Wireless Networks (2020) 26:913–924
https://doi.org/10.1007/s11276-018-1839-4

A heterogeneous user authentication and key establishment for mobile client–server environment

Fagen Li[1] · Jiye Wang[2] · Yuyang Zhou[1] · Chunhua Jin[3] · SK Hafizul Islam[4]

Published online: 24 September 2018

Abstract
In a mobile client–server environment, a low-power mobile device wants to access a strong server to get some kind of services. User authentication and key establishment are two basic security requirements for this environment. Without the user authentication, an unauthorized user can access the server and gets the services. Without the key establishment, the communication between the user and the server will be disclosed. Recently, some user authentication and key establishment protocols were designed. However, all of them are homogeneous since the client and the server belong to the same cryptosystem. That is, both the client and the server belong to public key infrastructure or identity-based cryptosystem or self-certified cryptosystem. Such design does not comply with the characteristic of mobile client–server application. In this paper, we design a heterogeneous user authentication and key establishment protocol using a signcryption scheme. In this protocol, the client uses identity-based cryptosystem and the server uses the public key infrastructure. As compared with existing works, our protocol has the lowest cost in computation and communication.

Keywords Security · Authentication · Key establishment · Mobile device · Signcryption

1 Introduction

With the rapid evolvement of wireless communication technology, more and more mobile devices (i.e., mobile telephone and personal digital assistant) are used to access the Internet and to carry out the electronic commerce. A typical network architecture is the Mobile Client–Server (MCS) environment [1]. In this environment, a user (client) with low-power mobile device accesses a strong server to get some kind of services. The MCS architecture has become one of the basic models of mobile network computing. Many types of mobile applications have used the MCS architecture, such as mobile e-mails, mobile web access and mobile social networks [2, 3]. For example, both Facebook and Twitter use the MCS architecture [4]. The security of this architecture plays a pivotal role for users applications [5]. To prevent unauthorized users from accessing the server's service, we need to authenticate the users. In addition, the transmitted messages between the client and the server may be sensitive. Therefore, a session key is required to be established between them. Hence the user authentication and the key establishment are two basic security requirements for this environment. However, it is not easy to develop such protocols since the computational power of mobile devices and the communication bandwidth are limited. Therefore, we should try our best to reduce the computational task of mobile devices and to shorten exchanged messages.

According to the public key authentication method, the public key cryptosystem can be divided into three types: Public Key Infrastructure (PKI) [6], Identity-Based Cryptosystem (IBC) [7] and Self-Certified (certificateless) Cryptosystem (SCC) [8]. In the PKI, each user sets a

✉ Fagen Li
fagenli@uestc.edu.cn

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

2 State Grid Corporation of China, Beijing 100031, China

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

4 Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, Kalyani, West Bengal 741235, India

private key and a matching public key. A Certificate Authority (CA) issues a digital certificate to authenticate this public key because this certificate binds the user identity and the public key. The PKI may not be a good candidate for the mobile communication since the certificates management (i.e., generation, distribution, use, query, storage and revocation) is too heavy for the low power mobile devices. In the IBC, a user selects a binary string as its public key. A trusted party called Private Key Generator (PKG) computes the corresponding private key using the binary string and a master key. The merit of the IBC is that we can explicitly check the validity of a public key without an attached certificate. So the IBC is very lightweight and is a good candidate for wireless communication. However, the flaw of IBC is that the PKG holds anyone's private key [7]. Therefore, the IBC only fits the small-scale networks and does not fit the large-scale networks, for instance the Internet. In the SCC, the PKG first computes a partial private key using the identity information and a master key. Then the user sets its full private key by uniting the partial private key and a chosen secret value. The SCC eliminates both certificates and key escrow problem.

Yang and Chang [9] gave an Identity-Based Authentication Protocol (IBAP) with key agreement using Elliptic Curve Cryptography (ECC) technique. The ECC can use a smaller key size to achieve a comparable security level to the other cryptographic scheme such as RSA [10]. However, Yoon and Yoo [11] showed that [9] can not satisfy the authentication property by launching an impersonation attack. Chou et al. [12] designed two IBAPs with key agreement. The first protocol sets up a session key between the client and server and the second protocol sets up a session key between two users. Farash and Attari [13] (hereafter called FA) showed that [12] is not secure under the impersonation attack and gave an improvement. Shi et al. [14] pointed out a weakness in the registration phase of [12, 13]. In this weakness, an authorized user is able to impersonate the server to generate a correct private key for the other users. Qi and Chen [15] solved the clock synchronization issue in the authentication key exchange by applying the challenged-response handshake method.

Wu and Tseng [16] (hereafter called WT) designed an Identity-Based User Authentication Protocol (IBUAP) with key exchange for MCS environment using bilinear pairings. The WT reduces the computational cost of the client by transferring the computation to the powerful server. He [17] (hereafter called He) further improved the performance of WT. In addition, He et al. [18] (hereafter called HCH) gave an IBUAP with key agreement for MCS environment without using *MapToPoint* function. The *MapToPoint* is a special hash function that maps an identity into a point on an elliptic curve and is slower than a general hash function. Unfortunately, Wang and Ma [19] pointed out that HCH is vulnerable to the reflection attack and parallel session attack. Hassan et al. [20] gave a SCC-based user authentication and key exchange protocol for the MCS environment.

Chuang and Tseng [21] (hereafter called CT) gave an IBUAP for mobile multi-server environment. The multi-server means that there are multiple servers. Many organizations, such as a university, a company and a hospital may have multiple servers that include a file server, an e-mail server, a web server, etc. The multi-server environment introduces a Registration Center (RC) that is in charge of the registration of clients. If a client has registered at the RC, it can access the multiple servers. Such an approach avoids repeated registration at every server. Liao and Hsiao [22] (hereafter called LH) designed a SCC-based authentication protocol for mobile multi-server environment. However, Hsieh and Leu [23] (hereafter called HL) pointed out that LH is not secure under the trace attack and presented an improvement.

From the above discussion, we know that all user authentication and key establishment protocols are homogeneous since the client and the server belong to the same cryptosystem. That is, both the client and the server belong to PKI or IBC or SCC. Such design does not conform to the characteristic of MCS environment. The characteristic of MCS is that the client is weak and the server is strong. Therefore, a good method is that the client uses the IBC and the server uses the PKI. Such design requires our protocol is heterogeneous. In this paper, we design a heterogeneous user authentication and key establishment protocol using a signcryption scheme. In this protocol, the client belongs to the IBC and the server belongs to the PKI. As compared with previous related protocols, our protocol has the least computational cost and communication overhead.

2 Preliminaries

In this section, we give the network model and bilinear pairings.

2.1 Network model

Figure 1 gives the overview of the MCS model. The model consists of three kinds of entities, a RC, a server and some clients. The clients want to access the server to get some kinds of services by WiFi, 3G or 4G wireless communication technique. The RC is in charge of the registration for clients and the server. That is, the RC acts as the role of the PKG in the IBC and the CA in the PKI. The clients are low-power and the server is powerful. Of course, the low-power devices are a relative and evolving process. For

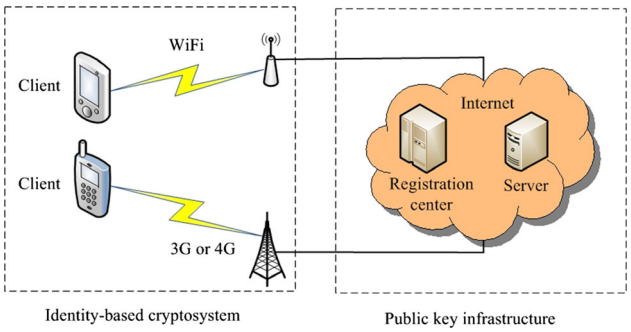


Fig. 1 Network model

example, mobile phones are low-power devices 10 years ago, but now they are getting stronger and stronger. Sensors, wearable devices [24], smart meters are low-power now, but they may become powerful in the future. We assume that the RC is believable and can never be compromised. When a client hopes to access the server, it will send an authentication message to the server and the server will verify the client has been authorized or not. If the client has been authorized, a session key will be established between them. Otherwise, the server will send a rejection message to the client. If the RC is compromised, the adversary can learn the private keys of clients and forge the certificate of the server. So it is a basic assumption that the RC can not be attacked. In addition, the key escrow problem exists since the RC learns the private keys of clients. To mitigate the RC compromise and key escrow problem, we can use distributed RC [7].

There are two main security goals for the MCS environment. The first goal is the authentication that assures that only authorized clients can access the server. The second goal is the key establishment that sets up a session key between a client and the server.

2.2 Bilinear pairings

Let G_1 be a additive cyclic group (the generator is P) and G_2 be a multiplicative group. Both groups have the same prime order p. A bilinear pairing is a special map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ that satisfies the following properties [7]:

1. *Bilinearity* $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ holds for all elements $P, Q \in G_1$ and $a, b \in \mathbb{Z}_p^*$.
2. *Non-degeneracy* $\hat{e}(P, Q) \neq 1_{G_2}$ (1_{G_2} is the identity element of G_2) holds for some $P, Q \in G_1$.
3. *Computability* $\hat{e}(P, Q)$ can be efficiently implemented for all P,$Q \in G_1$.

The modified Weil pairing and Tate pairing can supply such maps [7].

3 Our protocol

In this section, we design a heterogeneous user authentication and key establishment protocol using Li et al.'s signcryption (hereafter called LZT) [25]. Our protocol has four phases: initialization, registration, authentication and key establishment, and revocation. The main notations in our protocol are summarized in Table 1.

3.1 Initialization phase

Given a security parameter sp, RC selects $(G_1, G_2, p, \hat{e}, P)$ defined in Sect. 2.2 and three secure hash functions $H_1 : \{0,1\}^{\ell_3+\ell_4} \rightarrow \mathbb{Z}_p^*$, $H_2 : G_2 \rightarrow \{0,1\}^{\ell_1+\ell_2+\ell_3+\ell_4}$ and $H_3 : \{0,1\}^{\ell_1+\ell_2+\ell_3+\ell_4+\ell_5} \rightarrow \mathbb{Z}_p^*$. Then the RC selects a master key $s \in \mathbb{Z}_p^*$ and sets the public key $P_{pub} = sP$ and $g = \hat{e}(P, P)$. Finally, the RC releases the system parameters $\{p, G_1, G_2, \hat{e}, P, g, P_{pub}, H_1, H_2, H_3\}$ and retains s secret.

3.2 Registration phase

The clients and the server should register at the RC. For the registration of a client, the client first submits its identity ID to the RC. Then the RC sets an expiration date ED, computes the private key

$$S_{ID} = \frac{1}{H_1(ID||ED) + s} P$$

and sends (S_{ID}, ED) to the client in a secure way. Similar to [16], we can adopt off-line approach or on-line Transport Layer Security (TLS) approach to deliver the private key. The registration process of the client is summarized in Fig. 2.

For the registration of the server, the server first chooses a random w from $\mathbb{Z}_p^*$ and sets $SK = (1/w)P$ as its private key and $PK = wP$ as its public key. Then the server submits its identity and public key PK to the RC. The RC generates a digital certificate $Cert$ for the server by using a digital signature algorithm (here we recommend using Elliptic Curve Digital Signature Algorithm (ECDSA) [26]). Finally, the RC sends the digital certificate to the server. Note that the digital certificate can be delivered in an open channel. The registration process of the server is summarized in Fig. 3.

3.3 Authentication and key establishment phase

When a client with identity ID wants to access the server to get some kind of services, the client first randomly chooses a session key $k \in \{0,1\}^{\ell_1}$ and $x \in \mathbb{Z}_p^*$. Then the client computes $r = g^x$, $c = (k||TS||ID||ED) \oplus H_2(r)$, $h = H_3(k||TS||ID||ED||r)$, $S = (x + h)S_{ID}$ and $T = xPK$. Here TS is a timestamp

Table 1 Notations

Notations	Descriptions	Notations	Descriptions
sp	A security parameter	TS	A timestamp
G_1	A cyclic additive group	ID	The identity of a client
G_2	A cyclic multiplicative group	ED	An expiration date
P	A generator of group G_1	S_{ID}	The private key of a client with identity ID
$\hat{e}$	A bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$	SK	The private key of the server
p	The order of group G_1 and G_2	PK	The public key of the server
s	A master secret key of the PKG	ℓ_1	The number of bits of a session key k
P_{pub}	A master public key of the PKG	ℓ_2	The number of bits of a timestamp TS
g	A element in group G_2, where $g = \hat{e}(P, P)$	ℓ_3	The number of bits of an identity ID
$H_i()$	A one way hash function ($i = 1, 2, 3$)	ℓ_4	The number of bits of an expiration date ED
MAC	A message authentication code	ℓ_5	The number of bits of an element in G_2
k	A session key	$\|\|$	Concatenation

Fig. 2 Registration process of a client

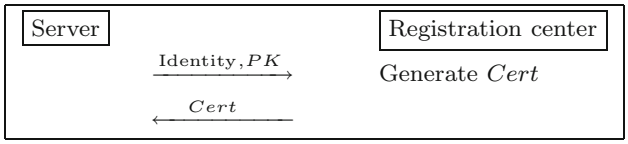


Fig. 3 Registration process of the server

to resist the replay attack. Note that we encrypt the client's identity to achieve the anonymity. The client sends (c, S, T) to the server. After receiving the (c, S, T), the server first computes $r = \hat{e}(T, SK)$ and $k||TS||ID||ED = c \oplus H_2(r)$. Then the server computes $h = H_3(k||TS||ID||ED||r)$ and checks if the

$$r = \hat{e}(S, H_1(ID||ED)P + P_{pub})g^{-h}$$

holds. If the above equation holds, the server accepts the client's access request. A session key k has been established between the client and the server. The k is only known by them, which guarantees the confidentiality for future communication. Otherwise, the server rejects the client's access request. If we hope to achieve the key confirmation property, the server computes $tag = MAC_k(TS)$ and sends to the client. When receiving the tag, the client computes $tag' = MAC_k(TS)$ and checks if $tag' = tag$. If yes, the client believes that the server has known the session key k. We summarize the above process in Fig. 4.

We have two methods to establish a session key: key exchange and key transport. The key exchange protocol allows the both parties to jointly decide the session key. The key transport protocol allows one party to decide the session key and to transfer to the other party. Obviously, our protocol is a key transport protocol. However, our protocol can be easily modified into a key exchange type. The server needs to choose a random k^* from $\{0,1\}^{\ell_1}$, computes $tag = MAC_{k \oplus k^*}(TS)$ and sends (tag, k^*) to the client. The client computes $tag' = MAC_{k \oplus k^*}(TS)$ and checks if $tag' = tag$. In this case, the session key is $k \oplus k^*$. We summarize the modified process in the Fig. 5.

3.4 Revocation phase

We can automatically revoke the registration right by attaching an expiration date ED. For instance, if we set the ED as "2018-12-31", the client only can access the server before December 31, 2018. If we must revoke the privilege before this date due to some irresistible reasons, the RC can send the identity of this client to the server. The server stores a list of revoked identities so that it can check if the privilege has expired.

4 Analysis of the protocol

In this section, we analyze the correctness, security and performance of the designed protocol.

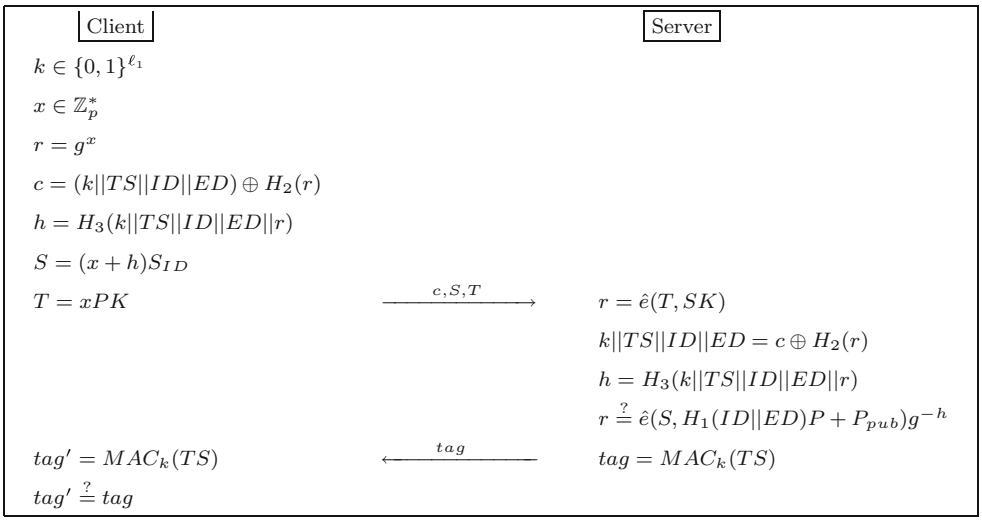


Fig. 4 Authentication and key establishment process

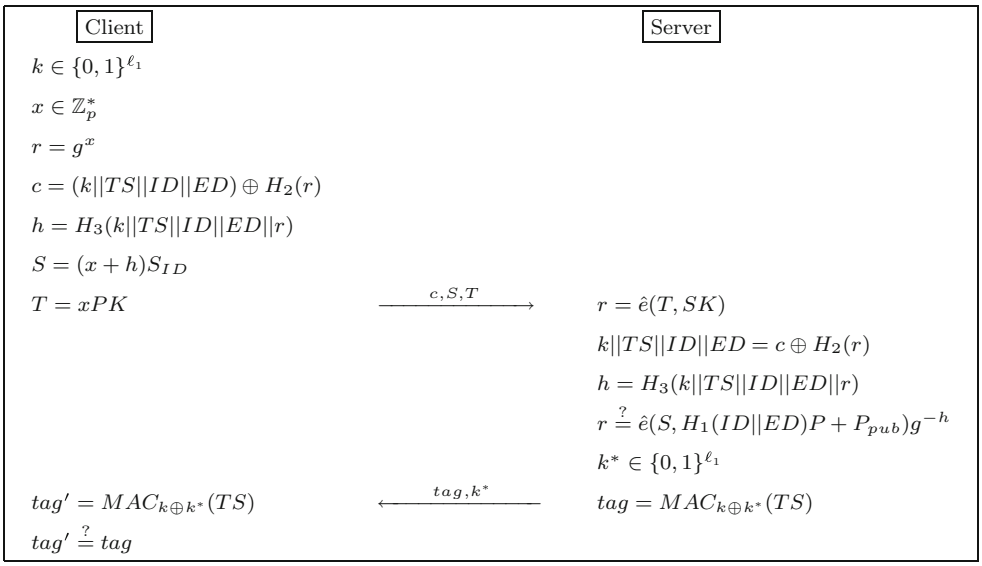


Fig. 5 Modified authentication and key establishment process

4.1 Correctness

We show the correctness of our protocol. That is, if the server receives the message (c, S, T), it can authenticate the client correctly and establish the session key.

Since $T = xPK$, $PK = wP$ and $SK = \frac{1}{w}P$, we have

$$\hat{e}(T, SK) = \hat{e}\left(xPK, \frac{1}{w}P\right) = \hat{e}\left(xwP, \frac{1}{w}P\right) = \hat{e}(P,P)^x = g^x = r$$

In addition, because $S = (x+h)S_{ID}$ and $S_{ID} = \frac{1}{H_1(ID||ED)+s}P$, we have

$$\begin{aligned}
&\hat{e}\big(S, H_1(ID||ED)P + P_{pub}\big)g^{-h} \\
&= \hat{e}\left(\frac{x+h}{H_1(ID||ED)+s}P, (H_1(ID||ED)+s)P\right)g^{-h} \\
&= g^{x+h}g^{-h} \\
&= g^x \\
&= r
\end{aligned}$$

4.2 Security

Our protocol is based on LZT signcryption that satisfies confidentiality under the bilinear Diffie–Hellman inversion problem and unforgeability under the q-strong Diffie–Hellman problem [25]. Now we prove our protocol in the authenticated key agreement model [27–30].

The model has a set of participants that are modeled by some oracles. We use the notation $\prod_{i,j}^{\tau}$ to denote that a participant i believes that it is carrying out the τ-th run of the protocol with a participant j. The model also contains an adversary $\mathcal{A}$ that can access all message flows in this system. $\mathcal{A}$ even can relay, modify and delete messages. All the oracles can communicate with each other by $\mathcal{A}$. In order to answer $\mathcal{A}$'s queries, oracles keep transcripts that record all messages they have sent or received. Note that $\mathcal{A}$ is neither a participant nor the PKG (CA). If $\mathcal{A}$ does not modify the messages and just transfer the messages, we call the adversary passive. At any time, the adversary is permitted to ask the following queries.

Create $\mathcal{A}$ chooses an identity *ID* and sets up a new participant (oracle). The participant's public key is the identity and the corresponding private key is gained from the PKG.

Send $\mathcal{A}$ sends a chosen message to an oracle i, $\prod_{i,j}^{\tau}$, in which case the participant i assumes that the message cames from the participant j. In addition, $\mathcal{A}$ can instruct the oracle j to start a new run of this protocol with i by sending an empty message λ. If the first message that an oracle receives is an empty message λ, the oracle is called an initiator oracle. Otherwise, the oracle is called a responder oracle.

Reveal $\mathcal{A}$ is permitted to ask an oracle to obtain the session key (if any) it currently holds.

Corrupt $\mathcal{A}$ is permitted to ask an oracle to gain its long term private key.

An oracle has one of the following several states:

Accepted After receiving the properly messages, the oracle makes up its mind to accept a session key.

Rejected If an oracle has not set up a session key, it decides to reject and to abort this run of the protocol.

* If an oracle has not made up its mind whether to accept or reject, the state of this oracle is *.

Opened If an oracle has replied a reveal query, the state is opened.

Corrupted If an oracle has replied a corrupt query, the state is corrupted.

At some point, $\mathcal{A}$ can choose one of these oracles, such as $\prod_{i,j}^{\tau}$, to ask a *Test* query. The chosen oracle must be accepted, be unopened and neither i nor j has been corrupted. In addition, there must be no opened oracle $\prod_{j,i}^{\tau}$ with which it has had a matching conversation. To reply this query, the oracle flips a fair coin $\beta \in \{0,1\}$. If $\beta = 0$, then the oracle replies the held session key. Otherwise, the oracle replies a random key from the given key space.

To attack a protocol, $\mathcal{A}$ runs an experiment with the set of oracles generated by a challenger $\mathcal{C}$. In this experiment, $\mathcal{A}$ can make a polynomially bounded number of *Create*, *Send*, *Reveal*, *Corrupt* queries and one *Test* query. In the end, $\mathcal{A}$ gives a bit β' as its guess for β.

We use $\mathrm{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - 1/2|$ to denote $\mathcal{A}$'s advantage, where $\Pr[\beta' = \beta]$ is the probability that $\beta' = \beta$.

Definition 1 If an authenticated key agreement protocol fulfils the following three terms, we say that this protocol is secure [28].

1. In the presence of a passive attacker on $\prod_{i,j}^{\tau}$ and $\prod_{j,i}^{\upsilon}$, both oracles always accept possessing the same session key. In addition, this key is uniformly distributed in the given key space.
2. For each adversary, if uncorrupted oracles $\prod_{i,j}^{\tau}$ and $\prod_{j,i}^{\upsilon}$ have matching conversations, both oracles accept and possess the same session key.
3. For each adversary, $\mathrm{Adv}(\mathcal{A})$ is negligible.

Now we give the security result of our protocol by the following Theorem 1.

Theorem 1 *Our protocol is a secure key establishment protocol.*

Proof In this proof, we show that our protocol satisfies the three terms in Definition 1.

First, our protocol uses LZT signcryption to transfer a session key. Because LZT satisfies the consistency, the server must obtain the same session key as the client. In addition, the session key is randomly selected from the key space $\{0,1\}^{\ell_1}$. Therefore, the first condition holds.

Second, because the consistency constraint of LZT, uncorrupted oracles $\prod_{i,j}^{\tau}$ and $\prod_{j,i}^{\upsilon}$ that have matching conversations must accept and possess the same session key. Therefore, the second condition holds.

Third, we show the third condition holds by contradiction. If there exists an attacker $\mathcal{A}$ that has a non-negligible advantage ϵ against the security of Definition 1 in our protocol, then we can either construct an algorithm $\mathcal{C}_1$ to break the unforgeability of LZT with a non-negligible advantage $\epsilon_1' \geq (\epsilon - \epsilon_2' mn\phi)/mn$ or construct an algorithm $\mathcal{C}_2$ to break the confidentiality of LZT with a non-negligible advantage $\epsilon_2' \geq (\epsilon - \epsilon_1' mn)/mn\phi$. Our proof method comes from [31].

We finish this proof by two parts. In the first part, if an event E (explained later) occurs, $\mathcal{C}_1$ is constructed. In the second part, if the event E does not occur, $\mathcal{C}_2$ is constructed. Let $\{ID_1, ID_2, \ldots, ID_m\}$ be set of m clients and $\{S_1, S_2, \ldots, S_n\}$ be set of n servers. We assume that each client is activated at most ϕ times and each server is activated at most μ times by $\mathcal{A}$.

In the first part, if there exists an attacker $\mathcal{A}$ that can distinguish a random value from a session key in a time t_1,

we can construct an algorithm $\mathcal{C}_1$ that can break the confidentiality of LZT in a time

$$t_1' \leq t_1 + n\mu t_s + m\phi t_u$$

where t_s and t_u are the response times for the signcryption and unsigncryption queries, respectively. $\mathcal{C}_1$ acts as $\mathcal{A}$'s challenger and runs $\mathcal{A}$ as a subroutine. The goal of $\mathcal{C}_1$ is to output a valid signcryption ciphertext (c^*, S^*, T^*) between a client ID_A and a server S_B. The input of $\mathcal{C}_1$ includes system parameters and key pairs of n servers in this protocol except the S_B's private key. $\mathcal{C}_1$ wins its game only if the target session selected by $\mathcal{A}$ is a session between ID_A and S_B. $\mathcal{A}$ can ask the following queries and $\mathcal{C}_1$ keeps a state list L_s that records internal state information.

Create $\mathcal{A}$ chooses an identity ID_i and asks a *Create* query. In this case, $\mathcal{C}_1$ asks its key extraction oracle with an input ID_i and obtains the corresponding private key S_{ID_i}. Note that if $ID_i = ID_A$, $\mathcal{C}_1$ will fail since its key extraction oracle can not output a correct private key.

Corrupt $\mathcal{A}$ asks a *Corrupt* query on ID_i or S_j. If $ID_i \neq ID_A$, $\mathcal{C}_1$ can answer it because $\mathcal{C}_1$ knows the private key S_{ID_i}. If $S_j \neq S_B$, $\mathcal{C}_1$ also can answer it because $\mathcal{C}_1$ knows the private key SK_j. Otherwise, $\mathcal{C}_1$ fails and stops. $\mathcal{C}_1$ should set the states of corrupted oracles *Corrupted*.

Send From *Create* and *Corrupt*, we learn that $\mathcal{C}_1$ knows private keys of all parties except ID_A and S_B. Therefore, if $\mathcal{A}$ asks a *Send* query that does not relate to ID_A and S_B, $\mathcal{C}_1$ answers it in the straightforward way since $\mathcal{C}_1$ knows their private keys. For a *Send* query that needs the private keys of ID_A and S_B, $\mathcal{C}_1$ answers it using its own oracles. We explain the method below. When $\mathcal{A}$ asks a *Send* $(\prod_{A,j}^{\tau})$ query with input an empty message λ, $\mathcal{C}_1$ randomly chooses a session key k and asks its signcryption oracle with input a message $k||TS||ID_A||ED$ and a public key PK_j to obtain a ciphertext $\sigma = (c, S, T)$. $\mathcal{C}_1$ returns σ to $\mathcal{A}$ and inserts $(\tau, A, j, k||TS||ID_A||ED, \sigma, *)$ into the state list L_s. When $\mathcal{A}$ asks a *Send* $(\prod_{B,i}^{\tau})$ query with input a ciphertext σ, $\mathcal{C}_1$ asks its unsigncryption oracle with input the ciphertext σ and identity ID_i. If it obtains a $k||TS||ID||ED$ from its unsigncryption oracle, $\mathcal{C}_1$ marks the oracle $\prod_{B,i}^{\tau}$ as accepted, inserts

$$(\tau, B, i, k||TS||ID||ED, \sigma, Accepted)$$

in the list L_s and returns $tag = MAC_k(TS)$ to $\mathcal{A}$. If the unsigncryption oracle outputs a failure symbol $\perp$, this session is not accepted. $\mathcal{C}_1$ returns *Rejected* to $\mathcal{A}$ and inserts $(\tau, B, i, \perp, \sigma, Rejected)$ into the list L_s. When $\mathcal{A}$ asks a *Send* $(\prod_{A,j}^{\tau})$ query with input an message *tag*, $\mathcal{C}_1$ checks if there exists an entry

$$(\tau, A, j, k||TS||ID_A||ED, \sigma, *)$$

in the list L_s. If none is found, $\mathcal{C}_1$ marks the oracle $\prod_{A,j}^{\tau}$ as rejected and inserts $(\tau, A, j, \perp, \perp, Rejected)$ in the L_s. Otherwise, $\mathcal{C}_1$ further computes $tag' = MAC_k(TS)$ and checks if $tag' = tag$. If no, $\mathcal{C}_1$ marks the oracle $\prod_{A,j}^{\tau}$ as rejected and updates $(\tau, A, j, k||TS||ID_A||ED, \sigma, *)$ into

$$(\tau, A, j, k||TS||ID_A||ED, \sigma, Rejected).$$

If yes, $\mathcal{C}_1$ marks the oracle $\prod_{A,j}^{\tau}$ as accepted and updates

$$(\tau, A, j, k||TS||ID_A||ED, \sigma, *)$$

into $(\tau, A, j, k||TS||ID_A||ED, \sigma, Accepted)$.

Reveal When $\mathcal{A}$ asks a *Reveal* query on the oracle $\prod_{i,j}^{\tau}$, $\mathcal{C}_1$ checks if there exists an entry for (τ, i, j) in the list L_s. Because a *Reveal* query can be asked only if a session has been accepted, the list L_s must contain an entry for (τ, i, j). Otherwise, this query is not valid. When $\mathcal{C}_1$ finds this entry, $\mathcal{C}_1$ returns the corresponding session key k and updates

$$(\tau, i, j, k||TS||ID_i||ED, \sigma, Accepted)$$

into $(\tau, i, j, k||TS||ID_i||ED, \sigma, Opened)$.

In the end of this simulation, $\mathcal{C}_1$ checks if there is an entry $(\tau, B, A, k||TS||ID_A||ED, \sigma, Accepted)$ such that no entry contains (τ, A, B). If yes, $\mathcal{C}_1$ outputs its forgery $\sigma^* = \sigma$. Let E be the event that $\mathcal{A}$ asked a *Send* $(\prod_{B,A}^{\tau})$ query with input an ciphertext σ^* such that σ^* is a valid ciphertext under ID_A and PK_B and σ^* is not a response of *Send* $(\prod_{A,B}^{\tau})$ query with input an empty message λ. If the event E occurs, $\mathcal{C}_1$ must find a wanted entry in the list L_s and successfully output a forgery.

If $\mathcal{A}$ stops its run without selecting a test session between ID_A and S_B or the event E does not occurs, $\mathcal{C}_1$ will fail. The probability that $\mathcal{A}$ selects a test session with ID_A as initiator and S_B as responder is $1 / mn$. Therefore, the advantage of $\mathcal{C}_1$ is

$$\epsilon_1' \geq \frac{\Pr[E]}{mn}$$

To answer the *Send* $(\prod_{A,j}^{\tau})$ queries, $\mathcal{C}_1$ needs to ask its signcryption oracle. The maximum number of such queries is $n\mu$. In addition, To answer the *Send* $(\prod_{B,i}^{\tau})$ queries, $\mathcal{C}_1$ needs to ask its unsigncryption oracle. The maximum number of such queries is $m\phi$. Therefore, $\mathcal{C}_1$ can forge a ciphertext in a time

$$t_1' \leq t_1 + n\mu t_s + m\phi t_u$$

In the second part, we assume that there exists an attacker $\mathcal{A}$ that can distinguish a random value from a session key in a time t_2 when the event E does not occur. By using $\mathcal{A}$ as a subroutine, we can construct an algorithm $\mathcal{C}_2$ in a time

$t_2' \leq t_2 + n\mu t_s + m\phi t_u$

The input of $\mathcal{C}_2$ includes system parameters, master secret key s, and key pairs of n servers in this protocol except the private key of S_B. The goal of $\mathcal{C}_2$ is to break the confidentiality of LZT created for S_B by any user in $\{ID_1, ID_2, \ldots, ID_m\}$ using $\mathcal{A}$ as a subroutine.

$\mathcal{C}_2$ returns a sender's identity ID_A and a random message $k||TS||ID_A||ED$ to its challenger. The challenger gives $\mathcal{C}_2$ a ciphertext σ^* (computed by LZT) as the challenge. $\mathcal{C}_2$ randomly selects $\rho \in \{1, 2, \ldots, \mu\}$. With this selection, $\mathcal{C}_2$ tries to guess $\mathcal{A}$'s selection of the target session. $\mathcal{C}_1$ keeps a state list L_s that records internal state information. Now we show that how $\mathcal{C}_2$ answers $\mathcal{A}$'s queries.

Since $\mathcal{C}_2$ knows all private keys except S_B, $\mathcal{C}_2$ can answer all queries except for the oracle S_B. For the queries that needs the private key of S_B, $\mathcal{C}_2$ uses its oracles. We explain these below.

Create $\mathcal{A}$ chooses an identity ID_i and asks a *Create* query. In this case, $\mathcal{C}_2$ can use the master secret key s to compute private key S_{ID_i}.

Corrupt $\mathcal{A}$ asks a *Corrupt* query on ID_i or S_j. $\mathcal{C}_2$ can answer the query for ID_i because $\mathcal{C}_2$ know the private key S_{ID_i}. If $S_j \neq S_B$, $\mathcal{C}_1$ also can answer it because $\mathcal{C}_2$ knows the private key SK_j. Otherwise, $\mathcal{C}_2$ fails and stops. $\mathcal{C}_2$ should set the states of corrupted oracles *Corrupted*.

Send From *Create* and *Corrupt*, we know that $\mathcal{C}_2$ knows private keys of all parties except S_B. Therefore, if $\mathcal{A}$ makes a *Send* query that does not involves S_B, $\mathcal{C}_2$ can answer it in the straightforward way because $\mathcal{C}_2$ knows their private keys. For a *Send* query that needs the private key of S_B, $\mathcal{C}_2$ answers it using its own oracles. When $\mathcal{A}$ asks a *Send* $(\prod_{B,i}^{\tau})$ query with input an ciphertext σ, $\mathcal{C}_2$ asks its unsigncryption oracle with input the ciphertext σ and identity ID_i. If it obtains a $k||TS||ID_i||ED$ from its unsigncryption oracle, $\mathcal{C}_2$ marks the oracle $\prod_{B,i}^{\tau}$ as accepted, inserts $(\tau, B, i, k||TS||ID_i||ED, \sigma, Accepted)$ in the list L_s and returns $tag = MAC_k(TS)$ to $\mathcal{A}$. If the unsigncryption oracle outputs a failure symbol $\perp$, this session is not accepted. $\mathcal{C}_2$ returns *Rejected* to $\mathcal{A}$ and inserts $(\tau, B, i, \perp, \sigma, Rejected)$ into the list L_s.

Reveal When $\mathcal{A}$ asks a *Reveal* query on the oracle $\prod_{i,j}^{\tau}$, $\mathcal{C}_2$ checks if there is an entry for (τ, i, j) in the list L_s. If yes, $\mathcal{C}_2$ returns the corresponding session key k and updates $(\tau, i, j, k||TS||ID_i||ED, \sigma, Accepted)$ into $(\tau, i, j, k||TS||ID_i||ED, \sigma, Opened)$. Otherwise, the query is considered invalid.

If $\mathcal{A}$ asks a *Send* $(\prod_{A,B}^{\rho})$ query with an empty message λ, $\mathcal{C}_2$ chooses two equal length messages k_0 and k_1 and submits $(k_0||TS||ID_A||ED, k_1||TS||ID_A||ED, ID_A)$ to its challenge oracle to get a challenge ciphertext σ^*. $\mathcal{C}_2$ returns σ^* to $\mathcal{A}$. Now, $\mathcal{A}$ can choose the $\prod_{A,B}^{\rho}$ session or a matching session established by *Send* $(\prod_{B,A}^{\rho})$ query with input an ciphertext σ^* as its target session.

If $\mathcal{A}$ chooses the above two session as the test session as expected by $\mathcal{C}_2$, $\mathcal{C}_2$ returns k_β to $\mathcal{A}$. In the end of this game, $\mathcal{A}$ outputs its guess ξ. If $\xi = 0$, $\mathcal{C}_2$ outputs $\beta = 0$ that implies k_β is a real session key and σ^* is a ciphertext of $k_\beta||TS||ID_A||ED$. Otherwise $\beta = 1$ is returned.

If the event E occurs, $\mathcal{A}$ may win this game without selecting the session in which the challenge ciphertext σ^* is injected, as the test session. In the case, $\mathcal{A}$ has no advantage. Further, when the event E occurs or $\mathcal{A}$ selects a different session other than the one wanted by $\mathcal{C}_2$ as the test session, $\mathcal{C}_2$ outputs a random bit β with a probability 1 / 2.

The probability that $\mathcal{A}$ selects a test session ρ with ID_A as initiator and S_B as responder is $1/mn\phi$. Therefore, the advantage of $\mathcal{C}_2$ is

$$\epsilon_2' \geq \frac{\Pr[\mathcal{A} \text{ wins the game}|\mathrm{E}]}{mn\phi}$$

To answer the *Send* $(\prod_{B,i}^{\tau})$ queries, $\mathcal{C}_2$ needs to ask its unsigncryption oracle. The maximum number of such queries is $m\phi - 1$. Therefore, the running time of $\mathcal{C}_2$ is

$t_2' \leq t_2 + (m\phi - 1)t_u$

According to the theorem of total probability, $\mathcal{A}$'s advantage is

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins}|E]\Pr[E] + \Pr[\mathcal{A} \text{ wins}|\bar{E}]\Pr[\bar{E}] \\ &\leq \Pr[E] + \Pr[\mathcal{A} \text{ wins}|\bar{E}] \\ &\leq \epsilon_1' mn + \epsilon_2' mn\phi\end{aligned}$$

Therefore, we have $\epsilon_1' \geq (\epsilon - \epsilon_2' mn\phi)/mn$ and $\epsilon_2' \geq (\epsilon - \epsilon_1' mn)/mn\phi$. Since LZT satisfies the confidentiality and unforgeability, ϵ_1' and ϵ_2' are negligible. Therefore, the advantage of $\mathcal{A}$ is also negligible. □

We summarize the security properties of our protocol as follows.

- *User authentication* Since the client signcrypts the message $k||TS||ID||ED$, the server can explicitly authenticate the client. Without a valid private key S_{ID}, anyone can not generate a (c, S, T) that makes the server to accept it. In addition, the server authenticates itself to the client by using *tag*.
- *Key authentication* Since the client signcrypts the message $k||TS||ID||ED$, the server can believe that k is computed by the client. So our protocol supplies explicit key authentication of the client to the server. In addition, only the server can unsigncrypt the ciphertext (c, S, T) to obtain the session key k and

Table 2 Performance comparison

Protocols	Client			Server			Communication overhead	Cryptosystem
	PC	PM	EC	PC	PM	EC		
FA [13]	0	3	0	0	3	0	$2\|G_1\| + \|ID\| + 3\|h\| + 2\|TS\| + \|ED\|$	Identity-based
WT [16]	0	4	0	2	2	0	$2\|G_1\| + \|\mathbb{Z}_p^*\| + \|ID\| + \|h\| + \|ED\|$	Identity-based
He [17]	0	3	0	1	3	0	$3\|G_1\| + \|\mathbb{Z}_p^*\| + \|ID\| + \|h\| + \|ED\|$	Identity-based
HCH [18]	0	3	0	0	3	0	$2\|G_1\| + 2\|ID\| + 2\|h\| + 2\|TS\| + \|ED\|$	Identity-based
CT [21]	0	4	1	2	3	1	$3\|G_1\| + \|ID\| + 2\|h\| + \|ED\|$	Identity-based
LH [22]	0	7	0	2	5	0	$5\|G_1\| + \|ID\| + 2\|h\| + \|ED\|$	Self-certified
HL [23]	0	7	0	2	5	0	$7\|G_1\| + 2\|h\| + \|ED\|$	Self-certified
Ours	0	2	1	2	1	1	$2\|G_1\| + \|ID\| + \|h\| + \|TS\| + \|ED\| + \|k\|$	Identity-based to PKI

Table 3 The size of q and p for the three security levels (bits)

Security level	Size of q	Size of p
80-bit level	512	160
112-bit level	1024	224
128-bit level	1536	256

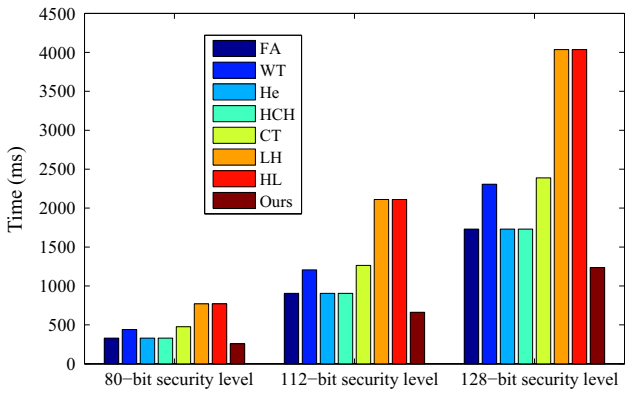


Fig. 6 The computational time of client part

send the $tag = MAC_k(TS)$. Our protocol also supplies explicit key authentication of the server to the client.

- *Key establishment* A client and the server share a session key k or $k \oplus k^*$. Without the server's private key SK, anyone can not derive the r to get session key k or $k \oplus k^*$.
- *Key confirmation* Since the client signcrypts the message $k\|TS\|ID\|ED$, the server can conceive that the client has possessed the session key k. Our protocol provides the key confirmation of the client to the server.

  In addition, the client checks if $tag' \stackrel{?}{=} tag$. If the client accepts this session, the client can believe that the server has possessed the session key k. So our protocol provides the key confirmation of the server to the client.
- *Anonymity* Since the client's identity information is encrypted using $H_2(r)$, the anonymity of the client is obtained.

4.3 Performance

We compare the major computational cost and communication overhead of our protocol with those of existing seven authentication and key establishment protocols for MCS environment in Table 2 (Here we only consider the authentication and key establishment phase and ignore the other phases). We represent Pairing Computation by PC, Point Multiplication in G_1 by PM and Exponentiation Computation in G_2 by EC. We ignore the other operations in this comparison since the above three operations consume the most running time of the whole algorithm. We denote the number of bits of x by $|x|$. In FA, WT, He, HCH, LH and HL, the expiration date ED is not considered. For the fair comparison, we add the ED in the communication overhead column for the six protocols. From Table 2, we find that our protocol has the lowest cost in computation and communication.

In order to more clearly demonstrate our protocol, we give a quantitative comparison for FA, WT, He, HCH, CT, LH, HL and our protocol. We use JPBC Type A pairing [32] in our analysis. The Type A pairing is constructed on the elliptic curve

$$y^2 \equiv (x^3 + x) \bmod q$$

with some prime numbers $q \equiv 3 \bmod 4$, where the embedding degree is two and the order of G_1 is p. Here we use three types of parameters which represents the 80-bit, 112-bit and 128-bit key size security levels, respectively. Table 3 describes the size of q and p for the three security levels.

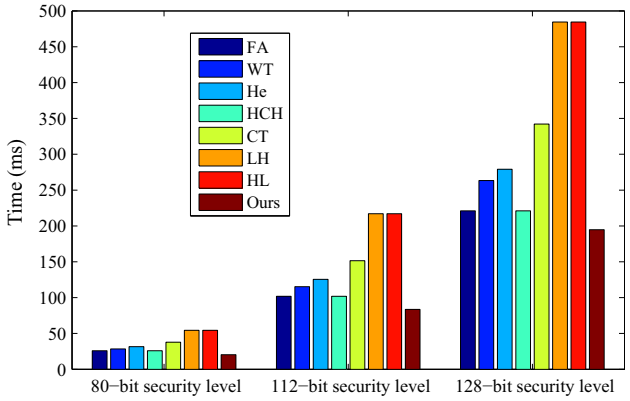


Fig. 7 The computational time of server part

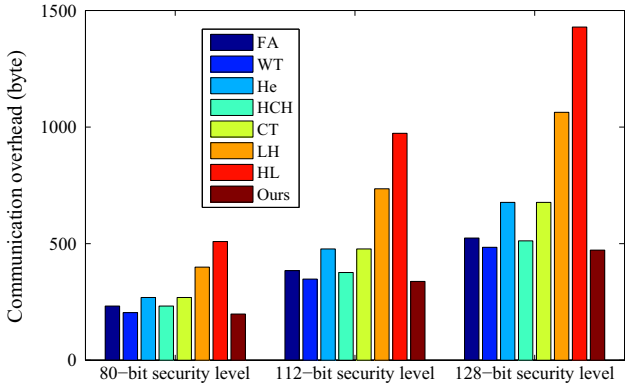


Fig. 8 The communication overhead

Figures 6 and 7 respectively give the computational time of the client and the server for the eight protocols at the three security levels. The client part is implemented on MEIZU M2 mobile phone that is equipped with a MediaTek MT 6735 1.3 GHz machine with 2G RAM. The server part is implemented on ThinkCentre E73 computer that is equipped with an Intel Core i7 4770S 3.1 GHz machine with 4G RAM. From Figs. 6 and 7, we find that our protocol is the most efficient.

Now we consider the communication overhead. We assume that $|ID| = 160$, $|TS| = 32$ and $|ED| = 112$. Note that for the 80-bit, 112-bit and 128-bit security levels, the corresponding sizes of hash value are 160 bits, 224 bits and 256 bits, respectively. When we accept the 80-bit security level, the size of p, q, G_1 and G_2 is 160, 512, 1024 and 1024 bits, respectively. Using the compression method in [33], we can reduce the size of an element in G_1 to 65 bytes. Therefore, the communication overheads of FA, WT, He, HCH, CT, LH, HL and our protocol are $2|G_1| + |ID| + 3|h| + 2|TS| + |ED|$ bits $= 2 \times 65 + 20 + 3 \times 20 + 2 \times 4 + 14 = 232$ bytes, $2|G_1| + |\mathbb{Z}_p^*| + |ID| + |h| + |ED|$ bits $= 2 \times 65 + 20 + 20 + 20 + 14 = 204$ bytes, $3|G_1| + |\mathbb{Z}_p^*| + |ID| + |h| + |ED|$ bits $= 3 \times 65 + 20 + 20 + 20 + 14 = 269$ bytes, $2|G_1| + 2|ID| + 2|h| + 2|TS| + |ED|$ bits $= 2 \times 65 + 2 \times 20 + 2 \times 20 + 2 \times 4 + 14 = 232$ bytes, $3|G_1| + |ID| + 2|h| + |ED|$ bits $= 3 \times 65 + 20 + 2 \times 20 + 14 = 269$ bytes, $5|G_1| + |ID| + 2|h| + |ED|$ bits $= 5 \times 65 + 20 + 2 \times 20 + 14 = 399$ bytes, $7|G_1| + 2|h| + |ED|$ bits $= 7 \times 65 + 2 \times 20 + 14 = 509$ bytes, $2|G_1| + |ID| + |h| + |TS| + |ED| + |k|$ bits $= 2 \times 65 + 20 + 20 + 4 + 14 + 10 = 198$ bytes, respectively. Similarly, we can obtain the communication overheads at the 112-bit and 128-bit security levels. We summarize the communication overheads of the eight protocols at the 80-bit, 112-bit and 128-bit security levels in Fig. 8. From Fig. 8, we find that the communication cost of our protocol is the lowest.

The Table 4 summarizes the advantage of our protocol over existing seven protocols. The advantage includes the computational efficiency (both client part and server part) and communication overhead. For the typical 80-bit security level, our protocol is 22.25%, 41.69%, 22.25%, 22.25%, 46.16%, 66.68% and 66.68% faster than FA, WT, He, HCH, CT, LH and HL, respectively, in the client part and is 21.44%, 28.29%, 35.31%, 21.44%, 45.91%, 62.51%

Table 4 Advantage of our protocol over existing seven protocols

Protocols	Advantage in computational efficiency						Advantage in communication overhead		
	Client part			Server part					
	80-bit (%)	112-bit (%)	128-bit (%)	80-bit (%)	112-bit (%)	128-bit (%)	80-bit (%)	112-bit (%)	128-bit (%)
FA [13]	22.25	26.87	28.62	21.44	17.9	11.92	14.66	11.98	9.92
WT [16]	41.69	45.16	46.46	28.29	27.54	26.02	2.94	2.87	2.48
He [17]	22.25	26.87	28.62	35.31	33.43	30.2	26.39	29.14	30.28
HCH [18]	22.25	26.87	28.62	21.44	17.9	11.92	14.66	10.11	7.81
CT [21]	46.16	47.69	48.29	45.91	44.81	43.08	26.39	29.14	30.28
LH [22]	66.68	68.66	69.41	62.51	61.51	59.79	50.38	54.01	55.6
HL [23]	66.68	68.66	69.41	62.51	61.51	59.79	61.1	65.26	66.97

and 62.51% faster than FA, WT, He, HCH, CT, LH and HL, respectively in the server part. For the same 80-bit security level, compared with FA, WT, He, HCH, CT, LH and HL, the communication overhead of our protocol is respectively reduced by 14.66%, 2.94%, 26.39%, 14.66%, 26.39%, 50.38% and 61.1%.

5 Conclusion

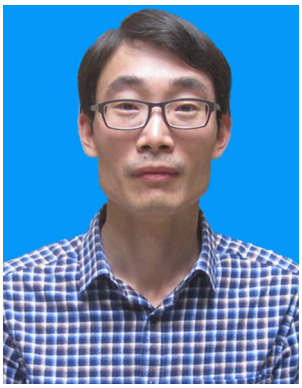
This paper proposed a heterogeneous user authentication and key establishment protocol using the LZT signcryption. In our protocol, the client belongs to the IBC and the server belongs to the PKI. As compared with previous works, our protocol has the lowest cost in computation and communication. Although this paper focuses on the single server environment, our protocol is also suitable for the multi-server environment. The heterogeneous user authentication and key establishment is very suitable for solving the security problems of heterogeneous networks. A key problem in designing such protocols is how to ensure the key consistency between the client and the server since they use different public key authentication method. A possible future work would be to design user authentication and key establishment protocols between PKI and SCC.

Acknowledgements This work is supported by the Science and Technology Programs of SGCC titled application research on improving the reliability guarantee capability of information systems (Grant No. 546803170005).

References

1. Lu, Y., Li, L., Peng, H., & Yang, Y. (2016). Robust anonymous two-factor authenticated key exchange scheme for mobile client–server environment. *Security and Communication Networks*, *9*(11), 1331–1339.
2. Najaflou, Y., Jedari, B., Xia, F., Yang, L. T., & Obaidat, M. S. (2015). Safety challenges and solutions in mobile social networks. *IEEE Systems Journal*, *9*(3), 834–854.
3. Zhang, K., Liang, X., Lu, R., & Shen, X. (2015). PIF: A personalized fine-grained spam filtering scheme with privacy preservation in mobile social networks. *IEEE Transactions on Computational Social Systems*, *2*(3), 41–52.
4. Hu, X., Chu, T. H. S., Leung, V. C. M., Ngai, E. C. H., Kruchten, P., & Chan, H. C. B. (2015). A survey on mobile social networks: Applications, platforms, system architectures, and future research directions. *IEEE Communications Surveys Tutorials*, *17*(3), 1557–1581.
5. Senftleben, M., Barroso, A., Bucicoiu, M., Hollick, M., Katzenbeisser, S., & Tews, E. (2016). On the privacy and performance of mobile anonymous microblogging. *IEEE Transactions on Information Forensics and Security*, *11*(7), 1578–1591.
6. Buchmann, J. A., Karatsiolis, E., & Wiesmaier, A. (2013). *Introduction to public key infrastructures*. Berlin: Springer.
7. Boneh, D., & Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, *32*(3), 586–615.
8. Girault, M. (1991). Self-certified public keys. In D. Davies (Ed.), *Advances in cryptology-EUROCRYPT'91. Lecture notes in computer science* (Vol. 547, pp. 490–497). Berlin: Springer.
9. Yang, J. H., & Chang, C. C. (2009). An ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem. *Computers & Security*, *28*(3–4), 138–143.
10. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126.
11. Yoon, E. J., & Yoo, K. Y. (2009). Robust ID-based remote mutual authentication with key agreement scheme for mobile devices on ECC. In *International conference on computational science and engineering (CSE '09)* (Vol. 2, pp. 633–640).
12. Chou, C. H., Tsai, K. Y., & Lu, C. F. (2013). Two ID-based authenticated schemes with key agreement for mobile environments. *The Journal of Supercomputing*, *66*(2), 973–988.
13. Farash, M., & Attari, M. (2014). A secure and efficient identity-based authenticated key exchange protocol for mobile client–server networks. *The Journal of Supercomputing*, *69*(1), 395–411.
14. Shi, R. H., Zhong, H., & Zhang, S. (2015). Comments on two schemes of identity-based user authentication and key agreement for mobile client–server networks. *The Journal of Supercomputing*, *71*(11), 4015–4018.
15. Qi, M., & Chen, J. (2017). An efficient two-party authentication key exchange protocol for mobile environment. *International Journal of Communication Systems*, *30*(16), e3341.
16. Wu, T. Y., & Tseng, Y. M. (2010). An efficient user authentication and key exchange protocol for mobile client–server environment. *Computer Networks*, *54*(9), 1520–1530.
17. He, D. (2012). An efficient remote user authentication and key agreement protocol for mobile client–server environment from pairings. *Ad Hoc Networks*, *10*(6), 1009–1016.
18. He, D., Chen, J., & Hu, J. (2012). An ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security. *Information Fusion*, *13*(3), 223–230.
19. Wang, D., & Ma, C. (2013). Cryptanalysis of a remote user authentication scheme for mobile client–server environment based on ECC. *Information Fusion*, *14*(4), 498–503.
20. Hassan, A., Eltayieb, N., Elhabob, R., & Li, F. (2017). An efficient certificateless user authentication and key exchange protocol for client–server environment. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-017-0622-1.
21. Chuang, Y. H., & Tseng, Y. M. (2012). Towards generalized ID-based user authentication for mobile multi-server environment. *International Journal of Communication Systems*, *25*(4), 447–460.
22. Liao, Y. P., & Hsiao, C. M. (2013). A novel multi-server remote user authentication scheme using self-certified public keys for mobile clients. *Future Generation Computer Systems*, *29*(3), 886–900.
23. Hsieh, W. B., & Leu, J. S. (2014). An anonymous mobile user authentication protocol using self-certified public keys based on multi-server architectures. *The Journal of Supercomputing*, *70*(1), 133–148.
24. Li, F., Han, Y., & Jin, C. (2018). Cost-effective and anonymous access control for wireless body area networks. *IEEE Systems Journal*, *12*(1), 747–758.
25. Li, F., Zhang, H., & Takagi, T. (2013). Efficient signcryption for heterogeneous systems. *IEEE Systems Journal*, *7*(3), 420–429.

26. Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, *1*(1), 36–63.
27. Bellare, M., & Rogaway, P. (1994). Entity authentication and key distribution. In D. R. Stinson (Ed.), *Advances in cryptology-CRYPTO'93. Lecture notes in computer science* (Vol. 773, pp. 232–249). Berlin: Springer.
28. Blake-Wilson, S., Johnson, D., & Menezes, A. (1997). Key agreement protocols and their security analysis. In M. Darnell (Ed.), *Crytography and coding. Lecture notes in computer science* (Vol. 1355, pp. 30–45). Berlin: Springer.
29. Chen, L., & Kudla, C. (2003). Identity based authenticated key agreement protocols from pairings. In *16th IEEE computer security foundations workshop (CSFW'03)* (pp. 219–233).
30. McCullagh, N., & Barreto, P. S. (2005). A new two-party identity-based authenticated key agreement. In A. Menezes (Ed.), *Topics in cryptology-CT-RSA 2005. Lecture notes in computer science* (Vol. 3376, pp. 262–274). Berlin: Springer.
31. Gorantla, M. C., Boyd, C., & González Nieto, J. M. (2007). On the connection between signcryption and one-pass key establishment. In S. Galbraith (Ed.), *Cryptography and coding. Lecture notes in computer science* (Vol. 4887, pp. 277–301). Berlin: Springer.
32. De Caro, A., & Iovino, V. (2011). jPBC: Java pairing based cryptography. In *16th IEEE symposium on computers and communications (ISCC 2011)*, Kerkyra, Greece (pp. 850–855).
33. Shim, K. A. (2012). CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, *61*(4), 1874–1883.



Fagen Li is a Professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P. R. China. He received his Ph.D. degree in Cryptography from Xidian University, Xi'an, P. R. China in 2007. From 2008 to 2009, he was a postdoctoral fellow in Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science (JSPS). He worked as a research fellow in the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan from 2010 to 2012. His recent research interests include cryptography and network security. He has published more than 80 papers in international journals and conferences. He is a member of the IEEE.

Jiye Wang is the Director of Department of Information and Communications Technology, State Grid Corporation of China. His research interests include electric power informatization and network security.

Yuyang Zhou is a Ph.D. student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P. R. China. Her research interests include cryptography and information security.

Chunhua Jin is now a Lecturer in the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huai'an, P. R. China. She received her Ph.D. degree in Information Security from University of Electronic Science and Technology of China (UESTC), Chengdu, P. R. China in 2016. Her recent research interests include cryptography and network security.

SK Hafizul Islam is an Assistant Professor in the Department of Computer Science and Engineering, Indian Institute of Information Technology, Kalyani, West Bengal, India. He received his Ph.D. Degree in Computer Science and Engineering from Indian School of Mines, Dhanbad, India in 2013. His research interests include cryptography and information security.