Journal of Ambient Intelligence and Humanized Computing (2018) 9:1713–1727
https://doi.org/10.1007/s12652-017-0622-1

ORIGINAL RESEARCH

An efficient certificateless user authentication and key exchange protocol for client-server environment

Alzubair Hassan[1] · **Nabeil Eltayieb**[1] · **Rashad Elhabob**[2] · **Fagen Li**[1,3]

Received: 19 September 2017 / Accepted: 7 November 2017 / Published online: 1 December 2017

Abstract
Identity-based user authentication protocols have been presented to be applicable to resource-constrained devices such as mobile phones. Unfortunately, the previous protocols have the drawback of the key escrow problem. A new protocol of a user authenticated key exchange for the mobile client-server environment is presented based on certificateless public key cryptography (CL-PKC). Our protocol solves the key escrow problem in user authentication schemes based on identity-based public key cryptography (ID-PKC). In addition, the proposed protocol is resisted to both adversaries' types I and II and achieves perfect forward secrecy. The security of the proposed protocol has been proved using computational Diffie-Hellman (CDH) assumption in the random oracle model. Experimental results show that our scheme is better than He et al. and Tsai et al. schemes respectively in communication cost.

Keywords Certificateless authentication · Bilinear pairing · Mobile devices environment · Random Oracle Model

1 Introduction

Researchers are developing innovative technologies and intelligence systems to enhance our daily life's quality by adopting information and communication technology (ICT). For example, in ambient assisted living (AAL) system, the use of ICT can provide health-care monitoring, telehealth services, and it is more convenient to help older adults to take care of themselves. Since the idea of AAL is primarily to support people with peculiar needs, as such, it would involve technical systems which are capable of taking over certain daily tasks whenever required, providing the needy with a high level of autonomy. Several technologies have been used in AAL system to make it more suitable for practical applications such as wireless body area networks (W-BANs), assistive robotics, mobiles, and wearable sensors. Practically, the range of information is received continuously from the embedded sensors and recorded by distributed devices (Shen et al. 2018; Wang et al. 2015). These provide a cooperation between the natural user interface and sensor interfaces around the person, resulting in an intelligent device environment; able to recognize the person, identify their needs, learn from their behaviors, as well as to act and react in their interest.

Mobile devices are used to send/receive and process sensitive data that are employed in medical health monitoring and emergency response systems (Wang et al. 2016, 2017; Hamida et al. 2017). Consequently, adversaries could corrupt the communication data, which in turn raises several security issues. It is paramount for mobile devices user to ensure communication with the right service provider. In addition, the service provider as well needs to be authenticated on the user. Moreover, both users and providers should agree on the session key to be used for the future

✉ Fagen Li
fagenli@uestc.edu.cn

Alzubair Hassan
alzubairuofk@gmail.com

Nabeil Eltayieb
nabeil9@yahoo.com

Rashad Elhabob
rashaduestc@gmail.com

[1] Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

[2] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

[3] Laboratory for Internet of Things and Mobile Internet Technology of Jiangsu Province, Huaiyin Institute of Technology, Huaian 223003, China

communication. So, considering the mobile devices, it is challenging to offer a user authentication, a key exchange and a mutual authentication protocols in this environment (Sabzevar and Sousa 2011; Zhang et al. 2014; Ren et al. 2016; Jaballah et al. 2015).

To find out eventual solutions for these security challenges, some schemes (Nam et al. 2005; Tseng 2006, 2007; Wong and Chan 2001; Wang et al. 2011) have been introduced for the applications of the mobile devices using the traditional public key cryptography. However, the computational cost in these protocols is higher on the user side. An ID-PKC resolving the certificate management in comparison with the traditional certificate-based public key cryptography was proposed by Shamir (1984). However, this system has a major weakness as all the private keys of the users are produced by the private key generator (PKG) which leads directly to the key escrow problem. The system of Shamir relies on the assumption of the integer factorization. Therefore, it is not readily realizable in real practical scenarios. Subsequently, several user authentication protocols for mobile environment have been discussed enormously.

1.1 Related work

In the last decade, several identity-based protocols (Fang and Huang 2006; Das et al. 2006; Tseng et al. 2008; Sun et al. 2014) based on the bilinear pairing have been discussed without mutual authentication and key agreement. Wu and Tseng (2010) worked on a user authenticated key exchange protocol which is secure against the impersonation attack, known session key attack, the identity attack, and the partial forward secrecy. Another user authentication and key agreement protocol for the environment of client-server was introduced by Yoon and Yoo (2010) to improve the performance. He (2012) established a user authentication and key exchange protocol. He mentioned that his proposed protocol protects from various known attacks while the biometrics-based authentication protocol heralded by Shen et al. (2015) is known have a little computational cost. Tsai and Lo (2015) introduced an identity-based authentication scheme. They declared that their scheme is provably secure and the communication overhead is reduced at the side of the mobile user. Wu et al. (2016) presented an efficient user authentication and secure key agreement protocol to overcome the drawbacks found in Tsai and Lo's scheme by maintaining users anonymity. A new authentication scheme for wireless sensor network as well as the formal proof and verification was introduced by Wu et al. (2017) for mutual authentication. In finding the solution for the key escrow problem of the CL-PKC, Al-Riyami and Paterson (2003) presented a new pattern labeled certificateless public key cryptography (CL-PKC). Based on CL-PKC, some certificateless user authentication schemes (Hou and Xu 2009; He 2012) have been suggested without providing mutual authentication. Hassan et al. (2016) proposed a certificateless user authentication and key agreement protocol which solved the key escrow problem. It is claimed that proposed protocol is resist to the adversary Type I and the adversary Type II. However, their protocol is not secure against the adversary Type 2. Therefore, it is imperative to introduce a provably secure user authentication and key agreement protocol using CL-PKC to ensure the scheme withstands the adversary Type 2. This paper is a revised version of work published in (Hassan et al. 2016).

1.2 Organization

This paper is organized as follows. Section 2 presents the preliminaries of bilinear pairing as well as the security model. Section 3 proposes our protocol and the security of the proposed protocol is shown in Sect. 4. Section 5 demonstrates the analysis of the protocol's performance while the conclusions are given in Sect. 6.

2 Preliminaries

2.1 Bilinear pairings

While $\mathbb{G}_1$ and $\mathbb{G}_2$ are the additive and multiplicative groups of exact prime order q respectively, the bilinear pairing function is given as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and P is the generator of $\mathbb{G}_1$. The following as described by (Boneh and Franklin 2001; Boneh et al. 2004) are properties of a bilinear pairing:

1. *Bilinearity* Where $\forall a, b \in \mathbb{Z}_q^*$ and $\forall Q, P \in \mathbb{G}_1$, then $e(aQ, bP) = e(Q, P)^{ab}$.
2. *Non degeneracy* Where $Q, P \in \mathbb{G}_1$ and $1_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$., there exist $e(Q, P) \neq 1_{\mathbb{G}_2}$.
3. *Computability* The $e(Q, P)$ is computed efficiently if $\forall Q, P \in \mathbb{G}_1$.

Computational Diffie-Hellman (CDH) Problem
Where $\forall a, b \in \mathbb{Z}_q^*$ and $(P, aP, bP) \in \mathbb{G}_1$ are given it is not easy to solve abP.

2.2 Security model

For user authentication and key agreement protocol proof there are several security models such as CK (Canetti and Krawczyk 2001) model and eCK model (LaMacchia et al. 2007). We have employed Wu and Tseng (2010)'s security model because it is suitable to our protocol. The efficiency of an adversary $\mathscr{A}_i$ where $\forall i \in \{1, 2\}$ and the security specification list for the mutual authentication and key exchange are described in this section. The schemes which use CL-PKC no doubt resists the two type adversaries, labeled type I and type II as noted in (Al-Riyami and Paterson 2003). While the type I adversary $\mathscr{A}_1$ can replace the users' public key, he/she cannot access the master key of the KGC. In the same line, though the type II adversary $\mathscr{A}_2$ own the KGC's master key, he/she nevertheless can't substitute the public key of the users.

The instance k of the member U is denoted as Π_U^k. Below, the following queries of game 1 and game 2 are illustrated:

– Game-1:

1. *Setup*(1^k) A certain security parameter k is given to the algorithm as input. Then, the system parameter *params* and the master key s are generated when the *Setup* algorithm is run by the challenger $\mathscr{C}$. Consequently, while s remains secret, $\mathscr{C}$ sends *params* to the adversary $\mathscr{A}_1$.
2. *Probing* Except the challenged identity ID_j, $\mathscr{A}_1$ performs a polynomial function for any identity ID_c of the under-listed queries:

   (a) *Extract partial private key query* Except the ID_j, $\mathscr{A}_1$ is capable of requesting the partial private key for any ID_c while $\mathscr{C}$ computes the corresponding partial private key D_{ID_c} and returns it to $\mathscr{A}_1$ accordingly.

   (b) *Extract private key query* Except the ID_j, $\mathscr{A}_1$ is capable of requesting the partial private key for any ID_c, $\mathscr{C}$ computes the corresponding private key and returns it to $\mathscr{A}_1$ accordingly.

   (c) *Request public key query* Except the ID_j, $\mathscr{A}_1$ is capable of requesting the public key for any ID_c while $\mathscr{C}$ computes the corresponding public key P_{ID_c} and returns it to $\mathscr{A}_1$.

   (d) *Replace public key* query: For any ID_c, $\mathscr{A}_1$ can choose a new secret value x'_{ID_c} and compute the new public key associated to the value x'_{ID_c}. Aftermath, $\mathscr{A}_1$ substitutes P_{ID_c} with P'_{ID_c}.

   (e) *Send* (Π_U^k, M) *query* When a message M is sent according to the proposed protocol from $\mathscr{A}_1$ to $\mathscr{C}$, $\mathscr{C}$ receives, makes the computation and responses to $\mathscr{A}_1$.

   (f) *Reveal* (Π_U^k, M) *query* A session key sk is received by $\mathscr{A}_1$ from $\mathscr{C}$ if it has been accepted. In the case it hasn't, it replies a null.

   (g) *Corrupt* (U) *query* A member U is issued a *Corrupt* query by $\mathscr{A}_1$ to $\mathscr{C}$ for the purpose of remitting its private key.

   (h) *Test* (Π_U^k) *query* When $\mathscr{A}_1$ forwards a single *Test* query to $\mathscr{C}$, $\mathscr{C}$ tosses a fair coin b. The session key sk is returned to $\mathscr{A}_1$ if $b = 1$. if not; a randomly string is returned to $\mathscr{A}_1$. The semantic security of sk is measured by this query.

   $\mathscr{A}_1$ outputs b' as its estimate for value of b in *Test* query. The value of b can be guessed correctly by $\mathscr{A}_1$ with probability $Adv(\mathscr{A}_1) = |\Pr[b' = b] - 1/2|$. The private key of ID_j can't be extracted by $\mathscr{A}_1$ at any point in this game. In addition, the public key of ID_j can't replaced by $\mathscr{A}_1$ before the challenge phase. But, the partial private key can be extracted by $\mathscr{A}_1$ in some phases.

– Game-2:

1. *Setup*(1^k) A certain security parameter k is given to the algorithm as input. Then, the system parameter *params* and the master key s are generated when the *Setup* algorithm is run by the challenger $\mathscr{C}$. Consequently, $\mathscr{C}$ sends *params* and s to $\mathscr{A}_2$.
2. *Probing* Except ID_j, $\mathscr{A}_2$ performs a polynomial function for any ID_c of the under-listed queries:
   The *Extract private key*, *Request public Key*, *Send* (Π_U^k, M), *Reveal* (Π_U^k, M), *Corrupt* (U), and *Test* (Π_U^k) queries are made by $\mathscr{A}_2$ similar to that in game-1. *Extract partial private key* query and *Replace public key* query can't be made by $\mathscr{A}_2$ in this game. However, $\mathscr{A}_2$ makes the extraction of the partial private key D_{ID_c} by oneself since, it has the master key s.

   $\mathscr{A}_2$ outputs b' estimate for value of b in *Test* query. The value of b can be guessed correctly by $\mathscr{A}_2$ with probability $Adv(\mathscr{A}_1) = |\Pr[b' = b] - 1/2|$. *Extract partial private key* and *Replace public key* queries of ID_j can't be made by $\mathscr{A}_2$ at any point in this game.

Send query, *Reveal* query, *Corrupt* query, and *Test* query can be made by $\mathscr{A}_i\{i = 1, 2\}$ for key exchange characteristics without authentication (Jakobsson and Pointcheval 2001). Figure that $\mathscr{A}_1$ and $\mathscr{A}_2$ are capable to establish limited queries under adaptive chosen message attacks (Choon and Cheon 2003). More details about security requirements for the mutual authentication and key exchange scheme could be found by the reader in (Boneh and Franklin 2001).

3 Proposed protocol

Our protocol is consisted of two phases, namely the initialization phase and the user authentication key exchange phase. He et al. (2013)'s short signature have employed by the proposed protocol. The notations apply in this paper are illustrated in Table 1. Our protocol's phases are described as follows:

3.1 Initialization phase

The following algorithms are executed during initialization as shown in Fig 1.

Table 1 Notation

Symbol	Description
k	A security parameter
$\mathbb{G}_1$	A cycle additive group
$\mathbb{G}_2$	A cycle 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 $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$
H_i	An one way hash function, where $\forall i \in \{1, 2, ..., 5\}$
s	A master secret key of RC
P_{pub}	A master public key of the server
ID_c	An identity of participants
D_{ID_c}	A partial private key of client
P_{ID_c}	A public key of the client
P_{ID_s}	A public key of the server
ID_j	A challenged identity

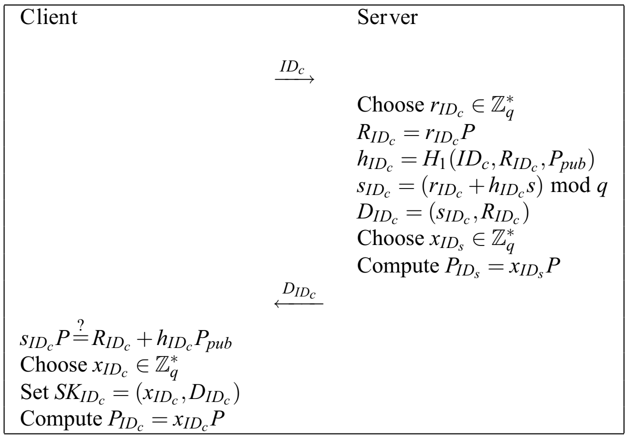
Client
Server
$\xrightarrow{ID_c}$
Choose $r_{ID_c} \in \mathbb{Z}_q^*$
$R_{ID_c} = r_{ID_c}P$
$h_{ID_c} = H_1(ID_c, R_{ID_c}, P_{pub})$
$s_{ID_c} = (r_{ID_c} + h_{ID_c}s) \bmod q$
$D_{ID_c} = (s_{ID_c}, R_{ID_c})$
Choose $x_{ID_s} \in \mathbb{Z}_q^*$
Compute $P_{ID_s} = x_{ID_s}P$
$\xleftarrow{D_{ID_c}}$
$s_{ID_c}P \overset{?}{=} R_{ID_c} + h_{ID_c}P_{pub}$
Choose $x_{ID_c} \in \mathbb{Z}_q^*$
Set $SK_{ID_c} = (x_{ID_c}, D_{ID_c})$
Compute $P_{ID_c} = x_{ID_c}P$

Fig. 1 Initialization phase

– *Setup* (1^k): This algorithm is executed by the key generator center (KGC). A security parameter k is taken by KGC while the parameters are generated as follows:

1. Two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are determined with exact prime order q and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where P is a generator of $\mathbb{G}_1$.
2. The master private key of the KGC is determined randomly $s \in \mathbb{Z}_q^*$ and the corresponding master public key $P_{pub} = sP$ is computed.
3. The public key $P_{ID_s} = x_{ID_s}P$ is computed where $x_{ID_s} \in \mathbb{Z}_q^*$ and four cryptographic secure hash functions $H_1 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_1 \times \{0,1\}^* \times \mathbb{Z}_q^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{Z}_q^* \times \mathbb{G}_1 \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, and $H_4 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{Z}_q^* \times \mathbb{G}_1 \times \mathbb{Z}_q^* \to \mathbb{G}_1$ are chosen.
4. $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, P_{pub}, P_{ID_s}, H_1, H_2, H_3, H_4\}$ are published as general.

– *Extract partial private key* The partial private key $R_{ID_c} = r_{ID_c}P$ where $r_{ID_c} \in \mathbb{Z}_q^*$, $h_{ID_c} = H_1(ID_c, R_{ID_c}, P_{pub})$, and $s_{ID_c} = (r_{ID_c} + h_{ID_c}s) \bmod q$ are computed by the server while a master private key, the public parameters and the identity of the client are given. The server's public key $P_{ID_s} = x_{ID_s}P$ is computed where $x_{ID_s} \in \mathbb{Z}_q^*$ is chosen randomly. Then, the D_{ID_c} is sent to the client.
– *Set private key* The reliability of the D_{ID_c} is checked by the client since it is received from the server by the following equation $s_{ID_c}P \overset{?}{=} R_{ID_c} + h_{ID_c}P_{pub}$. The client's private key $SK_{ID_c} = (x_{ID_c}, D_{ID_c})$ is computed by itself where $x_{ID_c} \in \mathbb{Z}_q^*$ is chosen randomly.
– *Set public key*: The client's public key $P_{ID_c} = x_{ID_c}P$ is computed by itself.

3.2 User authentication and key exchange phase

The interaction between the client and the server in this phase is displayed in Fig 2 while this phase is described as follows:

1. $U = rP$, $k_1 = rP_{pub}$, and $k_2 = rP_{ID_s}$ are computed by the client where $r \in \mathbb{Z}_q^*$ is chosen randomly. Then, (ID_c, U) is sent to the server.
2. $\alpha \in \mathbb{Z}_q^*$ is chosen randomly while $k_3 = sU$, $k_4 = x_{ID_s}U$, and $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$ are computed by the server since (ID_c, U) is received from the client. Finally, $(\alpha, Auth)$ is sent to the client.
3. $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$ is computed by the client since $(\alpha, Auth)$ is received from the server to ensure that the $Auth$ of the client is equal to the $Auth$

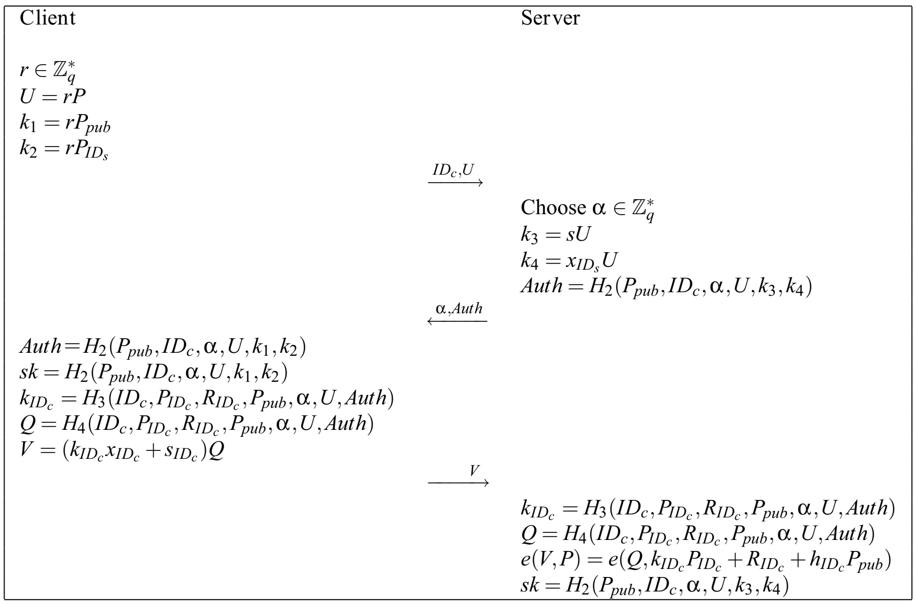
Client
Server
$r \in \mathbb{Z}_q^*$
$U = rP$
$k_1 = rP_{pub}$
$k_2 = rP_{ID_s}$
$\xrightarrow{ID_c, U}$
Choose $\alpha \in \mathbb{Z}_q^*$
$k_3 = sU$
$k_4 = x_{ID_s}U$
$Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$
$\xleftarrow{\alpha, Auth}$
$Auth = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$
$sk = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$
$k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$
$Q = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$
$V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q$
$\xrightarrow{V}$
$k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$
$Q = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$
$e(V, P) = e(Q, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$
$sk = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$

Fig. 2 User authentication and key exchange phase

that is computed in the server. Then, the session key $sk = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$ is computed while k_{ID_c}, Q and V are calculated as follows:

$$k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth) \quad (1)$$

$$Q = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth) \quad (2)$$

$$V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q \quad (3)$$

Finally, V is sent to the server.

4. $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ and $Q = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ are computed by the server since V is received from the client. Then, the reliability of V is checked by the following equality

$$e(V, P) = e(Q, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub}) \quad (4)$$

The session key $sk = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$ is computed by the server if the V is satisfied the above equality.

3.3 Correctness of our protocol

Where $P_{ID_c} = x_{ID_c}P$ is the client's public key, $s_{ID_c}P = R_{ID_c} + h_{ID_c}P_{pub}$ can be computed by the client to verify the partial private key and $V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q$ is the signature are given while V satisfy this equality $e(V, P) = e(Q, k_{ID_c}\ P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$ the proposed protocol is correct. Using the bilinear pairing properties the correctness is illustrated by the following equalities:

$$\begin{aligned} e(V, P) &= e((k_{ID_c}x_{ID_c} + s_{ID_c})Q, P) \\ &= e(Q, (k_{ID_c}x_{ID_c} + s_{ID_c})P) \\ &= e(Q, (k_{ID_c}x_{ID_c}P + s_{ID_c}P)) \\ &= e(Q, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub}) \end{aligned}$$

4 Security analysis

This section shows our protocol achieves the security requirements in Sect. 2, which is proved in the random oracle model (Bellare and Rogaway 1993). The approach in (Wu and Tseng 2010; He et al. 2013) is employed by our security proof.

4.1 Client-to-server authentication

Theorem 1 demonstrates that the communication from the client to the server can't be impersonated by $\mathcal{A}_i$ where $\{i = 1, 2\}$ under CDH assumption. The proof is given by the declaration in Lemma 1 and 2.

Theorem 1 *We assume the client-to-server authentication is broken by* $\mathcal{A}_i$ where $\{i = 1, 2\}$ *with a non-negligible advantage* ε. *Then, the CDH assumption is solved by an algorithm* $\mathcal{C}$ *with a non-negligible probability. At most* q_S *queries to the oracle* Π_s^i of the server , q_c *queries to the oracle* Π_c^j *of the client, and* q_{H_i} *queries on* H_i *oracle* $\forall i \in \{1, 2, .., 5\}$ *are made by* $\mathcal{C}$.

Proof Suppose that the client-to-server authentication of the presented protocol is broken by $\mathcal{A}_i$ with a non-negligible ε advantage while a polynomial time is given under adaptive chosen message and identity attacks. Then, by Lemma 1 in (Choon and Cheon 2003), for a chosen challenge identity the client-to-server authentication of our protocol is owned with ε by $\mathcal{A}_i$ while a polynomial time is given.

Lemma 1 *The proposed protocol can't be broken under CDH assumption by the type I adversary $\mathscr{A}_1$ in the random oracle model.*

The queries of $\mathscr{A}_1$ is answered by the algorithm $\mathscr{C}$ to prove Lemma 1. We assume that the random elements (P, $X = aP$, and $Y = bP$) $\in \mathbb{G}_1$ are received by $\mathscr{C}$ while $\forall a, b \in \mathbb{Z}_q^*$ are unknown values. The value of abP is derived by $\mathscr{C}$ with answering the $\mathscr{A}_1$'s oracle queries as follows:

- *Setup* (1^k) The system parameters $\{e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3, H_4\}$ are generated by $\mathscr{C}$ where $P_{pub} = Y$ and are sent to $\mathscr{A}_1$. An identity ID_j is selected randomly by $\mathscr{C}$ as the challenge identity in this game. For queries and responses six lists L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4}, L_{PK_1} and L_{PK_2} are maintained by $\mathscr{C}$ to avoid collision and consistency.
- H_1*query* When the adversary $\mathscr{A}_1$ submits this query on $(ID_c, R_{ID_c}, P_{pub})$, $\mathscr{C}$ randomly chooses $h_1 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_1} with $(ID_c, R_{ID_c}, P_{pub}, h_1)$.
- H_2*query* When the adversary $\mathscr{A}_1$ submits this query on $(P_{pub}, ID_c, \alpha, U, k_1, k_2)$, $\mathscr{C}$ randomly chooses $h_2 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_2} with $(P_{pub}, ID_c, \alpha, U, k_1, k_2, h_2)$.
- H_3*query* When the adversary $\mathscr{A}_1$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, $\mathscr{C}$ randomly chooses $h_3 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_3} with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, h_3)$.
- H_4*query* : When the adversary $\mathscr{A}_1$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, the following reactions are done by $\mathscr{C}$:

  1. If L_{H_4} consists of $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$, $\mathscr{C}$ returns Q_i to $\mathscr{A}_1$.
  2. Else, a coin $c_i \in \{0, 1\}$ is flipped with $\Pr[c_i = 0] = 1/(q_s + 1)$ and $Q_i = (1 - c_i)Y + t_iP$ is computed by $\mathscr{C}$ while $t_i \in \mathbb{Z}_q^*$ is generated randomly. Finally, $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ is added to L_{H_4} and Q_i is returned to $\mathscr{A}_1$ by $\mathscr{C}$.

- *Extract partial private key query* When the adversary $\mathscr{A}_1$ submits this query on ID_c, $\mathscr{C}$ generates two random numbers $a_{ID_c}, b_{ID_c} \in \mathbb{Z}_n^*$, sets $R_{ID_c} = a_{ID_c}P$, $h_{ID_c} = H_1(ID_c, R_{ID_c}, P_{pub}) = b_{ID_c}$, $r_{ID_c} = a_{ID_c}$ and $s_{ID_c} = r_{ID_c} + h_{ID_c}s$. Then, $\mathscr{C}$ adds $(ID_{ID_c}, R_{ID_c}, h_{ID_c})$ and $(ID_{ID_c}, R_{ID_c}, s_{ID_c})$ to L_{H_1} and L_{PK_1} separately.
- *Extract private key query* When $\mathscr{A}_1$ submits this query on ID_c, the following reactions are done by $\mathscr{C}$:

  1. If $ID_c \neq ID_j$, $\mathscr{C}$ searches in list L_{PK_2} and returns full private key (x_{ID_c}, D_{ID_c}) to $\mathscr{A}_1$.
  2. Otherwise, $\mathscr{C}$ stops and terminates.

- *Request public key query* When this query on ID_j is submitted by $\mathscr{A}_1$, $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ are looked in L_{PK_1} and L_{PK_2} respectively by $\mathscr{C}$. Then, $PK_{ID_c} = (P_{ID_c}, R_{ID_c})$ is computed by $\mathscr{C}$. Otherwise, *Extract private key* query is made by $\mathscr{C}$. Finally, PK_{ID_c} is returned to $\mathscr{A}_1$.
- *Replace public key query* Whenever $\mathscr{A}_1$ submits this query on $(ID_j, P'_{ID_j}, R'_{ID_c})$, $\mathscr{C}$ searches in L_{PK_1} and L_{PK_2} for tuples $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$. $\mathscr{C}$ sets $P_{ID_c} = P'_{ID_c}$, $R_{ID_c} = R'_{ID_c}$, $s_{ID_c} = \perp$ and $x_{ID_c} = \perp$.
- *Send* query:

  1. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_C^j, \}\}\text{start}'')$ query, $\mathscr{C}$ randomly chooses $U, k_1, k_2 \in G_1$ and returns (ID_j, U) to $\mathscr{A}_1$.
  2. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_S^i, (ID_j, U))$ query to the server and $ID_c \neq ID_j$, $\mathscr{C}$ randomly chooses an integer $\alpha \in \mathbb{Z}_q^*$, computes $k_3 = sU$, $k_4 = x_{ID_s}U$, and $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$. Otherwise $ID_c = ID_j$, $\mathscr{C}$ fails and terminates. Denote that the simulator does not know the s. Finally $\mathscr{C}$ returns $(\alpha, Auth)$ to $\mathscr{A}_1$.
  3. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_c^j, (\alpha, Auth))$ query to the client and $ID_c \neq ID_j$, $\mathscr{C}$ checks whether $Auth \overset{?}{=} H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$. If it holds, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2}, respectively. $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, carries out H_4 query, and obtains $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. Let $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ be the corresponding tuple on L_{H_4}. If $c_i = 0$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_i = 1$ and $Q_i = t_iP$ are set. $\mathscr{C}$ defines $V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q$. Then, $\mathscr{C}$ returns V to $\mathscr{A}_1$. If $ID_c = ID_j$, $\mathscr{C}$ is achieved correctly since $\mathscr{A}_1$ is not capable to satisfy $Auth'$ because of the lack of k_1 and k_2.
  4. When the adversary $\mathscr{A}_1$ submits *Send* (Π_S^i, V) query to the server, $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ and $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. $\mathscr{C}$ checks $e(V_i, P) = e(Q_i, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$. If it holds, $\mathscr{C}$ accepts and terminates. Else, $\mathscr{C}$ terminates.

Forgery Eventually, $\mathscr{A}_1$ outputs a valid signature (ID_c, V'). If $ID_c \neq ID_j$, $\mathscr{C}$ stops the simulation. Otherwise, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2}, respectively. Then, $Q_n = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ is computed by $\mathscr{C}$ while this equality

$e(V_n, P) = e(Q_n, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$ is tested to ensure that the V_n is satisfied the equality. If is satisfied, the simulation is terminated by $\mathscr{C}$ while PK_{ID_c} is noted as the original public key. The corresponding tuple on L_{H_4} is assumed with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_n, t_n, c_n)$. If $c_n = 1$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_n = 0$ and $Q_n = bP + t_iP$ is computed. Therefore, $V_i = t_i(b + t_iP)(k_{ID_c}a + r_{ID_c} + h_{ID_c}s)P$. As done in Forking Lemma (Pointcheval and Stern 1996, 2000), $\mathscr{C}$ outputs $(t_ik_{ID_c})^{-1}(V_i - t_i^2(k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}X) - t_i(r_{ID_c} + h_{ID_c}s)Y)$ as the solution of the CDH assumption.

Analysis In the analysis, we are interested in the probability of the following independent events:

E_1:the challenger $\mathscr{C}$ does not abort any of $\mathscr{A}_1$'s signature queries.

E_2: The adversary $\mathscr{A}_1$ successfully falsifies the $e(V_i, P) = e(Q_i, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$ on m_n.

E_3: The $\mathscr{A}_1$ falsifies V' and satisfies $ID_c = ID_j$.

E_4: E_2 happens and $c_n = 0$ in L_{H_4}.

The signature queries q_s and the hash queries q_H are made by $\mathscr{A}_1$ to $\mathscr{C}$. Then, these probability we have $\Pr[E_1] \geq (1 - 1/(q_S + 1))^{q_s}$, $\Pr[E_2|E_1] \geq \varepsilon$, $\Pr[E_3|E_1 \wedge E_2] \geq 1/q_H$ and $\Pr[E_4|E_1 \wedge E_2 \wedge E_3] = 1/q_S + 1$. Since ε is non-negligible, then $\mathscr{A}_1$ solves the CDHP with the non-negligible probability

$$\begin{aligned}&\Pr[E_1 \wedge E_2 \wedge E_3 \wedge E_4]\\&\Pr[E_1]\Pr[E_2|E_1]\Pr[E_3|E_1 \wedge E_2]\Pr[E_4|E_1 \wedge E_2 \wedge E_3]\\&\quad\geq ((1 - 1/q_S + 1)^{q_s}/q_H(q_S + 1))\varepsilon\end{aligned}$$

Therefore, the client-to-server authentication scheme is broken by $\mathscr{A}_1$ while the CDH assumption is solved by an algorithm $\mathscr{C}$. □

Lemma 2 *The proposed protocol can't be broken under CDH assumption by the type II adversary $\mathscr{A}_2$ in the random oracle model.*

Proof The queries of $\mathscr{A}_2$ is answered by the algorithm $\mathscr{C}$ to prove Lemma 2. We assume that the random elements (P, $X = aP$, and $Y = bP$) $\in \mathbb{G}_1$ are received by $\mathscr{C}$ while $\forall a, b \in \mathbb{Z}_q^*$ are unknown values. The value of abP is derived by $\mathscr{C}$ with answering the $\mathscr{A}_2$'s oracle queries as follows:

- *Setup* (1^k) The system parameters $\{e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3, H_4\}$ are generated by $\mathscr{C}$ where $P_{pub} = Y$ and are sent to $\mathscr{A}_2$. An identity ID_j is selected randomly by $\mathscr{C}$ as the challenge identity in this game. For queries and responses six lists L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4}, L_{PK_1} and L_{PK_2} are maintained by $\mathscr{C}$ to avoid collision and consistency.
- H_1*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, R_{ID_c}, P_{pub})$, $\mathscr{C}$ randomly chooses $h_1 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_1} with $(ID_c, R_{ID_c}, P_{pub}, h_1)$.
- H_2*query* When the adversary $\mathscr{A}_2$ submits this query on $(P_{pub}, ID_c, \alpha, U, k_1, k_2)$, $\mathscr{C}$ randomly chooses $h_2 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_2} with $(P_{pub}, ID_c, \alpha, U, k_1, k_2, h_2)$.
- H_3*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, $\mathscr{C}$ randomly chooses $h_3 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_3} with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, h_3)$.
- H_4*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, the following reactions are done by $\mathscr{C}$:

  1. If L_{H_4} consists of $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$, $\mathscr{C}$ returns Q_i to $\mathscr{A}_2$.
  2. Else, a coin $c_i \in \{0, 1\}$ is flipped with $\Pr[c_i = 0] = 1/(q_s + 1)$ and $Q_i = (1 - c_i)Y + t_iP$ is computed by $\mathscr{C}$ while $t_i \in \mathbb{Z}_q^*$ is generated randomly. Finally, $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ is added to L_{H_4} and Q_i is returned to $\mathscr{A}_2$ by $\mathscr{C}$.

- *Extract private key query* When The adversary $\mathscr{A}_2$ submits this query on ID_c, the following reactions are done by $\mathscr{C}$:

  1. If $ID_c \neq ID_j$, $\mathscr{C}$ searches in L_{PK_2} and returns full private key (x_{ID_c}, D_{ID_c}) to $\mathscr{A}_2$.
  2. Otherwise, $\mathscr{C}$ stops and terminates.

- *Request public key query* When this query on ID_j is submitted by $\mathscr{A}_2$, $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ are looked in L_{PK_1} and L_{PK_2} respectively by $\mathscr{C}$. Then, $PK_{ID_c} = (P_{ID_c}, R_{ID_c})$ is computed by $\mathscr{C}$. Otherwise, *Extract private key* query is made by $\mathscr{C}$. Finally, PK_{ID_c} is returned to $\mathscr{A}_2$. When the adversary $\mathscr{A}_2$ submits this query on ID_j, $\mathscr{C}$ searches in L_{PK_1} and L_{PK_2} for tuples $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$. $\mathscr{C}$ returns $P_{ID_c} = (P_{ID_c}, R_{ID_c})$ to $\mathscr{A}_2$. Otherwise $\mathscr{C}$ makes *Extract private key* query and returns P_{ID_c} to $\mathscr{A}_2$.
- *Send* query:

  1. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi_C^j, \}\}\text{start}'')$ query, $\mathscr{C}$ randomly chooses $U, k_1, k_2 \in G_1$ and returns (ID_j, U) to $\mathscr{A}_2$.
  2. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi_S^i, (ID_j, U))$ query to the server and $ID_c \neq ID_j$, $\mathscr{C}$ randomly chooses an integer $\alpha \in \mathbb{Z}_q^*$, computes $k_3 = sU$, $k_4 = x_{ID_s}U$, and $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$. Note that the simulator does not know the s. Oth-

erwise $ID_c = ID_j$, $\mathscr{C}$ fails and terminates. Finally $\mathscr{C}$ returns $(\alpha, Auth)$ to $\mathscr{A}_2$.

3. When the adversary $\mathscr{A}_2$ submits $Send(\Pi^j_C, (\alpha, Auth))$ query to the client and $ID_c \neq ID_j$, $\mathscr{C}$ checks whether $Auth \overset{?}{=} H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$. If it holds, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2} respectively. $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, carries out H_5 query and obtains $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. Let $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ be the corresponding tuple on L_{H_4}. If $c_i = 0$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_i = 1$ and $Q_i = t_iP$ are set. $\mathscr{C}$ defines $V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q$. Then, $\mathscr{C}$ returns V to $\mathscr{A}_2$. If $ID_c = ID_j$, $\mathscr{C}$ is achieved correctly since $\mathscr{A}_2$ is not capable to satisfy $Auth'$ because of the lack of k_1 and k_2.
4. When the adversary $\mathscr{A}_2$ submits $Send(\Pi^i_S, (V))$ query to the server, $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ and $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. $\mathscr{C}$ checks $e(V_i, P) = e(Q_i, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$. If it holds, $\mathscr{C}$ accepts and terminates. Else, $\mathscr{C}$ terminates.

Forgery Eventually, $\mathscr{A}_2$ outputs a valid signature (ID_c, V'). If $ID_c \neq ID_j$, $\mathscr{C}$ stops the simulation. Otherwise, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2}, respectively. Then, $Q_n = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ is computed by $\mathscr{C}$ while this equality $e(V_n, P) = e(Q_n, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$ is tested to ensure that the V_n is satisfied the equality. If is satisfied, the simulation is terminated by $\mathscr{C}$ while PK_{ID_c} is noted as the original public key. The corresponding tuple on L_{H_4} is assumed with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_n, t_n, c_n)$. If $c_n = 1$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_n = 0$ and $Q_n = bP + t_iP$ is computed. Therefore, $V_i = t_i(b + t_iP)(k_{ID_c}a + r_{ID_c} + h_{ID_c}s)P$. As done in Forking Lemma (Pointcheval and Stern 1996, 2000), $\mathscr{C}$ outputs $(t_ik_{ID_c})^{-1}(V_i - t_i^2(k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}X) - t_i(r_{ID_c} + h_{ID_c}s)Y)$ as the solution of the CDH assumption.

Analysis: Analysis is similar to the description in Lemma 1.

□

4.2 Key exchange

The following Theorem 2 demonstrates that a key exchange is provided by our protocol under CDH assumption. The proof is given by the declaration in Lemma 1 and 2.

Theorem 2 *We assume that the value of b in Test query can be guessed by $\mathscr{A}_i$ where $\{i = 1, 2\}$ with a non-negligible advantage ε. Then, the CDH assumption is solved by a challenger $\mathscr{C}$ with a non-negligible probability. At most q_S qu-eries to the oracle Π^i_s of the server, q_c queries to the oracle Π^j_C of the client, and q_{H_i} queries on H_i oracle $\forall i \in \{1, 2, ..., 5\}$ are made by $\mathscr{C}$.*

Lemma 1 *The proposed protocol can't be broken under CDH assumption by the type I adversary $\mathscr{A}_1$ in the random oracle model.*

Proof The coin in the *Test* query can be perfectly guessed with the probability 1 / 2 by $\mathscr{A}_1$. If the coin can be guessed by $\mathscr{A}_1$ with advantage ε, then the session key can be obtained correctly with advantage $\Pr[Ocsk] \geq \varepsilon/2$. The event that can achieving the correct session key is denoted by Osk. The success events of the oracle Π^j_C of the client and the oracle Π^i_S of the server are assumed by $Test(C^j)$ and $Test(S^i)$ respectively, while the event that succeed to break the client-server authentication is assumed by E^{C2S}. *Test* query may be submitted by $\mathscr{A}_1$ to the client, then the following probability is had

$$\Pr[Ocsk \wedge Test(S^i) \wedge E^{C2S}] + \Pr[Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}] + \Pr[Ocsk \wedge Test(C^j)] \geq \frac{\varepsilon}{2}$$

for specific j and i. Then, the following probability is had

$$\Pr[Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}] + \Pr[Ocsk \wedge Test(C^j)] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$$

for specific j and i. The queries of $\mathscr{A}_1$ is answered by the algorithm $\mathscr{C}$ to prove Lemma 1. We assume that the random elements $(P, X = aP$, and $Y = bP) \in \mathbb{G}_1$ are received by $\mathscr{C}$ while $\forall a, b \in \mathbb{Z}^*_q$ are unknown values. The value of abP is derived by $\mathscr{C}$ with answering the $\mathscr{A}_1$'s oracle queries as follows:

- *Setup* (1^k) The system parameters $\{e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3, H_4\}$ are generated by $\mathscr{C}$ where $P_{pub} = Y$ and are sent to $\mathscr{A}_1$. An identity ID_j is selected randomly by $\mathscr{C}$ as the challenge identity in this game. For queries and responses six lists L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4}, L_{PK_1} and L_{PK_2} are maintained by $\mathscr{C}$ to avoid collision and consistency.
- H_1 query: When the adversary $\mathscr{A}_1$ submits this query on $(ID_c, R_{ID_c}, P_{pub})$. $\mathscr{C}$ randomly chooses $h_1 \in \mathbb{Z}^*_q$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_1} with $(ID_c, R_{ID_c}, P_{pub}, h_1)$.

- H_2 query: Whenever $\mathscr{A}_1$ submits this query on $(P_{pub}, ID_c, \alpha, U, k_1, k_2)$, $\mathscr{C}$ randomly chooses $h_2 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_2} with $(P_{pub}, ID_c, \alpha, U, k_1, k_2, h_2)$.
- H_3 query: When the adversary $\mathscr{A}_1$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, $\mathscr{C}$ randomly chooses $h_3 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_1$. $\mathscr{C}$ updates L_{H_3} with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, h_3)$.
- H_4 query: the following reactions are done by $\mathscr{C}$:

  1. If L_{H_4} consists of $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$, $\mathscr{C}$ returns Q_i to $\mathscr{A}_1$.
  2. Else, a coin $c_i \in \{0, 1\}$ is flipped with $\Pr[c_i = 0] = 1/(q_s + 1)$ and $Q_i = (1 - c_i)Y + t_iP$ is computed by $\mathscr{C}$ while $t_i \in \mathbb{Z}_q^*$ is generated randomly. Finally, $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ is added to L_{H_4} and Q_i is returned to $\mathscr{A}_1$ by $\mathscr{C}$.

- *Extract partial private key query* When the adversary $\mathscr{A}_1$ submits this query on ID_c, $\mathscr{C}$ generates two random numbers $a_{ID_c}, b_{ID_c} \in \mathbb{Z}_n^*$, sets $R_{ID_c} = a_{ID_c}P$, $h_{ID_c} = H_1(ID_c, R_{ID_c}, P_{pub}) = b_{ID_c}$, $r_{ID_c} = a_{ID_c}$, and $s_{ID_c} = r_{ID_c} + h_{ID_c}s$. Then, $\mathscr{C}$ adds $(ID_{ID_c}, R_{ID_c}, h_{ID_c})$ and $(ID_{ID_c}, R_{ID_c}, s_{ID_c})$ to L_{H_1} and L_{PK_1}, separately.
- *Extract private key query* When $\mathscr{A}_1$ submits this query on ID_c, the following reactions are done by $\mathscr{C}$:

  1. If $ID_c \neq ID_j$, $\mathscr{C}$ searches in list L_{PK_2} and returns full private key (x_{ID_c}, D_{ID_c}) to $\mathscr{A}_1$.
  2. Otherwise, $\mathscr{C}$ stops and terminates.

- *Request public key* query: When this query on ID_j is submitted by $\mathscr{A}_1$, $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ are looked in L_{PK_1} and L_{PK_2} respectively by $\mathscr{C}$. Then, $PK_{_{ID_c}} = (P_{ID_c}, R_{ID_c})$ is computed by $\mathscr{C}$. Otherwise, *Extract private key* query is made by $\mathscr{C}$. Finally, $PK_{_{ID_c}}$ is returned to $\mathscr{A}_1$.
- *Replace public key* query: Whenever $\mathscr{A}_1$ submits this query on $(ID_j, P'_{ID_j}, R'_{ID_c})$, $\mathscr{C}$ searches in L_{PK_1} and L_{PK_2} for tuples $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$. $\mathscr{C}$ sets $P_{ID_c} = P'_{ID_c}$, $R_{ID_c} = R'_{ID_c}$, $s_{ID_c} = \perp$ and $x_{ID_c} = \perp$.
- *Send* query:

  1. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_C^j, \}\}\text{start}'')$ query, $\mathscr{C}$ randomly chooses $r \in \mathbb{Z}_q^*$. $\mathscr{C}$ computes $U = rP, k_1 = rP_{pub}, k_2 = rP_{ID_s}$, and returns (ID_j, U) to $\mathscr{A}_1$.
  2. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_S^i, (ID_j, U))$ query to the server and $ID_c \neq ID_j$, $\mathscr{C}$ randomly chooses an integer $\alpha \in \mathbb{Z}_q^*$, computes $k_3 = sU$, $k_4 = x_{ID_c}U$ and $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$. Note that the s isn't known by the. Otherwise $ID_c = ID_j$, $\mathscr{C}$ fails and terminates. Finally, $\mathscr{C}$ returns $(\alpha, Auth)$ to $\mathscr{A}_1$.
  3. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_C^j, (\alpha, Auth))$ query to the client and $ID_c \neq ID_j$, $\mathscr{C}$ checks whether $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$. If it holds, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2}, respectively. $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, carries out H_4 query and obtains $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. Let $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ be the corresponding tuple in the list L_{H_4}. If $c_i = 0$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_i = 1$ and $Q_i = t_iP$ are set. $\mathscr{C}$ defines $V = (k_{ID_c}x_{ID_c} + s_{ID_c})Q$, then $\mathscr{C}$ returns V to $\mathscr{A}_1$. If $ID_c = ID_j$, $\mathscr{C}$ is achieved correctly since $\mathscr{A}_1$ is not capable to satisfy $Auth'$ because of the lack of k_1 and k_2.
  4. When the adversary $\mathscr{A}_1$ submits *Send* $(\Pi_S^i, (V))$ query, to the server, $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ and $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. $\mathscr{C}$ checks $e(V_i, P) = e(Q_i, k_{ID_c}P_{ID_c} + R_{ID_c} + h_{ID_c}P_{pub})$. If it holds, $\mathscr{C}$ accepts and terminates. Else, $\mathscr{C}$ terminates.

- *Corrupt query* If the adversary $\mathscr{A}_1$ submits a *Corrupt* query on ID_c, $\mathscr{C}$ returns D_{ID_c}.
- *Reveal query* When the adversary $\mathscr{A}_1$ submits a *Reveal* query and the oracle of the participate u for $u \in c, s$ has accepted, $\mathscr{C}$ returns the session key sk.
- *Test query* When the adversary $\mathscr{A}_1$ submits a *Test* query and the query is not requesting, $\mathscr{C}$ sets $U = X$ as an instance of CDH assumption and aborts. Else, a fair coin b is flipped by $\mathscr{C}$. if $b = 1$. if not; a randomly string is returned to $\mathscr{A}_1$.

From above interactions with $\mathscr{A}_1$, without the occurs of E^{C2S}, $\mathscr{C}$ is correctly indistinguishable from our scheme. Note that the events $\exists i, Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}$ and $\exists j, Ocsk \wedge Test(C^j)$ are equivalent, then we have

$$\Pr[Ocsk \wedge Test(C^j)] \geq \varepsilon/2 - \Pr_{C2S}$$

also, we have the following probability

$$\Pr\left[sk = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)\Big|_{U, k_1, k_2 \leftarrow G_1}^{\alpha \in \mathbb{Z}_q^*}\right] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$$

Assume that if ε is non-negligible, then the $(\varepsilon/2) - \Pr_{C2S}$ is ε since the $\Pr_{C2S}$ is negligible by Theorem 1. Assume that the coin b in *Test* query is guessed with an advantage ε by $\mathscr{A}_1$. Given $(P, P_{pub} = bP = Y, U = aP = X)$ to $\mathscr{A}_1$. The instances $abP = k_1$, $abP = k_2$, $abP = k_3$, and $abP = k_4$ can be computed with an advantage ε by $\mathscr{A}_1$. The CDH assumption is resolved with a non-negligible advantage while $\mathscr{A}_1$ is used by $\mathscr{C}$. Indeed, our protocol provides a secure key exchange. □

Lemma 2 *The proposed protocol can't be broken under CDH assumption by the type II adversary $\mathscr{A}_1$ in the random oracle model.*

Proof The coin in the *Test* query can be perfectly guessed with the probability 1 / 2 by $\mathscr{A}_1$. If the coin can be guessed by $\mathscr{A}_1$ with an advantage ε, then the session key can be obtained correctly with advantage $\Pr[Ocsk] \geq \varepsilon/2$. The event that can achieving the correct session key is denoted by *Osk*. The success events of the oracle Π_C^j of the client and the oracle Π_S^i of the server are assumed by $Test(C^j)$ and $Test(S^i)$ respectively, while the event that succeed to break the client-server authentication is assumed by E^{C2S}. *Test* query may be submitted by $\mathscr{A}_1$ to the client, then the following probability is had

$$\Pr[Ocsk \wedge Test(S^i) \wedge E^{C2S}] + \Pr[Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}] + \Pr[Ocsk \wedge Test(C^j)] \geq \frac{\varepsilon}{2}$$

for specific j and i. Then, the following probability is had

$$\Pr[Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}] + \Pr[Ocsk \wedge Test(C^j)] \geq \frac{\varepsilon}{2} - \Pr_{C2S}$$

for specific j and i. The queries of $\mathscr{A}_2$ is answered by the algorithm $\mathscr{C}$ to prove Lemma 1. We assume that the random elements (P, $X = aP$, and $Y = bP$) $\in \mathbb{G}_1$ are received by $\mathscr{C}$ while $\forall a, b \in \mathbb{Z}_q^*$ are unknown values. The value of abP is derived by $\mathscr{C}$ with answering the $\mathscr{A}_2$'s oracle queries as follows:

- Setup (1^k) The system parameters $\{e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3, H_4\}$ are generated by $\mathscr{C}$ where $P_{pub} = Y$ and are sent to $\mathscr{A}_2$. An identity ID_j is selected randomly by $\mathscr{C}$ as the challenge identity in this game. For queries and responses six lists L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4}, L_{PK_1} and L_{PK_2} are maintained by $\mathscr{C}$ to avoid collision and consistency.
- H_1*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, R_{ID_c}, P_{pub})$. $\mathscr{C}$ randomly chooses $h_1 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_1} with $(ID_c, R_{ID_c}, P_{pub}, h_1)$.
- H_2*query* When the adversary $\mathscr{A}_2$ submits this query on $(P_{pub}, ID_c, \alpha, U, k_1, k_2)$, $\mathscr{C}$ randomly chooses $h_2 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_2} with $(P_{pub}, ID_c, \alpha, U, k_1, k_2, h_2)$.
- H_3*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, P_{ID_c} R_{ID_c}, P_{pub}, \alpha, U, Auth)$, $\mathscr{C}$ randomly chooses $h_3 \in \mathbb{Z}_q^*$ and returns it to $\mathscr{A}_2$. $\mathscr{C}$ updates L_{H_3} with $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, h_3)$.
- H_4*query* When the adversary $\mathscr{A}_2$ submits this query on $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, the following reactions are done by $\mathscr{C}$:

  1. If L_{H_4} consists of $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$, $\mathscr{C}$ returns Q_i to $\mathscr{A}_2$.
  2. Else, a coin $c_i \in \{0,1\}$ is flipped with $\Pr[c_i = 0] = 1/(q_s + 1)$ and $Q_i = (1 - c_i)Y + t_i P$ is computed by $\mathscr{C}$ while $t_i \in \mathbb{Z}_q^*$ is generated randomly. Finally, $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ is added to L_{H_4} and Q_i is returned to $\mathscr{A}_2$ by $\mathscr{C}$.

- *Extract private key* query: When The adversary $\mathscr{A}_2$ submits this query on ID_c, the following reactions are done by $\mathscr{C}$:

  1. If $ID_c \neq ID_j$, $\mathscr{C}$ searches in L_{PK_2} and returns full private key (x_{ID_c}, D_{ID_c}) to $\mathscr{A}_2$.
  2. Otherwise, $\mathscr{C}$ stops and terminates.

- *Request public key* query: When this query on ID_j is submitted by $\mathscr{A}_2$, $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ are looked in L_{PK_1} and L_{PK_2} respectively by $\mathscr{C}$. Then, $PK_{ID_c} = (P_{ID_c}, R_{ID_c})$ is computed by $\mathscr{C}$. Otherwise, *Extract private key* query is made by $\mathscr{C}$. Finally, PK_{ID_c} is returned to $\mathscr{A}_2$. When the adversary $\mathscr{A}_2$ submits this query on ID_j, $\mathscr{C}$ searches in L_{PK_1} and L_{PK_2} for tuples $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$. $\mathscr{C}$ returns $P_{ID_c} = (P_{ID_c}, R_{ID_c})$ to $\mathscr{A}_2$. Otherwise $\mathscr{C}$ makes *Extract private key* query and returns P_{ID_c} to $\mathscr{A}_2$.
- *Send query*

  1. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi_C^j, \}\}\text{start}'')$ query, $\mathscr{C}$ randomly chooses $w \in \mathbb{Z}_q^*$ and $\mathscr{C}$ computes $U = rP, k_1 = rP_{pub}, k_2 = rP_{ID_s}$ and returns (ID_j, U) to $\mathscr{A}_2$.
  2. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi_S^i, (ID_j, U))$ query to the server and $ID_c \neq ID_j$, $k_3 = sU$, $k_4 = x_{ID_s} U$, and $Auth = H_2(P_{pub}, ID_c, \alpha, U, k_3, k_4)$ are computed by $\mathscr{C}$ while $\alpha \in \mathbb{Z}_q^*$ is chosen randomly. Note that the s isn't known by the simulator. Otherwise $ID_c = ID_j$, $\mathscr{C}$ fails and terminates. Finally, $\mathscr{C}$ returns $(\alpha, Auth)$ to $\mathscr{A}_2$.
  3. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi_C^j, (\alpha, Auth))$ query to the client and $ID_c \neq ID_j$, $\mathscr{C}$ checks whether $Auth \overset{?}{=} H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)$. If it holds, $\mathscr{C}$ finds $(ID_c, s_{ID_c}, R_{ID_c})$ and $(ID_c, x_{ID_c}, P_{ID_c})$ in L_{PK_1} and L_{PK_2}, respectively. $\mathscr{C}$ computes

$k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$, carries out H_4 query and obtains $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. Let $(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth, Q_i, t_i, c_i)$ be the corresponding tuple on the list L_{H_4}. If $c_i = 0$, the simulation is failed and stopped by $\mathscr{C}$. Else, $c_i = 1$ and $Q_i = t_i P$ are set. $\mathscr{C}$ defines $V = (k_{ID_c} x_{ID_c} + s_{ID_c})Q$ then $\mathscr{C}$ returns V to $\mathscr{A}_2$. If $ID_c = ID_j$, $\mathscr{C}$ works perfectly since $\mathscr{A}_2$ is unable to verify $Auth'$ because of the lack of k_1 and k_2.

4. When the adversary $\mathscr{A}_2$ submits *Send* $(\Pi^i_S, (V))$ query, to the server. $\mathscr{C}$ computes $k_{ID_c} = H_3(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$ and $Q_i = H_4(ID_c, P_{ID_c}, R_{ID_c}, P_{pub}, \alpha, U, Auth)$. $\mathscr{C}$ checks $e(V_i, P) = e(Q_i, k_{ID_c} P_{ID_c} + R_{ID_c} + h_{ID_c} P_{pub})$. If it holds, $\mathscr{C}$ accepts and terminates. Else, $\mathscr{C}$ terminates.

– *Corrupt query* If the adversary $\mathscr{A}_2$ submits *Corrupt* query on ID_c, $\mathscr{C}$ returns the corresponding D_{ID_c}.
– *Reveal query* When the adversary $\mathscr{A}_2$ submits *Reveal* query and the oracle of the participate u for $u \in c, s$ has accepted, $\mathscr{C}$ returns the session key sk,
– *Test query* When the adversary $\mathscr{A}_2$ submits *Test* query and the query is not asked in the session, $\mathscr{C}$ sets $U = X$ as an instance of CDHP and aborts. Else, a fair coin b is flipped by $\mathscr{C}$. if $b = 1$. if not; a randomly string is returned to $\mathscr{A}_2$.

By the answers to the queries above, $\mathscr{C}$ is perfectly indistinguishable from our protocol unless the event E^{C2S} occurs. Note that the events $\exists i, Ocsk \wedge Test(S^i) \wedge \neg E^{C2S}$ and $\exists j, Ocsk \wedge Test(C^j)$ are equivalent, then we have

$$\Pr[Ocsk \wedge Test(C^j)] \geq \varepsilon/2 - \Pr_{C2S}$$

also, we have the following probability

$$\Pr\left[sk = H_2(P_{pub}, ID_c, \alpha, U, k_1, k_2)\Big|^{\alpha \in \mathbb{Z}^*_q}_{U, k_1, k_2 \leftarrow G_1}\right] \geq \frac{\varepsilon}{2} - Pr_{C2S}$$

Assume that if ε is non-negligible, then the $(\varepsilon/2) - \Pr_{C2S}$ is ε since the $\Pr_{C2S}$ is negligible by Theorem 1. Assume that the coin b in *Test* query is guessed with an advantage ε by $\mathscr{A}_1$. Given $(P, P_{pub} = bP = Y, U = aP = X)$ to $\mathscr{A}_1$. The instances $abP = k_1$, $abP = k_2$, $abP = k_3$, and $abP = k_4$ can be computed with an advantage ε by $\mathscr{A}_1$. The CDH assumption is resolved with a non-negligible advantage while $\mathscr{A}_1$ is used by $\mathscr{C}$. Hence, our protocol provides a secure key exchange. □

4.3 Server-to-client authentication

The following Theorem 3 demonstrates that the interaction from the server to the client can't be impersonated by $\mathscr{A}_i$ where $\{i = 1, 2\}$ under the CDH assumption.

Theorem 3 *We assume that the server-to-client authentication can be broken by adversary $\mathscr{A}_i$ where $\{i = 1, 2\}$ with an advantage ε. Then, the CDH assumption is solved by $\mathscr{C}$ with a non-negligible probability. At most q_S queries to the oracle Π^i_S of the server, q_C queries to the oracle Π^j_C of the client, and q_{H_i} queries on H_i oracle $\forall i \in \{1, 2, ..., 5\}$ are made by $\mathscr{C}$.*

Proof Let the algorithm be as presented previously in Theorem 2. Without the event E^{C2S} is happened, Our protocol is correctly indistinguishable from $\mathscr{C}$. The event that break the server-to-client authentication is denoted by E^{S2C}. The event E^{S2C} is accursed specifically, when $(ID_j, U = X)$ is sent and $(\alpha, Auth)$ is accepted by the client. Note that $(\alpha, Auth)$ is not processed by the right server. Due to this circumstance one of the following condition will be occurred:

1. $\mathscr{A}_i$ guessed the value $Auth$ with a probability less than $q_C/2^k$.
2. the value U occurred in another session with a probability $q_c/q \times (q_C - 1)$ less than q^2_C/q.
3. $\mathscr{A}_i$ asked $H_2(P_{pub}, ID_j, \alpha, U, k_3, k_4)$ with a probability $\Pr[(P_{pub}, ID_j, \alpha, U, k_3, k_4)|P_{pub} \in_R G_1, k_3 = bU, k_4 = bU]$

Then we have

$$\Pr[E^{S2C}|\neg E^{C2S}] \leq \Pr[P_{pub}, ID_j, \alpha, U, k_3, k_4)|P_{pub} \in_R G_1, k_3, k_4 = b.U] + \frac{q_C}{2^k} + \frac{q^2_C}{q}$$

The queries of $\mathscr{A}_i$ is answered by the algorithm $\mathscr{C}$ to prove Theorem 3. We assume that the random elements (P, $X = aP$, and $Y = bP$) $\in \mathbb{G}_1$ are received by $\mathscr{C}$ while $\forall a, b \in \mathbb{Z}^*_q$ are unknown values. The value of abP is derived by $\mathscr{C}$ with answering the $\mathscr{A}_i$'s oracle queries. The challenged identity ID_j is selected randomly by $\mathscr{C}$ while $P_{pub} = Y$, and $U = X$ are set. Given $(P, U, P_{pub}) = (P, X, Y)$ to the adversary $\mathscr{A}_i$. Then, $\mathscr{A}_i$ computes $abP = bX = k_3$ and $abP = bX = k_4$ with probability ε. After that, $\mathscr{C}$ uses $\mathscr{A}_i$ to compute abP. $\mathscr{C}$ solves the CDH assumption with the $\varepsilon' \geq \varepsilon - q_c/2^k - q^2_c/q$ is non-negligible. Hence, our protocol is secure against $\mathscr{A}_i$ and provides server-to-client authentication. □

Table 2 The comparison based on the computation and communication costs

Schemes	Computational cost		Communication cost
	Client	Server	
(Wu and Tseng 2010)	$4T_M + T_A + 3T_H$	$2T_e + 2T_M + T_A + 3T_H$	$\|ID\| + 2\|\mathbb{Z}_q^*\| + 2\|\mathbb{G}_1\|$
(He 2012)	$3T_M + 3T_H + T_i$	$T_e + 2T_M + 2T_A + 3T_H$	$\|ID\| + 2\|\mathbb{Z}_q^*\| + 3\|\mathbb{G}_1\|$
(Tsai and Lo 2015)	$2T_M + 3T_H + T_i$	$T_e + 5T_M + 2T_A + 5T_H$	$2\|\mathbb{Z}_q^*\| + 3\|\mathbb{G}_1\|$
Ours	$5T_M + T_A + 4T_H$	$2T_e + 4T_M + 2T_A + 6T_H$	$\|ID\| + 2\|\mathbb{Z}_q^*\| + 2\|\mathbb{G}_1\|$

Table 3 Comparison based on the security proprieties

	Wu and Tseng (2010)	He (2012)	Tsai and Lo (2015)	Our protocol
Mutual authentication	*Y*	*Y*	*Y*	*Y*
key exchange	*Y*	*Y*	*Y*	*Y*
Resistance to forgery attack	*Y*	*Y*	*Y*	*Y*
Perfect forward-secrecy	*N*	*N*	*Y*	*Y*
No key escrow problem	*N*	*N*	*N*	*Y*
provable secrecy	*Y*	*Y*	*Y*	*Y*

^{Y} refers to the scheme achieves this security property

^{N} refers to the scheme does not achieve this security property

5 Performance analysis

The computation cost, the communication cost and the security properties are assessed in this section. The proposed scheme is compared with (Wu and Tseng 2010), (He 2012), and (Tsai and Lo 2015) protocols. Due to convenience, the following notations are defined to evaluate the computational cost:

T_e: The time of $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
T_M: The time of a scalar multiplication operation of $\mathbb{G}_1$.
T_i: The time of performing a modular inversion operation.
T_A: The time of an addition operation of $\mathbb{G}_1$.
T_H: The performing time of a one-way hash function.

The computation and communication costs analysis of our protocol is represented in Table 2 compared with specific related protocols. Table 3 shows that the security properties that are provided by our protocol more than the other protocols.

We have implemented four schemes using java pairing-based cryptography Library (JPBC) (De Caro and Iovino 2011), on an Intel Core i 3 − 3110 CPU dual core 2.40 and GHz 2.40 GHz and machine with 4 GB RAM for the server and Honor EMUI 4.0.1 CPU Octa-core 1.5 GHz with 2.0 GB RAM for the client (mobile device). We have employed Type A pairings constructed from the curve $y^2 = x^3 + x$ over the field F_p for some prime $p = 3 \bmod 4$. Our experimental,

Table 4 Various security levels of our experiment (Bits)

Security Level	Size of *p*	Size of *q*
80-bit	1024	160
112-bit	2048	224
128-bit	3072	256

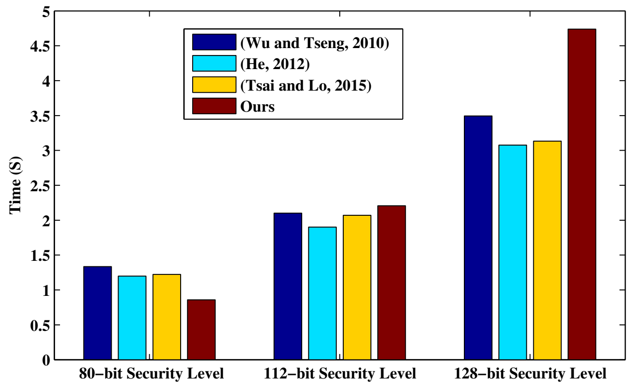

Fig. 3 The Client computational time

entailed 80-bit, 112-bit and 128-bit AES key sizes security levels (Daemen and Rijmen 2013) as shown in Table 4.

Fig 3 and Fig 4 respectively show the computational costs of the client side and the server-side for the four schemes at 80-bit, 112-bit and 128-bit security levels. The proposed

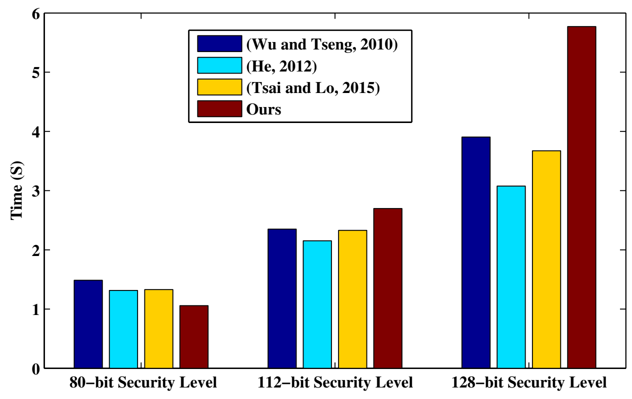

Fig. 4 The server computational time

scheme has higher computational cost than the other schemes because of using the CL-PKC.

Fig. 5 shows the communication cost, we assume that $|m| = \frac{160}{8}$ bytes , $|ID| = \frac{80}{8}$ bytes and $|\mathbb{G}_1| = 1024$ bits using an elliptic curve with $q = \frac{160}{8}$ bytes. By using the standard compression technique in (Shim et al. 2013), the size of $\mathbb{G}_1$ can be reduced to 65 bytes. A comparison of the computation cost for various schemes is as follows:

1. Communication cost in (Wu and Tseng 2010) is $|ID| + 2|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| = 10 + 2 \times 20 + 2 \times 65 = 180$ bytes.
2. Communication cost in (He 2012) is $|ID| + 2|\mathbb{Z}_q^*| + 3|\mathbb{G}_1| = 10 + 2 \times 20 + 3 \times 65 = 245$ bytes.
3. Communication cost in (Tsai and Lo 2015) is $2|\mathbb{Z}_q^*| + 3|\mathbb{G}_1| = 2 \times 20 + 3 \times 65 = 235$ bytes.
4. Communication cost in our scheme is $|ID| + 2|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| = 10 + 2 \times 20 + 2 \times 65 = 180$ bytes.

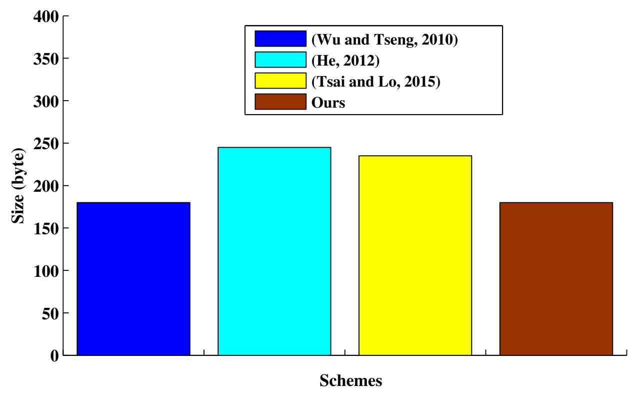

Fig. 5 The Communication Cost

According to Fig. 5 our scheme has a lower communication cost compared with that one obtained in He (2012) and Tsai and Lo (2015).

6 Application

When the information are communicated between the older person and the AAL server is sensitive, we need to consider the authentication and key exchange in this environment to prevent these information. The proposed scheme is suitable for the AAL applications that consider the authentication and key exchange. Here, we give application scenario that can be applicable to our scheme as shown in Fig. 6. In our application, the Controller (mobile device) is used as interface between the older person and the AAL server. The wearable sensors are connected to the controller sent data to the AAL server, which help the older person to care about his/her self. We assume that the server is the KGC in our protocol. The server runs the *Setup* algorithm to generate the public parameters and sends it to the client (controller). The older person sends his identity to the server. Since, the client identity is received by the AAL server, the *partial private key* D_{ID_c} is computed and is sent to the client. Next, the *Set private key* and the *public key* P_{ID_c} are performed by the client. Finally, the client and the server are cooperated to authenticate from each other by running the user authentication and key exchange algorithm.

7 Conclusion

A new certificateless user authenticated key exchange protocol is introduced to preform the key escrow problem of the identity-based user authentication schemes. Our protocol provides a mutual authentication and under the CDH assumption its security is demonstrated in the random oracle model. The proposed protocol needs 4.738 seconds in the client side and 5.772 seconds in the server side, when the security level is 128 bit. Furthermore, the communication

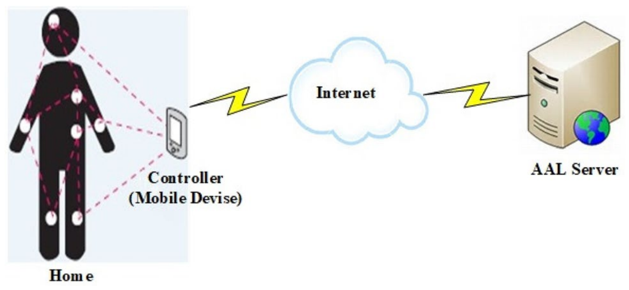

Fig. 6 The application scenario

cost of our protocol is 180 bytes. Indeed, Our protocol can be applied in the assisted living systems environment.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Grant No 61272525), the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2016J081) and the Laboratory for Internet of Things and Mobile Internet Technology of Jiangsu Province (Grant No JSWLW-2017-006).

References

Al-Riyami SS, Paterson KG (2003) Certificateless public key cryptography. In: International Conference on the Theory and Application of Cryptology and Information Security, Springer, pp 452–473

Bellare M, Rogaway P (1993) Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and communications security, ACM, pp 62–73

Boneh D, Franklin M (2001) Identity-based encryption from the weil pairing. In: Annual International Cryptology Conference, Springer, pp 213–229

Boneh D, Lynn B, Shacham H (2004) Short signatures from the weil pairing. J Cryptol 17(4):297–319

Canetti R, Krawczyk H (2001) Analysis of key-exchange protocols and their use for building secure channels. Springer, New York, pp 453–474

Choon JC, Cheon JH (2003) An identity-based signature from gap diffie-hellman groups. In: International Workshop on Public Key Cryptography, Springer, pp 18–30

Daemen J, Rijmen V (2013) The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media, Berlin

Das ML, Saxena A, Gulati VP, Phatak DB (2006) A novel remote user authentication scheme using bilinear pairings. Comput Secur 25(3):184–189

De Caro A, Iovino V (2011) jpbc: Java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28–July 1, pp 850–855

Fang G, Huang G (2006) Improvement of recentlyproposed remote client authentication protocols. http://eprint.iacr.org/2006/200

Hamida EB, Javed MA, Znaidi W (2017) Adaptive security provisioning for vehicular safety applications. Int J Space-Based Situat Comput 7(1):16–31

Hassan A, Eltayieb N, Elhabob R, Li F (2016) A provably secure certificateless user authentication protocol for mobile client-server environment. In: International Conference on Emerging Internetworking. Data & Web Technologies, Springer, pp 592–602

He D (2012) An efficient remote user authentication and key agreement protocol for mobile client-server environment from pairings. Ad Hoc Netw 10(6):1009–1016

He D, Huang B, Chen J (2013) New certificateless short signature scheme. IET Inform Secur 7(2):113–117

Hou Mb, Xu Ql (2009) Secure certificateless-based authenticated key agreement protocol in the client-server setting. In: IT in Medicine & Education, 2009. ITIME'09. IEEE International Symposium on IEEE, vol 1, pp 960–965

Jaballah WB, Mosbah M, Youssef H, Zemmari A (2015) Lightweight secure group communications for resource constrained devices. Int J Space-Based Situat Comput 5(4):187–200

Jakobsson M, Pointcheval D (2001) Mutual authentication for low-power mobile devices. In: International Conference on Financial Cryptography, Springer, pp 178–195

LaMacchia B, Lauter K, Mityagin A (2007) Stronger security of authenticated key exchange. Springer, Berlin, pp 1–16

Nam J, Lee J, Kim S, Won D (2005) Ddh-based group key agreement in a mobile environment. J Syst Softw 78(1):73–83

Pointcheval D, Stern J (1996) Security proofs for signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp 387–398

Pointcheval D, Stern J (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13(3):361–396

Ren Y, Wang H, Du J, Ma L (2016) Code-based authentication with designated verifier. Int J Grid Util Comput 7(1):61–67

Sabzevar AP, Sousa JP (2011) Authentication, authorisation and auditing for ubiquitous computing: a survey and vision. Int J Space-Based Situat Comput 1(1):59–67

Shamir A (1984) Identity-based cryptosystems and signature schemes. In: Workshop on the Theory and Application of Cryptographic Techniques, Springer, pp 47–53

Shen H, Gao C, He D, Wu L (2015) New biometrics-based authentication scheme for multi-server environment in critical systems. Journal of Ambient Intelligence and Humanized Computing

Shen J, Chang S, Shen J, Liu Q, Sun X (2018) A lightweight multi-layer authentication protocol for wireless body area networks. Future Gener Comput Syst 78:956–963

Shim KA, Lee YR, Park CM (2013) Eibas: An efficient identity-based broadcast authentication scheme in wireless sensor networks. Ad Hoc Netw 11(1):182–189

Sun X, Jiang Z, Zhou M, Wang Y (2014) Versatile identity-based signatures for authentication in multi-user settings. Int J Grid Util Comput 5(3):156–164

Tsai JL, Lo NW (2015) Provably secure and efficient anonymous id-based authentication protocol for mobile devices using bilinear pairings. Wirel Pers Commun 83(2):1273–1286

Tseng YM (2006) Gprs/umts-aided authentication protocol for wireless lans. IEE Proc Commun 153(6):810–817

Tseng YM (2007) A secure authenticated group key agreement protocol for resource-limited mobile devices. Comput J 50(1):41–52

Tseng YM, Wu TY, Wu JD (2008) A pairing-based user authentication scheme for wireless clients with smart cards. Informatica 19(2):285–302

Wang XA, Weng J, Yang X, Yang Y (2011) Cryptanalysis of an identity based broadcast encryption scheme without random oracles. Inform Proc Lett 111(10):461–464

Wang XA, Ma J, Yang X (2015) A new proxy re-encryption scheme for protecting critical information systems. J Ambient Intell Human Comput 6(6):699–711. https://doi.org/10.1007/s12652-015-0261-3

Wang XA, Ma J, Xhafa F, Zhang M, Luo X (2017) Cost-effective secure e-health cloud system using identity based cryptographic techniques. Future Gener Comput Syst 67:242–254

Wang Y, Ma J, Lu X, Lu D, Zhang L (2016) Efficiency optimisation signature scheme for time-critical multicast data origin authentication. Int J Grid Util Comput 7(1):1–11

Wong DS, Chan AH (2001) Efficient and mutually authenticated key exchange for low power computing devices. In: International Conference on the Theory and Application of Cryptology and Information Security, Springer, pp 272–289

Wu F, Xu L, Kumari S, Li X (2017) A privacy-preserving and provable user authentication scheme for wireless sensor networks based on internet of things security. Journal of Ambient Intelligence and Humanized Computing

Wu L, Zhang Y, Xie Y, Alelaiw A, Shen J (2016) An efficient and secure identity-based authentication and key agreement protocol with user anonymity for mobile devices. Wireless Personal Communications pp 1–17

Wu TY, Tseng YM (2010) An efficient user authentication and key exchange protocol for mobile client-server environment. Comput Netw 54(9):1520–1530

Yoon E, Yoo K (2010) A new efficient id-based user authentication and key exchange protocol for mobile client-server environment. In: Wireless Information Technology and Systems (ICWITS), 2010 IEEE International Conference on IEEE, pp 1–4

Zhang Y, Chen J, Li H, Cao J, Lai C (2014) Group-based authentication and key agreement for machine-type communication. Int J Grid Util Comput 5(2):87–95