Towards Smart Card Based Mutual Authentication Schemes in Cloud Computing

Haoxing Li[1], Fenghua Li[2,*], Chenggen Song[3], Yalong Yan[3]
[1] State Key Laboratory of Integrated Services Networks, Xidian University, Shaanxi, Xi'an, China
[E-mail: lhx595@126.com]
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[E-mail : lfh@iie.ac.cn]
[3] Institute of Information Security, Beijing Electronic Science and Technology Institute, Beijing, China
[E-mail:yalong@besti.edu.cn]
*Corresponding author: Fenghua Li

Received January 6, 2015; revised March 20, 2015; accepted April 19, 2015; published July 31, 2015

Abstract

In the cloud environment, users pay more attentions to their data security since all of them are stored in the cloud server. Researchers have proposed many mutual authentication schemes for the access control of the cloud server by using the smart card to protect the sensitive data. However, few of them can resist from the smart card lost problem and provide both of the forward security and the backward security. In this paper, we propose a novel authentication scheme for cloud computing which can address these problems and also provide the anonymity for the user. The trick we use is using the password, the smart card and the public key technique to protect the processes of the user's authentication and key exchange. Under the Elliptic Curve Diffie-Hellman (ECDH) assumption, it is provably secure in the random oracle model. Compared with the existing smart card based authentication schemes in the cloud computing, the proposed scheme can provide better security degree.

Keywords: Authentication, smart card, cloud computing, smart card lost, forward security

This work was supported by The National Natural Science Foundation of China(61170251), the National High Technology Research and Development Program of China (863 Program)(2012AA013102, 2012AA01A401) and the Major Science and Technology Project of Press and Publication-Research and Development (1681300000119). The authors would like to thank the editor and all the anonymous reviewers for their useful suggestion.

http://dx.doi.org/10.3837/tiis.2015.07.022 **ISSN : 1976-7277**

1. Introduction

Cloud computing is a new technology, which is a hot topic in the past dedicate in both of the academic and industry. On one hand, users can take advantages of the cloud server to accomplish complicate calculation which cannot be processed locally. On the other hand, users can store a large number of data in the cloud server to save their own memory space [1]. Therefore, the individuals especially the companies are interesting in outsourcing the service to cloud service provider in order to reduce the cost of management and deployment. Lots of international firms have established their cloud platforms and offered cloud computing services for the Internet users, such as Google App Engine, Microsoft Windows Azure, Amazon Web Services and IBM SmartCloud.

However, users who take advantages of these cloud computing services pay much attentions to the security of their data since the data are outsourced by the cloud server. The secure issues that the individual or the companies concern about in the cloud environment include access control, data integrity, data confidentiality, authentication and authorization [2]. Among them, authentication is important. Becase without a secure authentication scheme the data of the user will be obtained by the illegal person. The authentication between the user and the cloud server cannot only guarantee the data be accessed by the legitimate users successfully but also exclude the malicious visitor. So when using the cloud service, authentication between the user and the server should be considered firstly.

Authentication is the first step when a user accesses his cloud data. It is important for the authorized user to get his service safely and smoothly. Authentication schemes using smart card can provide more convenience and security for the user than other authentication schemes since on one hand users do not need to remember long secret value comparing with the public key mechanism; on the other hand it can provides more security property than authentication schemes using only password [3]. So lots of authentication scheme using smart card were proposed in cloud computing. However, many of them cannot resist the smart card lost attack. Meanwhile, few of them consider the forward and backword security since they cannot be implemented easily. However, these two properties are important. Because we do not know what will happen in the future, if the adversary gets all of our secrets in the future and recovers our conversation which had been encrypted by the session key in the old session or obtains the conversation which is encrypted by the session key in the new session, then it will be a big threat to us.

So how to get an authentication scheme which can both resist the smart card lost attack and provide the forward and backward security in the cloud computing is a chanllege to the researchers. Because the public key techniques do not lie in authentication schemes so these schemes cannot provide the strong secure property when the smart card is lost. Halevi and Krawczyk [4] have pointed out that public key techniques were unavoidable for password protocols that resist off-line dictionary attacks. Following this rule, in this paper, we propose a new authentication scheme for cloud computing using smart card. In the new scheme, even if the smart card is lost, the authentication scheme is still secure. Meanwhile, the new scheme can also provide the anonymity for the user and the properties of forward and backward security. Under the ECDH assumption, the new scheme is provably secure in the random oracle model. The communication framework is also very suitable for the mobile cloud computing which is a hot topic in the next generation communicaiton sicne both of them use the three-level authentication.

In Section 2, we review the previous work of authentication schemes in cloud computing using smart card. In Section 3, we give a security model for authentication schemes using smart card. In Section 4, we present a new authentication scheme for cloud computing using smart card. We then give the security analysis and the performance of the proposed scheme in Section 5. Finally, in Section 6 we make a conclusion of this paper and give the future work.

2. Related Work

Authentication schemes between the user and the server are based on the password technique in the early stage [5-6]. However, there are two drawbacks using such method. On one hand, the passwords of the user are short and they are often chosen from names or numbers they frequently use. So the passwords can be guessed by the attacker, i.e., the authentication shcemes are vulnerable to the dictionary attack [7]. On the other hand, in such scenario all the passwords of the users will be stored in the server. Once the server is corrupted, all the users'passwords will be revealed. In order to address these problems, authentication using password and smart card were proposed [3,8]. In such authentication, user owns a password and a smart card. Only if both of the password and the smart card are correct, the user can login to the server successfully. Lots of the authentication schemes using smart card were present in the cloud environment [9-16].

Choudhur et al. proposed a strong user authentication framework for cloud computing [9]. The new method Out of Brand (OOB) authentication combined with the smart card and the password authentication was used in [9]. However, Chen and Jiang found the security weaknesses of Choudhur et al.'s authentication scheme [10]. There are masquerading attack and the OOB attack in [9] if the smart card of the user is lost. Then, Chen and Jiang proposed an improvement user authentication framework for cloud computing [10]. However, the improvement scheme is not secure. When the attacker obtains the information in the smart card, he can launch the offline dictionary attack. Because the user's messages are only protected by the password and the information stored in the smart card, so the authentication scheme is vulnerable to the offline dictionary attack when the smart card is lost. The attacker can obtain the information stored in the smart card and guesse the password of the user, then he can verify the correctness of his guess by the authentication messages sent to the cloud server. The same attack also lies in Jiang's authentication scheme for cloud computing [11]. Han et al. propsed a scheme for data confidentiality in cloud computing for wireless body area networks which further expends the application of the cloud computing [12].

Different from the authentication schemes mentioned above, Nimmy and Sethumadhavan proposed a mutual authentication scheme for cloud computing using secret sharing [13]. The server splits the credential of the user into two shares, one is stored in the smart card and the other is stored in the server. It seems that only getting both of the shares can recover the credential of the user. However, the server still depends on the information stored in the smart card to verify the identity of the user. When getting the smart card, the adversary can also launch the offline dictionary attack and impersonate the user to access the cloud. So the method of the secret share does not provide more security. In order to reduce the time of the authentication, Hao et al. proposed a time-bound ticket-based mutual authentication scheme for cloud environment using smart card [14]. The advantage of Hao et al.'s authentication scheme is that the server issues a certain number of digital tickets to the user. The user can use one ticket for one time of data verification so it can save the time of the authentication since the user's data verification frequency is reduced. However, although Hao et al. claimed that the authentication scheme was secure, Pippal et al. found it was vulnerable to

Denial-of-Service attack and the password change phase was insecure [15]. To resist these weaknesses, Pippal et al. proposed an enhancement to Hao et al.'s scheme. The trick they used in [15] is that the smart card verifies both password and the user's identifier at the user side before sending the authentication message to the cloud server. If the smart card is a tamper resistant device, the trick they used is helpful. However, as we know, most of the smart cards are not tamper resistant for the two reasons that the tamper resistant smart cards are expensive and the parameters in these smart cards can also be extracted by the side-channel attack [16].

Using the tamper resistant smart card to address the problem in authentication between the user and the server is not a good choice. Huang et al. proposed robust and privacy protection authentication in cloud computing using a third trusted party [17] without using the tamper resistant smart card. The authentication scheme is secure and also has good performance. The only doubtful point is that the discussion of the reliability and complexity of introducing the trusted party. Meanwhile, the forward security and backward security are not considered in [17].

3. Security Model

The security model we use is based on the models proposed by Bellare et al. [6] and Zhou et al. [18], respectively. In the model, there are three entities: the user *U*, the server *S* and the adversary $\mathcal{A}$. The user owns his password and a smart card which is issued from the server. The server owns his private information and the user's registration information. The adversary can control all the communication between the user and the server. The ability of the adversary is based on the queries to the protocol instances. One execution of the protocol is called an instance. The queries that $\mathcal{A}$ can ask are as follows:

Execute ($\prod_U^i, \prod_S^j$):This query models passive attacks. The adversary $\mathcal{A}$ often gets the protocol flows between instances $\prod_U^i$ and $\prod_S^j$ by eavesdropping. The output of this query is the honest execution of the protocol.

Send ($\prod_U^i / \prod_S^j, m$):This query models the active attacks. $\mathcal{A}$ who impersonates *U* to send an message *m* to instance $\prod_U^i / \prod_S^j$. The output of this query is the response generated by the instance after it processes *m* according to the protocol.

Reveal ($\prod_U^i / \prod_S^j$):This query models the misuse of the session key or the known key attack. The output of this query is the session key of instance $\prod_U^i / \prod_S^j$. This only happens when the attacked instance actually holds a session key.

Corrupt (*U*, *password*): The output of this query is the password of the user.

Corrupt (*U*, *smart card*): The output of this query is the secret information which are stored in the smart card.

Test ($\prod_U^i / \prod_S^j$):The semantic security of the session key is modeled by this query. When $\mathcal{A}$ chooses a session as the Test session and asks this session the *Test* ($\prod_U^i / \prod_S^j$) query. The query is answered as follows: one flips a coin *b*, if *b*=1 it outputs the session key sk_{US} to $\mathcal{A}$; if *b*=0, it outputs a random value chosen from session key space to $\mathcal{A}$. This query can only be asked to instance which is *fresh* and can be asked at most once. An instance $\prod_U^i$ is fresh if: (1).

it is not asked by the *Reveal* query; (2). the instance which has a matching conversation with $\prod_U^i$ is not asked by the *Reveal* query either.

AKE(Authenticated Key Exchange) Security The privacy of the session key is modeled by the game between the adversary and a simulator . The simulator simulates the protocol for the adversary and answer the queries $\mathcal{A}$ asks. When $\mathcal{A}$ asks a *Test* ($\prod_U^i / \prod_S^j$) query, he needs to output a bit *b*'. The aim of $\mathcal{A}$ is correctly guessing the bit *b* in the Test session. The protocol *P* is said to be AKE-secure if for any polynomial time adversary $\mathcal{A}$ the following equation holds:

$$Adv_{P,D}^{AKE}(\mathcal{A}) = \left|2\Pr[b' = b] - 1\right| = \frac{O(q_{send})}{N} + neg(l)$$

Where q_{send} is the number of the *Send* query, N is the the size of the password dictionary and $neg(l)$ is a negligible value.

4. A New Authentication Scheme for Cloud Computing Using Smart Card

In this section we propose a new authentication scheme for cloud computing using smart card. We first give the authentication structure of the cloud computing we used in **Fig. 1**.

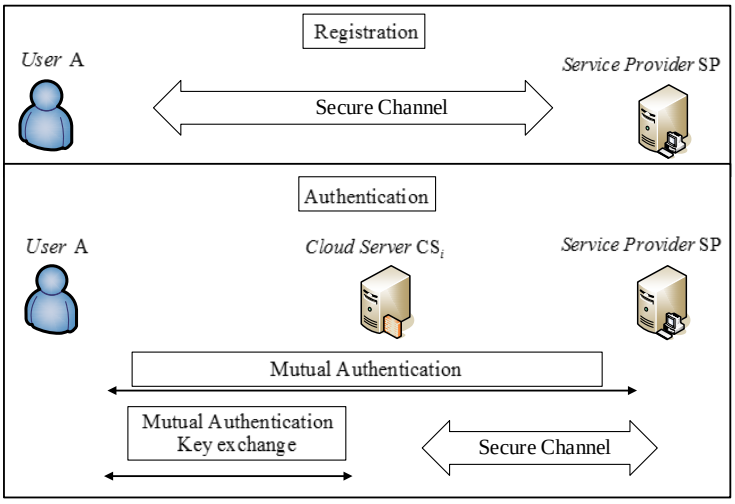


Fig. 1. Authentication structure of the cloud computing used in this paper

As shown in **Fig. 1**, there are three kinds of entities in the scheme: a cloud user A, some cloud servers CS_i and a service provider SP. Here we assume SP is a trusted third party and CS_i are semi-trusted servers, i.e., CS_i are honest but curious and they cannot launch the active attack but they are curious about the password of the user. When the user wants to get the cloud service, he needs to register in the service provider. The service provider SP issues a credential for the use in a secure channel. Here SP does not provide service for the user directly. Actually, it administrates a group of cloud servers $\{CS_1, CS_2, ..., CS_n\}$. These cloud servers provide service for the user directly, such as storing the sensitive data or dealing with complex computation. There is a secure channel between CS_i and SP. However, when the user logins to cloud servers, these cloud servers cannot authenticate the user alone. The authentication between the user and CS_i must be completed by the help of SP.

Let's consider a real example in the cloud computing to show the problem we want to solve. A cloud user has registered in a service provider SP and owns his password and smart card. In order to reduce the burden of SP and avoid the case of single point of failure, SP disperses some of the service to certain cloud server CS_i. So some of the service are provided by CS_i now. Unfortunately, the smart card of the user is lost some day. In this case, how can the user believe his data stored in CSi are still secure if the attacker gets his smart card but no password? In this paper, we propose a new authentication scheme using smart card to answer this question.

The new authentication scheme includes three phases, the registration phase, the authentication and key exchange phase and the password-changing phase. The first and third parts are similar to that of existing authentication schemes [14-15]. The innovation is in the second part. Firstly, we invite the three-level authentication model into cloud computing which is different from the trick in other authentication schemes in cloud computing [8-16]. The advantage of the three-level authentication model is that: on one hand, it is more easy to convert the static authentication schemes to the dynamic authentication schemes in the three-level model since they use the same framework, i.e., the authentication scheme is more easy to be evolved into a roaming authentication for cloud computing; one the other hand, the new scheme disperses the computation and communication burden of the service provider to the cloud servers. This method avoids the case that when a large number of connection requests between the user and the server provider, the service provider may not be able to deal with these requests in time, then there is a delay experienced by the users. Secondly, we use a new trick in the authentication scheme, i.e., using the ECDH problem to establish a secure key between the user and the service provider, then using this key to encrypt the user's credential and complete the authentication and generate a new session key between the user and the cloud server. The new trick makes the scheme can resist the smart lost attack. Thirdly, we bring the forward and backward security into the scheme which makes the session be secure even if the long term secret of the user and the server are corrupted.

The notations we use are in **Table 1** Note we do not give a definition in detail for the hash function, we just use $h(\cdot)$ as a class of cryptographically secure hash functions.

Table 1. Notations

Notation	Description
G	a group with order a large prime q
P	a generator of G
l	a secure parameter
q	a large prime
A	the user
CS_i	the ith cloud server
SP	the cloud service provider
PW_A	A's human-memorizable password
s	SP's private key
sP	SP's public key
E	a symmetric encryption algorithm
K_{CS_i-SP}	the encryption key shared between CS_i and SP
$\oplus$	bitwise exclusive-OR operator
$h(\cdot)$	a class of cryptographically secure hash functions whose length is l bits.

4.1 Registration Phase

The registration phase happens between users and the cloud service provider. When a user wants to get the cloud service, he needs to register in the service provider SP. **Fig. 2** shows the details of the registration phase.

Step 1. User A first selects PW_A as his password. Then, in order to increase the entropy of PW_A, A chooses a random value $r \in Z_q^*$ and computes $h(PW_A \| r)$. A sends $\{ID_A, h(ID_A \| r)\}$ to the service provider through a secure channel.

Step 2. When receiving the messages, SP selects a random value $R \in \{0,1\}^{64}$ and creates a credential $C_A = h(s \| ID_A \| R) \oplus h(PW_A \| r)$ for A. SP puts the value R in his data space and issues a smart card which contains $\{ID_A, C_A\}$ to A.

Step 3. When receiving the smart card, A imbeds r into the smart card. Now the information in the smart card is $\{ID_A, r, C_A\}$.

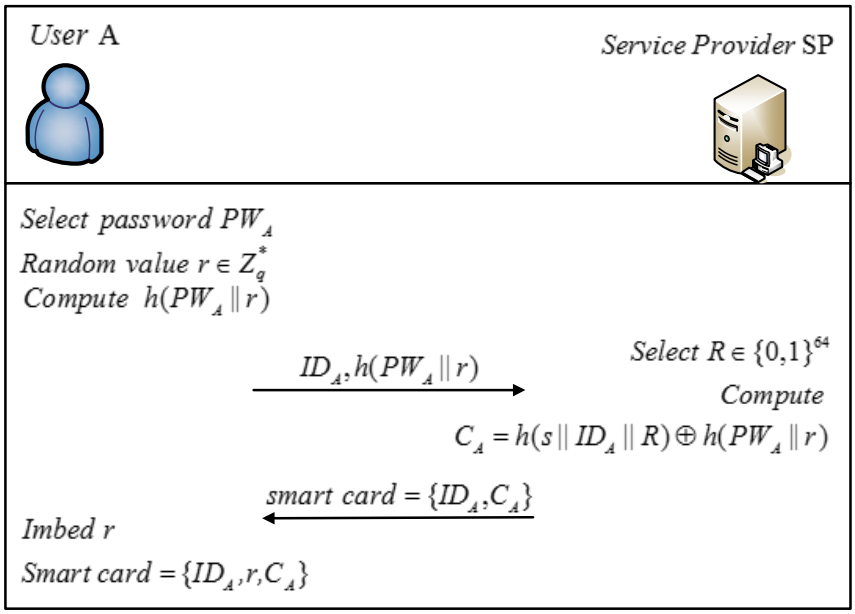


Fig. 2. The registration phase

4.2 Authentication and Key Exchange Phase

When the user wants to get the cloud service, he needs to complete a mutual authentication and key exchange with the *i*th cloud server CS_i. **Fig. 3** shows the details of the authentication and key exchange phase.

Step 1. $A \rightarrow CS_i$

User A inserts his smart card and inputs his password PW_A. Then, he selects two random values, $a, r_1 \in Z_q^*$. A computes $K = h(a \cdot sP)$ and $M_A = h(K \| r_1) \oplus ID_A$ and reveals the secret value $X_A = h(PW_A \| r) \oplus C_A$. Then, A computes authentication message $N_A = h(K \| r_1 \| X_A)$ and sends $\{aP, r_1, M_A, N_A\}$ to the cloud server CS_i.

Step 2. $CS_i \rightarrow SP$

On receiving $\{aP, r_1, M_A, N_A\}$, CS_i selects a random value $b \in Z_q^*$ and computes

$M_{CS_i} = E_{K_{CS_i-SP}}(aP, bP, r_1, M_A, N_A)$. Then, CS_i sends $\{ID_{CS_i}, M_{CS_i}\}$ to the service provider SP.

Step 3. $SP \rightarrow CS_i$

On receiving $\{ID_{CS_i}, M_{CS_i}\}$, SP first decrypts M_{CS_i} and obtains $\{aP, bP, r_1, M_A, N_A\}$. Then, SP computes $K = h(s \cdot aP)$ and gets the identity of the user by $ID_A = h(K \| r_1) \oplus M_A$. SP computes $X_A = h(s \| ID_A \| R)$ by the user's identity ID_A. After that, *SP* verifies whether $N_A = h(K \| r_1 \| X_A)$ holds. If it does, SP rejects it and requires the user to send the messages again. Otherwise, SP selects a random value $s_1 \in Z_q^*$ and computes its authentication message $Auth_{SP} = h(K \| s_1 \| aP \| bP)$, $M_{SP} = E_{K_{CS_i-SP}}(ID_A, aP, bP, s_1, Auth_{SP})$. *SP* sends M_{SP} to CS_i.

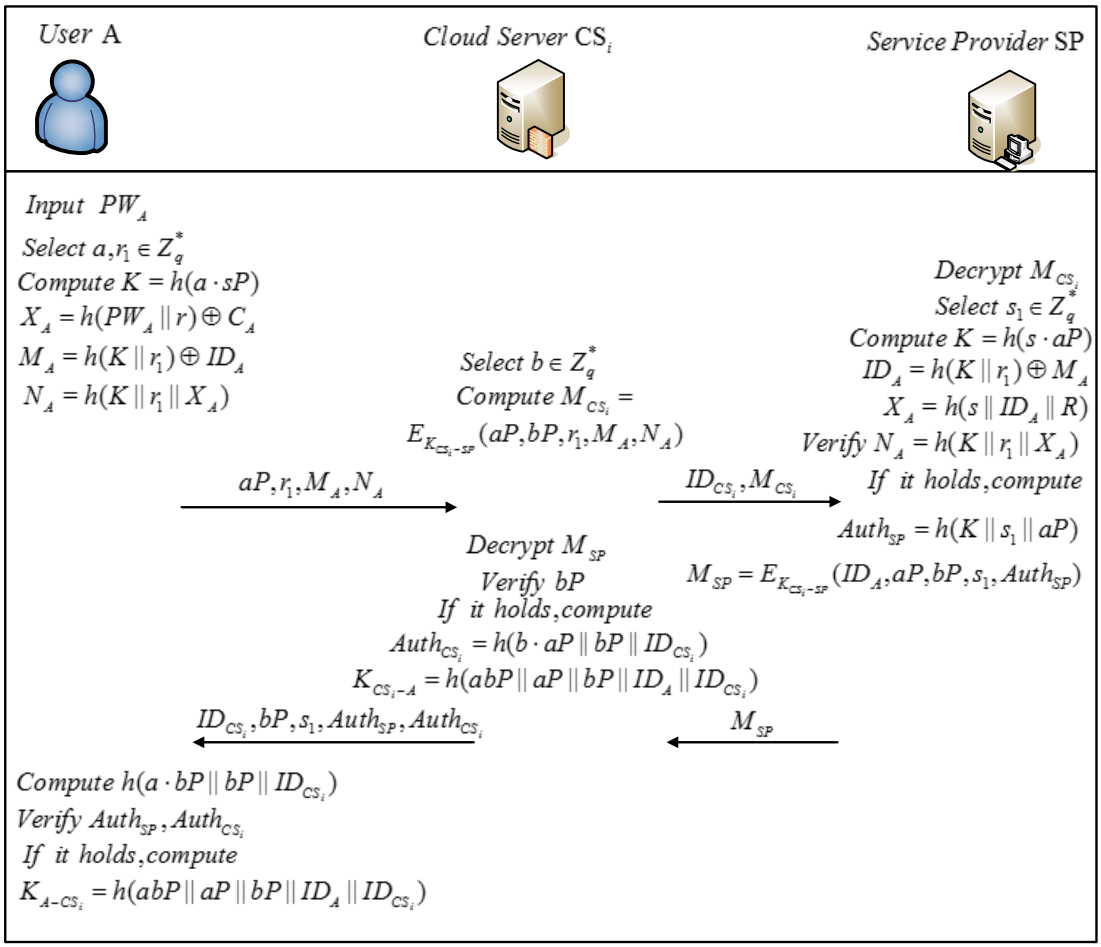


Fig. 3. The authentication phase of the proposed protocol

Step 4. $CS_i \rightarrow A$

On receiving M_{SP}, CS_i first decrypts M_{SP} and obtains $\{aP, bP, s_1, Auth_{SP}\}$. Then CS_i verifies whether bP is equal to the random value it chooses. If it is not equal, CS_i rejects it. Otherwise, CS_i computes its authentication message $Auth_{CS_i} = h(b \cdot aP \| bP \| ID_{CS_i})$ and the session key between CS_i and A, $K_{CS_i-A} = h(abP \| aP \| bP \| ID_A \| ID_{CS_i})$. CS_i sends $\{ID_{CS_i}, bP, s_1, Auth_{SP}, Auth_{CS_i}\}$ to A.

Step 5. On receiving the messages from CS_i, user A computes and verifies whether $Auth_{SP} = h(K \| s_1 \| aP \| bP)$ and $Auth_{CS_i} = h(a \cdot bP \| bP \| ID_{CS_i})$ hold. If one of them does not hold, user A rejects them. Otherwise, A computes the session key between A and CS_i $K_{CS_i-A} = h(abP \| aP \| bP \| ID_A \| ID_{CS_i})$.

4.3 Password-changing Phase

If the user wants to change his password, he needs to go through the authentication phase first. It means if the user wants to change the password, he needs to have the old password in hand and so does the smart card. After a successful authentication, the user A gets the secret information $h(K \| s_1)$ shared with SP. Then, A inputs his new password PW_{new}, selects a random value $r' \in Z_q^*$ and submits $E_{h(K\|s_1)}(ID_A, h(PW_{new} \| r'))$ to the cloud service provider. On receiving the message, SP decrypts it and obtains the new password of A. Then, SP selects another random value $R' \in \{0,1\}^{64}$ and creates a new credential $C_A' = h(s \| ID_A \| R') \oplus h(PW_{new} \| r')$ for A. SP sends $E_{h(K\|s_1)}(ID_A, C_A')$ to A. A decrypts $E_{h(K\|s_1)}(ID_A, C_A')$ and updates the information in the smart card with $\{ID_A, C_A', r'\}$.

5. Security and Performance Analysis

5.1 Security Analysis

We analyze the security of the proposed authentication and key exchange protocol in the model mentioned in Section 3. The security of the scheme is based on the Elliptic Curve Computational Diffie-Hellman (ECDH) Assumption. We first summarize the proof in order to give a clear understanding for readers. The proof is based on the security games between the adversary and a simulator who simulates the protocol for the adversary. The simulator revises the games one by one and imbeds an ECDH problem into the protocol in the last game. In the last game, the protocol is almost random so does the session key which is computed from the ECDH tuple that the simulator imbeds. So if the adversary can correctly guess the session key and wins the game, then the simulator can breaks the ECDH assumption by using the adversary as a subroutine. This is the mainline of the proof.

Elliptic Curve Diffie-Hellman (ECDH) Assumption: Let *e* be an elliptic curve and G be an additive group with order *q* which consists of the points of *e*. Let *P* be a generator of *G*, aP and bP be two elements of *G* and $\mathcal{A}$ be an ECDH-adversary with running time at most *t*. The probability that $\mathcal{A}$ succeeds in computing abP from (*aP*, *bP*) is denoted by $Adv_G^{ECDH}(\mathcal{A})$. The ECDH assumption holds if $Adv_G^{ECDH}(\mathcal{A})$ is negligible.

Theorem 1 (AKE Security) *Let D be the distribution of user's password which size is* $|D|$. *Let* $\mathcal{P}$ *be the protocol we proposed. For any adversary* $\mathcal{A}$ *running within a time bound t, with less than* q_{send} *Send queries,* q_{exe} *Execute queries and less than* q_h *Hash queries, we have:*

$$Adv_{\wp,D}^{AKE}(\mathcal{A}) < \frac{2q_{send}}{|D|} + 4q_h Adv_G^{ECDH}(t + (q_{exe} + q_{send} + 1)\tau_G) + \frac{(q_{send})^2}{q} + \frac{(q_{exe})^2}{q^2} + \frac{(q_h)^2}{2^l} + 2(q_{send} + q_{exe})Adv^E(t) \quad (1)$$

Where $Adv^E(t)$ is the probability that $\mathcal{A}$ breaks the encryption scheme. τ_G denotes the computational time of the point multiplication in group *G*.

Proof. The security analysis is based on the AKE-game between the adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. The simulator $\mathcal{S}$ initializes the system for all the users, the cloud server and the cloud service provider. We define a sequence of games starting from G_0 to G_4. For each game G_i ($0 \le i \le 4$), we define S_i be the event that $\mathcal{A}$ correctly guesses the bit *b* in the Test session and $\Pr[S_i]$ be the probability of this event. Let $D_i = |\Pr[S_i] - \Pr[S_{i-1}]|$. By using the games bellow we can get the Theorem 1.

Game G_0 : This game is in the real protocol and corresponds to the real attack. So by the definition of S_0, we have:

$$Adv_{\wp,D}^{AKE}(\mathcal{A}) = 2\Pr[S_0] - 1 \quad (2)$$

Making a transformation, we have:

$$\begin{aligned} Adv_{\wp,D}^{AKE}(\mathcal{A}) &= 2\Pr[S_0] - 1 + 2\Pr[S_4] - 2\Pr[S_4] \\ &= 2\Pr[S_4] - 1 + 2(\Pr[S_0] - \Pr[S_4]) \\ &\le 2\Pr[S_4] - 1 + 2\sum_{i=1}^{4} D_i \end{aligned} \quad (3)$$

Game G_1: In this game, $\mathcal{S}$ simulates the hash function *h* as a random oracle and creates a hash list which records the queries to *h* and the corresponding answers. The *Hash* queries, the *Send* queries, the *Execute* queries, the *Reveal* queries, the *Corrupt* queries and the *Test* query are answered as the G_0. The difference lies in G_0 is in the real protocol and G_1 is in the random oracle model. From the definition of the random oracle, we can see that G_0 and G_1 are indistinguishable. So we have:

$$D_1 = 0 \quad (4)$$

Game G_2 : $\mathcal{S}$ cancels the game when some collisions appear on the transcripts $\{aP, r_1, M_A, M_A\}$, $\{ID_{CS_i}, M_{CS_i}\}$, M_{SP}, and $\{ID_{CS_i}, bP, s_1, Auth_{SP}, Auth_{CS_i}\}$. In the *Send* queries, we can see at least one of the transcripts is generated by the honest participant. In the *Execute* queries, we can see all of them is generated by the honest participant. So by the birthday paradox, we can get the probability of collisions on the transcripts is $(q_{send})^2 / 2q + (q_{exe})^2 / 2q^2$. The same conclusion can be got in the collisions of the hash function. Then, we have:

$$D_2 \le \frac{(q_{send})^2}{2q} + \frac{(q_{exe})^2}{2q^2} + \frac{(q_h)^2}{2^{l+1}} \quad (5)$$

Game G_3: In this game, $\mathcal{S}$ simulates all the oracles in game G_2, except $\mathcal{S}$ stops the game when the adversary breaks the Encryption algorithm *E*. If the algorithm *E* is broken, then the adversary $\mathcal{A}$ can impersonate CS_i and chooses the random value *b* himself. $\mathcal{A}$ sends $\{ID_{CS_i}, bP, s_1, Auth_{SP}, Auth_{CS_i}\}$ to the user A. In such case, $\mathcal{A}$ will compute the correct session

key since he have the value *b*. $\mathcal{A}$ can also distinguish between the value returned from the Test session and a random value chosen from the key space successfully. It means $\mathcal{A}$ will distinguish the game G_2 and game G_3. Thus,

$$D_3 \le (q_{send} + q_{exe}) Adv^E(t) \tag{6}$$

Game G_4: In this game, $\mathcal{S}$ first chooses one session $\prod_A^i$ as the *Test* session and another session $\prod_{CS_i}^j$ as the matching session of the *Test* session. Then, $\mathcal{S}$ adds a random value *mP* into the *Test* session $\prod_A^i$ to instead *aP* and add another random value *nP* into the matching session $\prod_{CS_i}^j$ to instead *bP*. In such case, if $\mathcal{A}$ can successfully distinguish between the value returned from the session and a random value and win the AKE security game, then we can solve the ECDH problem using $\mathcal{A}$ as a subroutine, *i.e*., computing *mnP*. In order to obtain this conclusion, we need to use the random oracle *h*. As we know, in the random oracle model, all of the outputs of the random oracle is random. So if $\mathcal{A}$ can distinguish between the value returned from the *Test* session and a random value, it means $\mathcal{A}$ must have computed the session key himself, i.e., $h(mnP \| mP \| nP \| ID_A \| ID_{CS_i})$. It further means that $\mathcal{A}$ must have asked a *Hash* query by $(mnP, mP, nP, ID_A, ID_{CS_i})$ to the hash oracle before. By retrieving the hash list $\mathcal{S}$ kept, $\mathcal{S}$ can get the value *mnP*, i.e., solving the ECDH problem. Note here we have to show how *h* answer the query in order to correctly simulates the protocol. If a *Hash* query *x* is asked to the hash oracle (here *x* is a group of data), the simulator $\mathcal{S}$ first checks whether this query has been asked before. If it has been asked, $\mathcal{S}$ lookups the hash list and returns the corresponding answer. Otherwise, $\mathcal{S}$ chooses a random value as the answer to this hash query and returns it. Then, $\mathcal{S}$ updates the hash list with this record.

Now in G_4 the protocol is correctly simulated in the random oracle model. Suppose we let the event that $\mathcal{A}$ has asked a *Hash* query by $(mnP, mP, nP, ID_A, ID_{CS_i})$ in the *Test* session be $Event_4$. Then we can see if $Event_4$ does not happen, the advantage of $\mathcal{A}$ in winning the AKE security game in G_4 is 1/2 since the session key of the protocol in G_4 is random in the random oracle model. Thus, the probability of $\mathcal{A}$ wins the AKE-game in G_4 is:

$$\Pr[S_4] = \frac{1}{2} + \Pr[Event_4] \tag{7}$$

Next we consider the probability of $Event_4$. Actually, in game G_4, $Event_4$ will happen in the following three cases:

Case 1: $\mathcal{A}$ asks a $Corrupt(\mathrm{A}, smart\ card)$ query to the user A and obtains the secret information in the smart card, *i.e.*, $\{ID_A, C_A, r\}$. Using these secret information, $\mathcal{A}$ chooses a potential password PW_A' of A and a random value *m*, then the adversary $\mathcal{A}$ computes $K = h(m \cdot sP)$, $M_A = h(K \| r_1) \oplus ID_A$ and $X_A' = h(PW_A' \| r) \oplus C_A$. After that, the adversary $\mathcal{A}$ asks a $Send(mP, r_1, M_A, N_A)$ query to $\prod_{CS_i}^i$. $\mathcal{A}$ chooses this session as the *Test* session and asks the *Test* query. It means $\mathcal{A}$ launches the online dictionary attack. If $\mathcal{A}$'s guess is correct, then *SP* will return the message which shows $\mathcal{A}$'s authentication request can pass through. In such case, $\mathcal{A}$ will ask a *Hash* query by $(mnP, mP, nP, ID_A, ID_{CS_i})$ to the hash oracle, i.e., $Event_4$ happens. We bound the probability of this event by:

$$\Pr[case1] \le \frac{q_{send}}{|D|} \tag{8}$$

Case 2: In this case, the adversary $\mathcal{A}$ also corrupts the smart card of the user *A* and gets the secret information as in **Case 1**. Then, different from **Case 1**, $\mathcal{A}$ does not choose the random value a himself. $\mathcal{A}$ just asks $Execute(\prod_A^i, \prod_{CS_i}^i, \prod_{SP}^i)$ to $\prod_A^i$, $\prod_{CS_i}^i$ and $\prod_{SP}^i$. Then, $\mathcal{A}$ chooses this session as the *Test* session which means $\mathcal{A}$ launches an off-line dictionary attack. In such case, $\mathcal{S}$ embeds a tuple (*mP, nP*) into the protocol and substitutes *aP* with *mP* and substitutes *bP* with *nP*. If $\mathcal{A}$ wins the game, he should ask a *Hash* query by $(mnP, mP, nP, ID_A, ID_{CS_i})$, i.e., computing *mnP* without knowing *m* and *n*. In such case, $Event_4$ happens and $\mathcal{S}$ can get *mnP* and solve the ECDH problem by using $\mathcal{A}$. In this case, $Event_4$ is bounded by:

$$\Pr[case2] \le q_h \cdot Adv_G^{ECDH}(t + (q_{exe} + 1) \cdot \tau_G) \tag{9}$$

Case 3: In this case, $\mathcal{A}$ first asks an $Execute(\prod_A^i)$ query to $\prod_A^i$. Then, when gets the messages (mP, r_1, M_A, N_A) output by $\prod_A^i$, $\mathcal{A}$ continues to ask $Execute(\prod_{CS_i}^i)$) and $Execute(\prod_{SP}^i)$. When obtaining the messages $\{ID_{CS_i}, bP, s_1, Auth_{SP}, Auth_{CS_i}\}$ from $\prod_{CS}^i$, $\mathcal{A}$ does not send the messages to the user A. $\mathcal{A}$ chooses b' and s_1' himself and impersonates CS_i to send $\{ID_{CS_i}, b'P, s_1', Auth_{SP}', Auth_{CS_i}'\}$ to A. If both of $Auth_{SP}'$ and $Auth_{CS_i}'$ pass through the user's verification, then $Event_4$ will happen since $\mathcal{A}$ already knows the value b' in $b'P$. However, the probability of this event is also bounded by the advantage of breaking the ECDH problem. As shown in the protocol, mP is authenticated by $K = h(msP)$ in $Auth_{SP}$. Without knowing the *m* and *s*, $\mathcal{A}$ cannot compute correct the authentication message, *i.e.*, the random value chosen by $\mathcal{A}$ cannot pass through by the user's verification. So $Event_4$ in **Case 3** cannot happen unless $\mathcal{A}$ breaks the ECDH problem. So we have:

$$\Pr[case3] \le q_h \cdot Adv_G^{ECDH}(t + (q_{exe} + q_{send} + 1) \cdot \tau_G) \tag{10}$$

So we can bound the probability of $Event_4$ in G_4:

$$\begin{aligned}\Pr[Event_4] &\le \Pr[case1] + \Pr[case1] + \Pr[case1] \\ &\le \frac{q_{send}}{|D|} + q_h \cdot Adv_G^{ECDH}(t + (q_{exe} + 1) \cdot \tau_G) + q_h \cdot Adv_G^{ECDH}(t + (q_{exe} + q_{send} + 1) \cdot \tau_G) \\ &\le \frac{q_{send}}{|D|} + 2q_h \cdot Adv_G^{ECDH}(t + (q_{exe} + q_{send} + 1) \cdot \tau_G)\end{aligned} \tag{11}$$

Consequently from the equations (2)-(11), we can get the result of the Theorem 1.

Theorem 2 (Anonymity). *The proposed scheme provides strong anonymity against an active adversary if the ECDH problem is hard.*

Proof. The proof of the anonymity is similar to that of AKE security, so we just give a brief explanation. As the description in **Fig. 2**, if the adversary $\mathcal{A}$ wants to reveal the identity of the user A, he needs to compute the value $h(K \| r_1)$. So if $\mathcal{A}$ obtains the identity of A, it means $\mathcal{A}$ must asked a (K, r_1) query to the hash oracle, where $K = h(asP)$. Then the simulator $\mathcal{S}$ can imbed a random tuple (*mP*, *nP*) into the protocol to replace the *aP* and *sP* respectively. Then, $\mathcal{S}$ can check the hash list to find the value *mnP* if $\mathcal{A}$ obtains the identity of A. So under

the ECDH assumption, the proposed scheme can provide anonymity for the user.

Theorem 3 (Forward security). *The proposed scheme provides forward security against an active adversary if the ECDH problem is hard.*

Proof. Forward security means that if the long term secret value is corrupted by $\mathcal{A}$ he cannot recover the session key which is agreed by the honest user before this point. As we can see, the session key of the protocol is consist by the random values chosen from A and CS_i respectively. The session key is not related with the long term secret value of A, PW and C_A or the long term secret value of SP, *s*. As the proof of the AKE security, if $\mathcal{A}$ can break the forward security of the scheme, then $\mathcal{S}$ will embed a random tuple (*mP*, *nP*) into the protocol to replace the *aP* and *bP* respectively. After $\mathcal{A}$ recovers the session key, $\mathcal{S}$ will check the hash list and get the value *abP*, *i.e.*, solving the ECDH problem. The detailed proof is similar to that of AKE security and we do not repeat here.

Theorem 4 (Backward security). *The proposed scheme provides backward security against an active adversary if the ECDH problem is hard.*

Proof. The backward security means that if the long term secret value is corrupted by $\mathcal{A}$ he cannot obtain the session key which is agreed by the honest user after this point. The purpose of $\mathcal{A}$ is to obtain the secret which is encrypted by the session key between A and CS_i. Suppose that $\mathcal{A}$ has corrupted user A and gets its password and the information in the smart card without being detected by the user. Now $\mathcal{A}$ wants to obtain the session key of the user in the new session. In the new session $\mathcal{A}$ can intercept the message of A and impersonate A to communication with CS_i and SP. However, without knowing the random value *a* chosen by A, $\mathcal{A}$ cannot obtain the authentication message of SP $Auth_{SP} = h(K \| s_1 \| aP \| bP)$ where $K = h(a \cdot sP)$. So without getting the correct authentication message of SP, even if $\mathcal{A}$ impersonates CS_i and sends $\{ID_{CS_i}, bP, s_1, Auth_{SP}, Auth_{CS_i}\}$ to A, $Auth_{SP}$ cannot pass the verification of A. Therefore, $\mathcal{A}$ cannot get the session key of the new session of the user who losts all of his secret either, i.e., the scheme can provide backward security. The detailed proof is similar to that of forward security and we do not repeat here.

Table 2. The Security properties comparison between related schemes and ours

Schemes	P1	P2	P3	P4	P5	P6
Choudhury *et al.*'s	YES	NO	NO	YES	NO	NO
Chen *et a.l*'s	YES	NO	NO	NO	NO	NO
Huang *et al.*'s	YES	YES	YES	YES	YES	NO
Ours	YES	YES	YES	YES	YES	YES

P1:Mutual authentication;
P2:Providing secure key agreement;
P3:Preventing the dictionary attack;
P4:Identity protection;
P5:Secure when the smart card is lost;
P6:Forward security.

5.2 Performance Analysis

In this section we only discuss the authentication scheme using smart card in cloud computing and authentication schemes using other technical are not the main motivation of this paper. We analyze the performance of the proposed authentication scheme in two aspects: one is the

security property and the other is the efficiency. **Table 2** shows the security properties of the proposed schemes compared with some other authentication schemes for cloud computing using smart card. Actually, security is the primary question we have to answer in the cloud computing since the data are not stored in the users' computer. As far as the authentication schemes using smart card are concerned, identity protection and security when the smart card is lost and forward/backward security are important properties. From **Table 2**, we can see only our scheme has all the security properties, so from the security aspect our scheme has better performance than other schemes in the table.

Table 3. Computation cost comparison among related schemes

Schemes	Pre-computation	User	Server
Choudhury *et al.*'s	3h	1e+13h	1e+8h
Chen *et a.l*'s	2h	1e+5h+1E	1e+8h
Huang *et al.*'s	2m+1h	3h+1E	1m+4h+1E
Ours	2m+4h	1m+3h	2m+7h+4E

m:point multiplication; e:exponent;
h:hash; E:encryption/decryption.

Table 3 shows the computation cost between the proposed scheme and some related schemes. In **Table 3**, the cost of point multiplication operation is similar to that of exponent operation and the cost of hash operation is similar to that of encryption/decryption operation. Point multiplication and exponent operation are more time consuming than hash and encryption/decryption operation. From **Table 3** we can see, Huang et al.'s scheme [17] is the most efficient in the four schemes. In the user side, its computation cost is 3h+1E and in the server side its computation cost is 1m+4h+1E after pre-computation. Our authentication scheme has one more point multiplication than Huang et al.'s scheme both in the user side and the server side (here we only consider the multiplication cost since it is more time-consuming than other operations). The reason why this happens is that our scheme has the property of forward security and backward security. As we know if an authentication scheme has the forward security or backward security, one more ECDH tuple (mP, nP) will be added. So if without the forward security and the backward security our scheme has the same performance as Huang et al.'s scheme. Choudhury et al.'s scheme [9] and Chen et al.'s scheme [10] has better performance than our scheme in the server side, however, they are not as good as ours in the user side. We pay much attention to the cost of the user side since its processor speed is often limited. Meanwhile, there is a further advantage of our scheme, in our scheme the server does not need to store the ephemeral secret of the user which can reduce the probability of the attack to the server.

Table 4. Experimental data between related authentication schemes after pre-computation

Computation/Protocols	Computation-Time		
	user(PKI controller)	server(laptop)	total(one AKE)
1024 bit DH(160 bit exponent)	135ms	0.13ms	--
192 bit ECDH	25ms	0.24ms	--
AES	21us/16 byte	0.39us/16 byte	--
SHA256	1.4ms/64 byte	2.96us/64 byte	--
Choudhury *et al.*'s	156.1ms	0.13ms	156.3ms
Chen *et al.*'s	144.3ms.	0.13ms	144.5ms
Huang *et al.'s*	6.15ms	0.24ms	6.4ms
Ours	29.5ms	0.48ms	30.1ms

In order to give an objective efficiency comparison, we make efficiency analysis on the basis of the implementation of the scheme in [19] and the implementation of our scheme. The operation used in experiment is implemented on an exponent with 1024 bits prime, an ellipse curve which is over a finite field with 192 bits prime, an AES encryption with 256 bits key and a hash function SHA256. The computation cost of the user is evaluated by NXP smartMX PKI controller P5CT072 with 4.5KB RAM and PKI crypto-engine. The computational costs on the server side are evaluated using laptop with a 2.5GHz Intel Core i5-4200M processor and 4GB RAM. The communication between the smart card and the server is completed by the USB bus. **Table 4** shows the result of the experiment between related authentication schemes after pre-computation. From **Table 4** we can see, our scheme does not have the best efficiency especially in the Server side. The reason lies in that our scheme provides the forward/backward security which is absent in the other schemes. If they can provide this property, at least one more point multiplication is needed as aforementioned.

So overall consideration of the security and computation cost, our scheme has better performance in the following aspects:

(1). Comparing with the authentication schemes [9-10] using smart card in cloud computing, our scheme can resist the smart card lost attack which is important for the security of the user's data in the cloud server.

(2). Comparing with the authentication scheme [17], although both our scheme and [17] can resist the smart card lost attack, our scheme has two more advantages than [17]. Firstly, our scheme has the security property of forward and backward security which protects the session of the user in the past and in the future. Secondly, we use three-level authentication model which is different from Huang et al.'s authentication scheme [17]. It makes the authentication scheme in our scheme can be easily converted to an authentication scheme in the mobile roaming scenario since they use the same authentication model.

(3). There is a further advantage of our scheme, in our scheme the service provider does not need to store the ephemeral secret of the user which can reduce the probability of the attack to the service provider. Actually, it disperses the venture to hundreds of cloud server which avoids the case of the single point of failure. Meanwhile, it also reduces the burden of the service provider since the session key is computed between the user and different cloud servers other than the service provider.

5. Conclusion and Future Work

Authentication schemes using smart card are more practical in the real word since they can provide more convenience and strong security for the user, such as e-commerce transactions and other Internet connection activities. However, few of these scheme can resist the smart card lost attack in the cloud computing. It is more complicate when we considering more security property, such as the forward and backward security as well as the smart card lost attack. In this paper we had a close look at the authentication in the scenario of cloud computing and propose a new authentication scheme for cloud computing using smart card. The new scheme is able to address two tough problems in the smart card authentication for cloud computing: (1). the problem of smart card lost attack; (2). the problem of forward and backward security. Meanwhile, the new scheme takes advantage of the three-level authentication model which makes it easy to be converted into an authentication scheme in the roaming scenario for cloud computing since they use the same framework (in the mobile roaming cloud computing scenario, the cloud server in our framework is regarded as the foreign server or the roaming server). The distributed authentication model (i.e., the service

provider distributes the authentication to the different cloud servers) may be helpful to the researchers to design authentication scheme in multi-party authentication scenario in future. The proposed scheme has all of the security requirements in cloud authentication which are better than other schemes in this scenario. As a compromise, the computation cost in our scheme is not the best. However, it is still efficient and acceptable as shown in Table 4.

As for future work, we want to discuss more complicate authentication scenario for cloud computing, such as: (1). authentication schemes between different domains, i.e., how do the users from different cloud providers authenticate each other and agree on a common session key; (2). authentication between a group of users, i.e., how do a group of users share their own secret data in the cloud server to other users in the group securely using the smart card mechanism.

References

[1] M. Armbrust, *et al*., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp.50-58, April, 2010. Article (CrossRef Link)

[2] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1-11, January, 2011. Article (CrossRef Link)

[3] W.S. Juang, S.T. Chen, and H.T. Liaw, "Robust and efficient password authenticated key agreement using smart cards," *IEEE Transaction on Industrial Electronics*, vol. 55, no. 6, pp. 2551-2556, June, 2008. Article (CrossRef Link)

[4] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," In *Proc. of ACM Security CCS*, pp. 122-131, November 2-5, 1998. Article (CrossRef Link)

[5] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770-771, January, 1981. Article (CrossRef Link)

[6] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," In *Proc. of CT-RSA*, pp. 191-208, February 14-18, 2005. Article (CrossRef Link)

[7] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. of EUROCRYPT*, pp. 139-155, August 14-18, 2000. Article (CrossRef Link)

[8] X. Li and Y. Zhang, "A simple and robust anonymous two-factor authenticated key exchange protocol," *Security and Communication Networks*, vol. 6, no. 6, pp. 711-722, June, 2013. Article (CrossRef Link)

[9] A.J. Choudhury, *et al*., "A strong user authentication framework for cloud computing," in *Proc. of IEEE Asia-Pacific Services Computing Conference*, pp. 110-115, December 12-15, 2011. Article (CrossRef Link)

[10] N. Chen and R. Jiang, "Security analysis and improvement of user authentication framework for cloud computing," *Journal of Networks*, vol. 9, no. 1, pp. 198-203, January, 2014. Article (CrossRef Link)

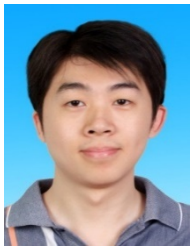
[11] R. Jiang, "Advanced secure user authentication framework for cloud computing," *International Journal of Smart Sensing and Intelligent Systems*, vol. 6, no. 4, pp. 1700-1724, September, 2013. Article (CrossRef Link)

[12] N.D. Han, *et al*., "A scheme for data confidentiality in cloud-assisted wireless body area networks," *Information Sciences*, vol. 284, no. 10, pp. 157-166, November, 2014. Article (CrossRef Link)

[13] K. Nimmy and M. Sethumadhavan, "Novel mutual authentication protocol for cloud computing using secret sharing and steganography," *Journal of Information Security Research*, vol. 5, no. 2, pp. 17-19, June, 2014. Article (CrossRef Link)

[14] Z. Hao, S. Zhong and N. Yu, "A time-bound ticket-based mutual authentication scheme for cloud computing," *International Journal of Computers, Communications & Control*, vol. 6, no. 2, pp. 227-235, June, 2011. Article (CrossRef Link)

[15] R.S. Pippal, C.D. Jaidhar and S. Tapaswi, "Enhanced time-bound ticket-based mutual authentication scheme for cloud computing," *Informatica*, vol. 37, no. 2, pp. 149-156, June, 2013. Article (CrossRef Link)
[16] T.S. Messerges, E.A. Dabbish and R.H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computing*, vol. 51, no. 5, pp. 541-552, August, 2002. Article (CrossRef Link)
[17] J.J. Huang, *et al*., "Robust and privacy protection authentication in cloud computing," *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 11, pp. 4247-4261, November, 2013.Article (CrossRef Link)
[18] T. Zhou, J. Xu, "Provable secure authentication protocol with anonymity for roaming service in global mobility networks," *Computer Networks*, vol. 55, no. 7, pp. 205-213, January, 2011. Article (CrossRef Link)
[19] T.Y. Wu and Y.M. Tseng, "An efficient user authentication and key exchange protocol for mobile client-server environment," *Computer Networks*, vol. 54, no. 9, pp. 1520-1530, June, 2010. Article (CrossRef Link)



Li Haoxing received his B.S. degree in communication engineering from Beijing Electronic Science and Technology Institute in 2004, and he received M.S. degree in electronics and communication engineering from Beihang Univercity in 2010. He is pursuing his Ph.D. degree in Xidian University. His current research interests include access control & cloud data protection.
(E-mail:lhx595@126.com).

Li Fenghua（corresponding author） received his B.S. degree, M.S. degree, and Ph.D. degree in Computer Software and Computer Systems Architecture from Xidian University in 1987, 1990, and 2009 respectively. Currently, he is working as professor and doctoral supervisor in State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. And he is also a doctoral supervisor of Xidian University. His current research interests include network security, system security & evaluation and trusted computation.
(E-mail: lfh @iie.ac.cn)

Song Chenggen received his B.S. degree, M.S. degree and Ph.D. degree communication engineering from Peking University in 2006, 2009 and 2013 respectively. Currently, He is working as an engineer in Institute of Information Security Engineering, Beijing Electronic Science Technology Institute. His current research interests include PEKS and security protocol.
(Email:songgeng87@gmail.com)

Yan Yalong received his B.S. degree communication engineering from Beijing Electronic Science Technology Institute in 2001. Currently, He is working as a senior engineer in Institute of Information Security Engineering, Beijing Electronic Science Technology Institute. His current research interests include network security and information security.
(E-mail:yalong@besti.edu.cn)