guangzhou University, where he worked as a Postdoctoral Researcher. His research activities focused on developing secure and efficient machine learning solutions for industry 4.0 applications while maintaining data privacy. In February 2020, he started working as a Researcher at the Big Data Integration Research Center, NICT, Tokyo. The NICT team focuses on developing data collection and analysis technologies that aim to leverage real-world information for better social life and enable cross-domain combinations (https://www.xdata.nict.jp). In April 2022, he became an Associate Professor at Tokyo International University. His research interests include privacy-preserving machine learning for real-world industrial technologies and applied cryptography for big data applications.

FAGEN LI (Member, IEEE) received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2007. From 2008 to 2009, he was a Postdoctoral Fellow with Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science (JSPS). He worked as a Research Fellow with the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan, from 2010 to 2012. He is currently a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China. He has published more than 100 papers in international journals and conferences. His research interests include cryptography and network security.

AWAD ALI received the Ph.D. degree in computer science from the University of Technology Malaysia (UTM), in 2016. He is currently an Assistant Professor with the Computer Science Department, Najran University. His research interests include component-based systems, software quality, software modeling and measurement, and machine learning techniques.

MOHAMMED BAKRI BASHIR received the B.Sc. degree from SUST University, Sudan, and the Ph.D. degree from the University of Technology Malaysia (UTM), Malaysia, in 2014. He is currently an Associate Professor at the Faculty of Computer Science and Information Technology, Shendi University, Sudan. In addition, he is also an Associate Professor at the Turubah University College, Taif University, Saudi Arabia. His research interests include computer network and data processing, grid computing, distributed computing, NoSQL databases, and AI applications.

SAMAR M. ALQHTANI received the Ph.D. degree in information technology from the University of Newcastle, Australia. She is currently an Assistant Professor at Najran University, Saudi Arabia. She has lectured and developed curricula for computer science and information system courses. She has recently led and worked on various projects, including event detection applications in social media, medical applications, and applying artificial intelligence and machine learning algorithms to emerging technologies. Her research interests include information technology and multimedia, including artificial intelligence, machine learning, deep learning, health informatics, data mining, image processing, computer vision, text processing, and the IoT.

TAWFEEG MOHMMED TAWFEEG received the bachelor's degree in information for communication and technology and the master's degree in information security from the University of Science and Technology, Sudan, in 2016 and 2018, respectively. He is currently working as a Lecturer at the University of Science and Technology. His research interests include cloud computing security, network security, and data science.

ADIL YOUSIF received the B.Sc. and M.Sc. degrees from the University of Khartoum, Sudan, and the Ph.D. degree from the University of Technology in Malaysia (UTM). He is currently an Associate Professor at the College of Arts and Sciences Sharourah, Najran University, Saudi Arabia. He is also a principal investigator of several research projects in artificial intelligence and emerging technologies. His research interests include computer networks, cloud computing, artificial intelligence, and optimization techniques.