An Enhanced User Authentication Protocol Based on Elliptic Curve Cryptosystem in Cloud Computing Environment

Chenyu Wang, Ke Ding, Bin Li, Yiming Zhao, Guoai Xu, Yanhui Guo, and Ping Wang

1 School of Cyberspace Security, Beijing University of Posts and Telecommunications, China
2 School of EECS, Peking University, China
3 School of Software and Microelectronics, Peking University, China

Correspondence should be addressed to Guoai Xu; xga@bupt.edu.cn

Received 13 April 2018; Revised 24 June 2018; Accepted 12 July 2018; Published 1 October 2018

Copyright © 2018 Chenyu Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the popularity of cloud computing, information security issues in the cloud environment are becoming more and more prominent. As the first line of defense to ensure cloud computing security, user authentication has attracted extensive attention. Though considerable efforts have been paid for a secure and practical authentication scheme in cloud computing environment, most attempts ended in failure. The design of a secure and efficient user authentication scheme for cloud computing remains a challenge on the one hand and user’s smartcard or mobile devices are of limited resource; on the other hand, with the combination of cloud computing and the Internet of Things, applications in cloud environments often need to meet various security requirements and are vulnerable to more attacks. In 2018, Amin et al. proposed an enhanced user authentication scheme in cloud computing, hoping to overcome the identified security flaws of two previous schemes. However, after a scrutinization of their scheme, we revealed that it still suffers from the same attacks (such as no user anonymity, no forward secrecy, and being vulnerable to offline dictionary attack) as the two schemes they compromised. Consequently, we take the scheme of Amin et al. (2018) as a study case, we discussed the inherent reason and the corresponding solutions to authentication schemes for cloud computing environment in detail. Next, we not only proposed an enhanced secure and efficient scheme, but also explained the design rationales for a secure cloud environment protocol. Finally, we applied BAN logic and heuristic analysis to show the security of the protocol and compared our scheme with related schemes. The results manifest the superiority of our scheme.

1. Introduction

With the development of IT technology, cloud computing has become one of the hottest research directions in recent years. As a new type of service, cloud computing is rapidly integrated into our daily lives with its high scalability, high service efficiency, and low-cost charge [1]. It fundamentally changed the traditional model of service providers providing services and consumers’ access to resources: as a service provider (such as Google, Microsoft, Amazon) of cloud computing, it effectively improves the utilization of resources by centralizing the demands; consumers not only gain the convenience of using resources, but also reduce the using cost through paying on demand. Therefore, more and more big firms build their own cloud platforms and provide services, including Google App Engine, Amazon Web services, and IBM SmartCloud [2]. Furthermore, both the individuals and small companies enjoy the benefits of cloud services. Generally speaking, there are three kinds of cloud services: (1) IaaS, Infrastructure as a Service, which means providing user with the infrastructure such as storage and networks to use; (2) PaaS, Platform as a Service, which means providing user with the platform to develop various applications; (3) SaaS, Software as a Service, which means providing user with software applications [3, 4].

However, with the increasing popularity of the cloud services, the security issues have become more prominent, how to protect user privacy and restrict data from being
illegally accessed has become a challenging problem and research hotspot. The first step to solve these issues is user authentication which can verify the authenticity of communication participants. A secure user authentication scheme will firstly verify the authenticity of the user when he/she applies to access the cloud data; then to prevent a malicious cloud server trick users, the validity of the cloud server should be checked; once confirming the identity of the user and the cloud server, a session key will be established to encrypt the communication messages.

Generally speaking, there are three ways to authenticate a user, which are based on the following: (1) what you know (such as the password); (2) what you have (such as the smart card); (3) who you are (such as the biometric characteristic: fingerprint and iris). Due to its simplicity and practicality, passwords have been used more widely. While a password-based authentication protocol has natural flaws, it cannot resist against offline dictionary guessing attacks. Consequently, as a factor to help enhance security, the smart card gets used [5–9]. A scheme combined two factors (such as the password and the smart card) is called two-factor user authentication scheme. The participants of the two-factor authentication scheme in cloud computing environment involves a user, a cloud server and a register authority. Note that among the three participants, only the register authority is trusted. At first, the user and the cloud server register to the register authority, respectively. Then the cloud server will send the user a smart card with some sensitive information and negotiate a shared secret parameter with the cloud server. Later on, when the user initiates an access request to the cloud server in login phase, the three participants will authenticate themselves to each other. If they all are authenticated, the user will be allowed to access the cloud server.

Motivations. In 2013, Yang et al. [10] devoted to design a secure authentication scheme for cloud computing environment, while their scheme is vulnerable to dictionary attack. Then Yang et al. [11] proposed a new scheme; unfortunately, Chen et al. [12] then showed this scheme is not secure to insider attack and impersonation attack and proposed a new version which once again is broken by Wang et al. [13]. Wang et al. [13] pointed out that Chen et al.’s scheme [12] is subject to offline dictionary attack and impersonation attack. Most recently, Amin et al. [3] identified the security weaknesses in the schemes of Xue et al. [14] and Chuang et al. [15] by revealing the two schemes fail to provide user anonymity and forward secrecy while being not able to resist against offline dictionary attack. Therefore, they designed a new scheme that claims to overcome the security flaws of the two schemes and be secure to various attacks. However, after a scrutinization of Amin et al.’s scheme, we found their scheme still cannot overcome their identified security threats.

In these years, considerable efforts have been paid for a secure and practical authentication scheme in cloud computing environment, some typical schemes including [16–19], yet most of them are found having security flaws more or less. Designing of a secure authentication scheme for cloud computing environment is still a challenge. With the widespread use of cloud computing, the potential security threats will lead to greater harm. This unsatisfactory situation motivates us to explore the inherent reasons of the failure in those schemes, find the basic method to fix the security flaws, and design a robust and efficient user authentication protocol for cloud computing environment.

Our contributions. Amin et al.’s scheme [3] is a very typical scheme which suffers from the common attacks, while the scheme’s structure is widely accepted. So we take Amin et al.’s protocol as a study case to elaborate the common issues (and its corresponding solutions) in most authentication schemes and provide rationales for designing a secure cloud environment protocol. In addition, based on the analysis, we design a secure authentication protocol. In a short, our contributions can be summarized as follows:

1. We demonstrated that Amin et al.’s scheme [3] fails to achieve user anonymity and forward secrecy while being not able to resist against offline dictionary attack.

2. We discussed the inherent reasons of the identified flaws and its corresponding solutions; furthermore, we realized the way of deploying a public key algorithm rightly is challenging. Therefore, we showed the essential points for deploying public key algorithms.

3. We improved Amin et al.’s scheme from security and effectiveness two aspects, proved the security of our scheme via BAN logic and heuristic analysis and, finally, compared our scheme with other related schemes. The results show that our scheme is more suitable for cloud computing environment.

The remainder of this paper is organized as follows: Section 2 sketches complexity assumptions and extends adversary model; then, Amin et al.’s scheme is reviewed and analyzed in Section 3; in Section 4, we propose a secure scheme and elaborate on design rationales; Section 5 proves the security of our scheme; in Section 6, we compare our scheme with other related schemes; finally, the conclusion is drawn in Section 7.

2. Preliminary

This section introduces the preliminary of the whole paper, including complexity assumptions in designing a scheme and some notations and abbreviations.

2.1. Computational Problems. Given two large primes \( p \) and \( q \), let \( \mathbb{F}_p^* \) be a finite field, \( E/\mathbb{F}_p \) be an elliptic curve over \( \mathbb{F}_p \), and \( G \) be a \( q \)-order subgroup of \( E/\mathbb{F}_p \). For \( \alpha, \beta \in \mathbb{Z}_p^* \) and a point \( P \) in \( G \), we can define the discrete logarithm problem as follows:

1. Elliptic curve discrete logarithm (ECDL) problem: given \( (P, \alpha P) \), it is impossible to compute \( \alpha \) within polynomial time.

2. Elliptic curve computational Diffie-Hellman (ECCDH) problem: given \( (\alpha P, \beta P) \), it is impossible to compute \( \alpha \beta P \) within polynomial time.

2.2. Adversary Models. Understanding the adversary models is the most basic step to design and analyze a protocol. In
2015, Wang et al. [20] proposed the capabilities of adversary in distributed systems: (1) exhaust passwords and identities; (2) learn ID\_i when evaluating security strength; (3) control of the open communication channel; (4) learn PW\_i or extract information in the smart card; (5) acquire previous session keys; (6) know the long-term secret key x when considering forward secrecy. As both the distributed systems and the cloud computing systems have similar network environment, their adversary models are also similar too. Therefore, we adopt Wang et al’s adversary models [20] which have been accepted by various schemes [21–23].

2.3. Notations and Abbreviations. As shown in Table 1, we summarize the notations and abbreviations used in this paper.

### 3. Cryptanalysis of Amin et al.’s Scheme

After identifying the security pitfalls in other two user authentication schemes, Amin et al. [3] attempted to design a new light weight protocol in cloud computing environment. After analyzing their scheme using AVISPA tool, they claimed the new scheme achieves forward security while being resistant to various attacks. However, this section will show that, under the assumptions on adversary capabilities in Section 2.2, their scheme cannot provide forward security while being subject to two kinds of offline dictionary attacks [27] and so on. Thus their scheme is not a truly two-factor scheme. To address these issues, this section first reviews the scheme of Amin et al. and then analyzes Amin et al’s scheme [3].

#### 3.1. Review of Amin et al’s Scheme

This section briefly reviews the scheme of Amin et al. [3]; their scheme consists of five phases. As the password change phase and identity update phase have little relevance, we omit them. Furthermore, we adjust some symbols of their scheme for the ease of reading and the unity of the paper.

### Registration Phase

**(1) Cloud Server Registration Phase**

Step 1. CS\_j \rightarrow RA: [SID\_j, d]. d is a random number.

Step 2. RA \rightarrow CS\_j: \{Skey\_j\}, RA computes PSID\_j = h(SID\_j ∥ d), Skey\_j = h(PSID\_j ∥ y).

Step 3. CS\_j keeps \{Skey\_j, d\}.

**(2) User Registration Phase**

Step 1. U\_i \rightarrow RA: \{A\_i, PID\_i\}. A\_i = h(PW\_i ∥ b\_1), PID\_i = h(ID\_i ∥ b\_2), bb\_i = b\_2 ∥ A\_i, where b\_1, b\_2 are two random number chosen by U\_i.

Step 2. RA \rightarrow U\_j: smart card \{C\_j, E\_j, bb\_j, DP, h()\}. RA calculates C\_j = h(A\_i ∥ PID\_i), D\_j = h(PID\_j ∥ x), E\_j = D\_j ∥ A\_i.

Step 3. U\_i inputs DP, bb\_j into the card where DP = h(ID\_i ∥ PW\_i) ∥ b\_1.

### Login and Authentication Phase

**Step 1. U\_i \rightarrow CS\_j: \{G\_i, F\_i, Z\_i, PID\_i, TS\_j\}.** After U\_i inputs ID\_i and PW\_i, the card computes b\_i = DP ∥ h(ID\_i ∥ PW\_i), A\_i = h(PW\_i ∥ b\_i), b\_2 = b\_2 ∥ A\_i, PID\_i = h(ID\_i ∥ b\_2), C\_j = h(A\_i ∥ PID\_i). If \( C\_j = \neq C\_j \), the card produces a random number N\_i (128 bit) and computes D\_j = E\_j ∥ A\_i, G\_i = h(PID\_i ∥ SID\_j ∥ N\_i ∥ TS\_j ∥ D\_j), F\_i = D\_j ∥ N\_i, Z\_i = SID\_j ∥ h(D\_j ∥ N\_i).

**Step 2. CS\_j \rightarrow RA: \{J\_i, K\_i, PSID\_i, G\_i, F\_i, Z\_i, PID\_i, TS\_i, TS\_j\}.** CS\_j first checks \( TS\_j - TS\_i < ΔT \), then selects a 128 bit random number N\_j, and computes I\_j = Skey\_j ∥ N\_j, K\_i = h(N\_i ∥ Skey\_j ∥ G\_i ∥ TS\_j).

#### Table 1: Notations and abbreviations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_i</td>
<td>i\textsuperscript{th} user</td>
</tr>
<tr>
<td>CS_j</td>
<td>j\textsuperscript{th} cloud server</td>
</tr>
<tr>
<td>RA</td>
<td>the register authority</td>
</tr>
<tr>
<td>x</td>
<td>the adversary</td>
</tr>
<tr>
<td>y</td>
<td>the long term secret key of RA</td>
</tr>
<tr>
<td>ID_j</td>
<td>identity of U_i</td>
</tr>
<tr>
<td>PW_i</td>
<td>password of U_i</td>
</tr>
<tr>
<td>SID_j</td>
<td>identity of CS_j</td>
</tr>
<tr>
<td>Skey_j</td>
<td>shared key between G_i and CS_j</td>
</tr>
<tr>
<td>⊕</td>
<td>bitwise XOR operation</td>
</tr>
<tr>
<td>‖</td>
<td>concatenation operation</td>
</tr>
<tr>
<td>h(⋅)</td>
<td>one-way hash function</td>
</tr>
<tr>
<td>⇒</td>
<td>a secure channel</td>
</tr>
</tbody>
</table>
Step 4. RA → CS; \{P_{cs}, R_{cs}, Q_{cs}, V_{cs}\}. RA checks (TS_{cs} < \Delta T^*) and computes \(D_i = h(PID_i \parallel \mathit{A})\), \(N^*_j = F_i \oplus D_i\), \(SID_j = Z_j \oplus h(D_i \parallel N^*_j)\), \(G^*_i = h(PID_i \parallel SID_j \parallel N^*_j \parallel D_i \parallel TS_j)\).

If (\(G^*_i \neq G_i\)), compute \(\text{Skey}^*_i = h(PSID_j \parallel y)\), \(N^*_j = \text{Skey}^*_i \oplus K^*_i = h(\text{Skey}^*_i \parallel N^*_i \parallel G_i \parallel TS_j)\).

If (\(K^*_i = K_i\)), compute \(P_{cs} = N_i \oplus N_{cs} \oplus h(N_j \parallel D_i), R_{cs} = \{N_j \oplus N_{cs} \oplus h(\text{Skey}^*_i \parallel N^*_j)\}, SK_i = h(N_j \oplus N_{cs} \oplus N_i), Q_{cs} = h((N_j \oplus N_{cs} \parallel SK_i)\), V_{cs} = h((N_i \parallel N_{cs}) \parallel SK_i)\). If (\(V_{cs} \neq V_{cs}\)), send \(\{P_{cs}, Q_{cs}\}\) to \(U_i\).

Step 5. \(U_i\) computes \(L_i = h(N_i \parallel D_i), N_j \oplus N_{cs} = P_{cs} \oplus L_i, SK_j = h(N_j \oplus N_{cs} \parallel N_i), Q_{cs}^* = h((N_j \parallel N_{cs}) \parallel SK_j)\). If (\(Q_{cs}^* = Q_{cs}\)), the whole authentication phase finishes successfully.

3.2. Cryptanalysis of Amin et al.’s Scheme. Amin et al.’s scheme [3] does not point out the adversarial model, while, according to their attack on the scheme of Xue et al. [14] and Chuang et al. [15], we can infer their adversary model which is included into our model (see Section 2.2). Although Amin et al.’s scheme [3] provides many admirable features, such as changing password locally and high efficiency, it still suffers from various attacks like most authentication protocols in cloud computing environment. Therefore, their scheme is a typical case to show the security threat in cloud environment. Through Amin et al.’s scheme, we can get insight into the inherent reasons of the failure in other authentication protocols for cloud and, based on it, learn to design a secure one. In brief, this section, on one hand, demonstrates that Amin et al.’s scheme [3] is vulnerable to various attacks and, on the other hand, indicates the failure reasons of their scheme.

Off-Line Dictionary Attack I. (i) The adversary’s capability: it obtains the message \(\{C_i, E_i, b_b, DP\}\) in \(U_i\)’s smart card.

(ii) The attack steps: the steps are as follows.

Step 1. Guess PW\(_i\) to be \(PW^*_i, ID_i\) to be \(ID^*_i\). Note that, it is quite realistic for \(A\) to obtain the password and identity simultaneously, because their spaces are limited [28].

Step 2. Compute \(b^*_i = DP \oplus h(ID^*_i \parallel PW^*_i)\).

Step 3. Compute \(A^*_i = h(PW^*_i \parallel b^*_i)\).

Step 4. Compute \(b^*_2 = b_b \oplus A^*_i\).

Step 5. Compute \(PID^*_i = h(ID^*_i \parallel b^*_2)\).

Step 6. Compute \(C^*_i = h(A^*_i \parallel PID^*_i)\).

Step 7. Verify the correctness of \(PW^*_i\) and \(ID_i\) by checking if \(C^*_i = C_i\).

Step 8. Repeat steps 1 ∼ 6 until the correct values of \(PW^*_i\) and \(ID_i\) are found.

(iii) The time complexity: \(O(|D_{pw}| \ast |D_{id}| \ast 4T_H)\), where \(T_H\) is the time of hash-function.

Remark. Generally speaking, achieving two-factor security is the most essential requirement of a two-factor authentication protocol; that is, any one of the factors being broken will not trigger the security of another factor, which in turn threatens the entire system. In recent years, many protocols have tried to propose a secure two-factor security protocol, but most have failed. It was not until the work of Ma et al. [29] and Wang et al. [20, 30] did such a stagnant situation completely changed. In 2012, Ma et al. [29] pointed out that public key algorithm is necessary to design a secure two-factor authentication scheme; in 2015, Wang et al. [20] found that there is a conflict between changing password locally and resisting against smart card loss attack under the current technique; therefore, Wang et al. [30] put forward a way of “honeywords”+“fuzzy-verifier” to solve the conflict; in 2016, Wang et al. [27] further pointed out that there are two offline dictionary attacks and then combined with the results of [29, 30] and matched the corresponding solutions for each attack.

In this paper, we follow the classification method of Wang et al. [27] and demonstrate that Amin et al.’s scheme [3] cannot resist against the two kinds of dictionary attack. Looking back at the above attack process, we can find that the key to the problem is that \(A\) can find the verification value \(C_i\) to check the correctness of the guessed result. According to Wang et al. [30], this issue can be settled with the integration of “fuzzy-verifier” and “honeywords”; let \(C_i = h(A_i \parallel PID_i \parallel D_i) \mod n_0\) where \(n_0\) is an integer \((2^* \leq n_0 \leq 2^8)\). The detailed explanation on this method can be found in Section IV of [30] or Section 5.2 of this paper.

Off-Line Dictionary Attack II. (i) The adversary’s capability: (1) it eavesdrops on one of \(U_i\)’s login requests \(\{G_i, F_i, Z_i, PID_i, TS_j\}\) and (2) obtains the message \(\{C_i, E_i, b_b, DP\}\) in \(U_i\)’s smart card.

(ii) The attack steps: the steps are as follows.

Step 1. Guess PW\(_i\) to be \(PW^*_i, ID_i\) to be \(ID^*_i\).

Step 2. Compute \(b^*_i = DP \oplus h(ID^*_i \parallel PW^*_i)\).

Step 3. Compute \(A^*_i = h(PW^*_i \parallel b^*_i)\).

Step 4. Compute \(D^*_i = E_i \oplus A^*_i\).

Step 5. Compute \(N^*_i = F_i \parallel D^*_i\).

Step 6. Compute \(G^*_i = h(PID_i \parallel SID_j \parallel N^*_i \parallel TS_j \parallel D^*_i)\). Note that Amin et al. [3] view SID\(_j\) as a secret only known to the legitimate user. However, it not practical: \(A\) at least can register as a legitimate user to get SID\(_j\).

Step 7. Verify the correctness of \(PW^*_i\) and \(ID_i\) by checking if \(G^*_i = G_i\).

Step 8. Repeat steps 1 ∼ 6 until the correct values of \(PW^*_i\) and \(ID_i\) are found.

(iii) The time complexity: \(O(|D_{pw}| \ast |D_{id}| \ast 3T_H)\).
Remark. In this attack, the pivotal parameter is $G_i$. To adversary $\mathcal{A}$, the only challenge in computing $G_i$ is the value of $D_i$, which can be derived from $(ID_i, PW_i)$, so once $\mathcal{A}$ guesses the value of $(ID_i, PW_i)$, he/she can check the correctness of them via $G_i$. Now considering a situation where $G_i$ consists of the secret shared $D_i$ and an another nonpublic dynamic parameter which should not be derived from $(ID_i, PW_i)$. In this situation, $\mathcal{A}$ cannot use $G_i$ to check the guessed value anymore, since there is another uncertain parameter besides $D_i$. Consequently, constructing such a dynamic parameter which is known to $U_i$ and RA is our critical step to address this attack. Taking into account the fact that Ma et al.’s emphasizes [29] on the necessity of lightweight public-key algorithm in designing a secure authentication protocol, we then apply a lightweight public-key algorithm to construct such a dynamic parameter. In addition, our specific ideas on solving this attack are shown in Section 4.

User anonymity: these days user anonymity has become one of the security issues that people are widely concerned about, especially in the case of cloud computing and the Internet of Things that involve massive data. The adversary can acquire people’s sensitive personal information via various ways including analyzing the session transcript in the open channel when the services are accessed [31]. Moreover, with the development of the technology, the adversary may even trace users’ movement and learn the location of their home or company, which triggers a huge potential threat [32]. Under these circumstances, user anonymity is a pivotal attribute of the authentication scheme to protect user privacy.

Generally speaking, user anonymity covers two aspects [31]: (1) user identity protection; (2) user untraceability. The former requires the scheme does not expose users’ identity; and the latter prevents an adversary from linking the session transcripts to a specific user or distinguishes the sessions sent by different users. This definition on user anonymity is widely applied in most authentication schemes [8, 24, 33, 34]. Unfortunately, in Amin et al.’s scheme, the parameter $PID_i$ that identifies the user identity is a static value exposed in the insure channel, which means the adversary can trace $U_i$ via $PID_i$. Consequently, Amin et al.’s scheme cannot provide user anonymity. As we can see, one of the keys to achieve user anonymity is concealing the real identity with a dynamic parameter. The way to implement this is called dynamic-ID technique [20]. According to Wang et al.’s suggestion [31], we can employ the dynamic-ID technique to protect user anonymity via applying a lightweight public-key algorithm to the authentication schemes as described in Section 4.

Forward Secrecy. (i) The adversary’s capability: (1) it eavesdrops on $\{F_i, PID_i\}$ and $\{P_{\omega}\}$ and (2) obtains the secret long-term key $x$.

(ii) The attack steps: the steps are as follows.

Step 1. Compute $D^*_i = h(PID_i \parallel x)$.

Step 2. Compute $N^*_i = F_i \oplus D^*_i$.

Step 3. Compute $L^*_i = h(N^*_i \parallel D^*_i)$.

Step 4. Compute $(N_j \oplus N_{e_j})^* = L^*_i \oplus P_c$.

Step 5. Compute $SK = h((N_j \oplus N_{e_j})^*) \oplus N^*_i$.

(iii) The time complexity: $O(|\mathcal{D}_{pu}| \times |\mathcal{D}_{id}| \times 3T_{H})$.

Remark. In Amin et al.’s scheme [3], there are two secret keys ($x, y$) in the register authority. We have shown that the leakage of the long-term key $x$ will lead to the exposure of previous sessions key. In the following attack, we can see that the leakage of $y$ leading to the same question too. As a result, the use of two system parameters is of little significance but consumes resource.

(i) The adversary’s capability: (1) it eavesdrops on $\{I_j, PSID_i\}$ and $\{R_{\omega}\}$ and (2) obtains the secret key $y$.

(ii) The attack steps: the steps are as follows.

Step 1. Compute $Skey_j^* = h(PID_j \parallel y)$.

Step 2. Compute $N^*_j = I_j \oplus Skey_j^*$.

Step 3. Compute $W^*_j = h(Skey_j^* \parallel N_j)$.

Step 4. Compute $(N_i \oplus N_{e_j})^* = W^*_j \oplus R_{\omega}$.

Step 5. Compute $SK = h(N_j \oplus (N_i \oplus N_{e_j})^*)$.

(iii) The time complexity: $O(|\mathcal{D}_{pu}| \times |\mathcal{D}_{id}| \times 3T_{H})$.

Remark. When considering the forward secrecy, the adversary almost has the same capacity with RA except that $\mathcal{A}$ does not know the verifier-table. As a result, if RA can compute the session key according to the processes of the scheme, then $\mathcal{A}$ is very likely to break the session key. For the above considerations, we do not recommend that RA have the ability to calculate session keys. To achieve this, a public-key algorithm is suggested too [29]. In addition, the more concrete improved methods will be explained in Section 4.

Other flaws: using timestamps to resist replay attack is not recommended. As we all know, due to the network congestion, network latency, or other issues, maintaining a consistent network clock between different systems is very difficult, which often results in the desynchronization attacks. As a matter of fact, many papers [20, 30] in their evaluation criteria pointed out that a protocol using timestamps cannot resist against desynchronization attacks. Furthermore, determining an appropriate value of $\Delta t$ always faces many challenges in practice: if this value is too big, a replay attack occurs; if it is too small, a valid participant may be stopped. Therefore, in protocol design, the use of random numbers is usually a more recommended way. Unfortunately, the timestamps method was applied in Amin et al.’s scheme.

Insecure identity update phase: similar to the process in “offline dictionary attack II” of Section 3.2, an adversary can carry out an offline dictionary attack via using $DD_j$ or $DD_i$ as the verification parameter when $U_i$ tries to update his/her identity.
4. The Proposed Scheme

In this section, we design a secure, simple, and efficient user authentication scheme for cloud computing environment (as shown in Figures 1 and 2) which overcomes all the flaws of Amin et al.'s scheme [3] but provides more attractive attributes, such as updating password and identity and user reregistering with same identity. As a matter of fact, we improve Amin et al.'s scheme from two aspects, efficiency and security. The design rationales are sketched as follows.

(i) Improvements in Security. According to our discussion in Section 3.2, the public-key algorithm is indispensable to a secure two-factor user authentication scheme [20, 29, 31]. As a matter of fact, this theory has been widely accepted by massive new authentication schemes [26, 30, 35, 36], while the main difficulty lies in deploying the public key algorithm properly. Consequently, we will show the subtleties of deploying a public key algorithm in detail as follows.

Note that, as its high efficiency, the elliptic curve cryptosystem has been used widely in authentication schemes [21, 37–39]. Therefore, our scheme deploys the elliptic curve algorithm to achieve our secure authentication scheme. Under this circumstance, compared with Amin et al.'s scheme [3], our scheme adds a parameter initialization process to set the parameters of elliptic curve cryptosystem.

(1) Apply public-key algorithm to resist against offline dictionary guessing attack II. In Section 3.2, we have shown the pivotal point in addressing such an attack is to add a dynamic nonpublic parameter in \( G_i \). Here we show the ideas about the way of achieving that, to \( U_i \) and \( RA \), it is easy to compute \( X_1 \), yet to \( A \), there is another uncertain parameter in \( G_i \), which stops \( A \) carrying out the offline dictionary attack II.

(2) Apply public-key algorithm to provide user anonymity. With the help of \( X_1 \), we conceal the identity related parameter \( PID_i \) in \( F_i \) as \( PID_i \oplus h(X_1) \). Since \( F_i \)
is changed with $N_i$, user anonymity is provided. Note that $X_i$ is the dynamic parameter we mention in “user anonymity” of Section 3.2.

(3) Apply public-key algorithm to achieve forward secrecy. We set the ECCDH problem in session key to achieve forward secrecy as follows: let SK consist of $X_0/Y_0$ where $X_0/Y_0 = N_i/N_jP = N_iX_0, Y_0 = N_jP$. As a result, to the one (including RA) who does not know $N_i/N_j$, computing SK is equivalent to solve the ECCDH problem which cannot be solve in the polynomial time. Therefore, the forward secrecy is achieved.

(4) Following Wang et al.’s way [30] of resisting against offline dictionary attack I, as the detailed explanation on this method can be found in Section IV of [30], Section 6.2.4 of [24], or Section 5.2 of this paper, we do not repeat here.

(ii) Improvements in Efficiency. Note that Amin et al.'s scheme [3] only involves some hash operation, while our scheme deploys a public-key algorithm, so the performance of our scheme is certainly not as efficient as Amin et al.'s scheme. However, except the increased cost of the public-key algorithm, we try to optimize other aspects of the performance in Amin et al.'s scheme through reducing unnecessary parameters or calculations as follows:

(1) Reduce the number of random numbers selected by $U_i$ to one during the user registration process. In Amin et al.'s scheme, there are two random numbers in the smart card. While they actually can derive from each other, in addition, the “ability” of computing them is the same, which means they are “equivalent”. As a result, using one random number is enough, which saves the storage space and computing resources.

(2) Reduce the number of secret keys in RA. As we see in Section 3.2, the secret key $y$ is of little effect in improving the security. Furthermore, it makes the register phase of $CS_j$ and the authentication of $CS_j$ more complex. Therefore, we only set one system secret parameter and simplify the register phase of $CS_j$. Such changes also bring other improvements on computing performance.

4.1. Registration. RA selects two large primes $\{p, q\}$ and a medium integer $n_0 (2^k \leq n_0 \leq 2^k)$. Let $\mathbb{F}_p$ be a finite field, $E/\mathbb{F}_p$ be an elliptic curve over $\mathbb{F}_p$, and $G$ be a $q$-order subgroup of $E/\mathbb{F}_p$, then RA chooses a point $P$ in $G$ and a long-term secret key $x \in Z^*_p$ and computes its public key $Pub$ as $xP$. In our cloud computing environment, the cloud server and the user registration phases are conducted as follows.

For the Cloud Server $CS_j$

Step 1. $CS_j \implies RA: \{SID\}_j$.

Step 2. $RA \implies CS_j; \text{Skey}_j = h(\text{SID}_j \parallel x)$.

Step 3. $CS_j$ stores $\text{Skey}_j$ as a secret key.

For the User $U_i$

Step 1. $U_i \implies RA: \{A_i, PID_i\}$. A new user $U_i$ firstly selects the password $PW_i$, identity $ID_i$, and a random number $b_i$ as his/her personal information and then computes the registration parameters as follows: $A_i = h(PW_i \parallel b_i), PID_i = h(ID_i \parallel b_i)$, and it initiates the registration phase via submitting the request $\{A_i, PID_i\}$ to RA.

Step 2. $RA \implies U_i$: a smart card with $\{C_i, E_i, Pub, n_0, h(\cdot)\}$. To guarantee the uniqueness of user identity, RA will firstly check whether the PID has been used via traverse user – list. If it is available, RA chooses a unique random number $r_i$ for $U_i$ and computes $D_i = h(PID_i \parallel x \parallel r_i), C_i = h(A_i \parallel PID_i \parallel D_i) \mod n_0, E_i = D_i \oplus A_i$, then inputs $U_i$ related parameters $\{PID_i, r_i, \text{HoneyList} = 0\}$ into User – list. Note that $\text{HoneyList}$ will record the number of login failures. Finally RA accepts $U_i$’s registration through issuing him/she a smart card with $\{C_i, E_i, Pub, n_0, h(\cdot)\}$.

Step 3. $U_i$ inputs DP into the card where $DP = h(ID_i \parallel PW_i) \oplus b_i$.

4.2. Login phase. As shown in Figure 1, if the user wants to access a cloud server, $U_i$ will submit the information to prove his/her legitimacy in login phase. If the smart card has verified $U_i$’s legality, it initiates the access request to RA for $U_i$. In short, our login phase involves two aspects: (1) verifying the validity of $U_i$; (2) initiating the access request.

Step 1. $U_i \rightarrow CS_j$: login request $\{G_i, F_i, Z_i, X_2\}$. $U_i$ enters $\{ID_i, PW_i\}$, then the smart card computes $b'_i = DPb_i h(ID_i \parallel PW_i), A'_i = h(PW_i \parallel b'_i), PID'_i = h(ID_i \parallel b'_i), D'_i = E_i \oplus A'_i, C'_i = h(A'_i \parallel PID'_i \parallel D'_i) \mod n_0$. The card compares $C'_i$ with the stored $C_i$ to verify the valid $U_i$. If $C'_i \neq C_i$, exit the session.

Step 2. Otherwise, the card accepts $U_i$’s legitimacy and initiates an access request for $U_i$: select a random number $N_i \in [1, q - 1)$, computes $X_1 = N_iPub, X_2 = N_iP, G_i = (PID_i \parallel SID_i \parallel X_1 \parallel D'_i), F_i = PID'_i \oplus h(X_1 \parallel X_2), Z_i = SID_i \oplus h(D'_i \parallel X_2)$, finally transmits $\{G_i, F_i, Z_i, X_2\}$ to $CS_j$.

4.3. Authentication Phase. Once getting the access request, the $CS_j$ will do some calculation to embed its unique parameters and transmit the request to RA for it is unable to check the authenticity of the request. Then RA will check the validity of the user and the cloud server, respectively, and help them to negotiate the session key. The whole authentication steps are as follows.

Step 1. $CS_j \rightarrow RA: \{Y_2, K_i, \text{PID}_j, G_i, F_i, Z_i, X_2\}$. $CS_j$ selects a random number $N_j$, computes $Y_1 = N_jPub, Y_2 = N_jP, PID_j = SID_j \oplus Y_1, K_i = h(Y_1 \parallel \text{Skey}_j \parallel G_i \parallel X_2 \parallel Z_i)$, then it sends $\{Y_2, K_i, \text{PID}_j, G_i, F_i, Z_i, X_2\}$ to RA.
Step 2. RA → CS_j: \{V_{cs}, Q_{cs}\}. RA will firstly verify U_i via computing the predefined shared parameter D_i and potential shared secret parameters \(X_1^i = xX_2, PID_j^i = f_j \oplus h(X_1^i), D_j^i = h(PID_j^i \oplus x \oplus r_j)\) where \(r_j\) is retrieved from the User – list through PID_j^i, SID_j^i = Z_i \oplus h(D_j^i \oplus X_1^i). If U_i's Hoeny – List exceeds a predetermined secure value or the received G_i \# (PID_j^i || SID_j^i || X_1^i || D_j^i), RA views U_i as an adversary and let Hoeny – List = Hoeny – List + 1 (if it exceeds the preset value, RA will suspend the card till U_i reregisters), then rejects the request.

Otherwise, RA will continue to verify the authenticity of the cloud server as follows: calculate \(Y_{1}^{\prime \prime} = xY_2, SID_j^{\prime \prime} = PSID_j \oplus Y_1.\) If \(SID_j^{\prime \prime} \neq SID_j\), exit (RA thinks CS_j is not the server which U_i actually desires to access); otherwise, continue to compute \(SK_j^{\prime \prime} = h(SID_j^{\prime \prime} \oplus x), K_i^{\prime \prime} = h(Y_{1}^{\prime \prime} \oplus SK_j^{\prime \prime}) \oplus h(Y_{1}^{\prime \prime} \oplus SK_j^{\prime \prime}) = G_j \oplus X_2 \oplus Z_i).\) Finally, RA tests the authenticity of CS_j by checking whether \(K_i^{\prime \prime} = K_i.\) If they are not equal, CS_j does not pass RA's authentication, the session will be terminated.

Otherwise, RA authenticates CS_j and then help them to establish the session key as follows: computes \(V_{cs} = h(Y_{1}^{\prime \prime} \oplus Y_2 \oplus SK_j^{\prime \prime} \oplus X_2), Q_{cs} = h(X_1^i \oplus X_2 \oplus D_j || Y_2),\) then responds \{V_{cs}, Q_{cs}\} to CS_j.

Note that V_{cs} and Q_{cs} are used to CS_j and U_i, respectively. They are used to verify the authenticity of RA and then convince CS_j/U_i that X_2/Y_2 is truly generated by U_i/CS_j.

Step 3. CS_j → U_i: \{Y_2, Q_{cs}\}. CS_j computes \(V_{cs}^{\prime} = h(Y_1^i \oplus Y_2 \oplus SK_j^{\prime \prime} \oplus X_2)\), then checks whether \(V_{cs}^{\prime} = V_{cs}.\) If the equation is not satisfied, CS_j rejects U_i’s request. Otherwise, CS_j believes that U_i is a legitimate user who generates X_2, then CS_j computes the session key as \(S_K_j = h(Y_2 \oplus X_2 \oplus Y_2)\) where Y_3 = N_j \cdot X_2. Finally, CS_j transmits \{Y_2, Q_{cs}\} to U_i.

Step 4. On receiving this message, U_i will firstly check the valid of RA via comparing \(h(X_1^i \oplus X_2 \oplus D_j^i || Y_2)\) with the received Q_{cs}. If they are equal, U_i trusts RA and CS_j, then he/she computes their shared session key as \(X_3^i = N_j \cdot Y_2 = N_j \cdot (N_j \cdot P) = Y_3, SK_j = h(Y_2 \oplus X_2 \oplus Y_2).\) Till now, the authentication phase finishes successfully.

4.4 Password Change Phase. Considering the security and user friendliness, our password change phase is conducted locally, which guarantees the efficiency. In other words, the user can change his/her password freely even when he/she does not connect the Internet. All in all, our password change phase is performed as follows.

Step 1. U_i enters ID_{new}, PW_{new}, and new password PW_{new}.

Step 2. The smart card verified U_i’s authenticity as step 1 of Section 4.2. If U_i is authenticated, the card will carry out the password change process as step 3; otherwise, the card will reject this request.

Step 3. The smart card computes some new parameters for new password as follows: \(A_{i}^{\text{new}} = h(PW_{i}^{\text{new}} \oplus b_i^i), E_{i}^{\text{new}} = E_i \oplus A_i^i \oplus D_{i}^{\text{new}}, D_{i}^{\text{new}} = h(ID_i \oplus PW_{i}^{\text{new}}) \oplus b_i^i, C_{i}^{\text{new}} = h(A_i^i \oplus PID_i \oplus D_i \oplus D_{i}^{\text{new}}) \mod n_0.\) Note that b_i^i, A_i^i, D_i, PID_i^i is acquired in step 2. Finally, replace \{C_i, E_i, DP\} with \{C_{i}^{\text{new}}, E_{i}^{\text{new}}, D_{i}^{\text{new}}\}.

4.5. Identity Update Phase. Considering the following occasions, a user sets the phone number as his/her identity, then when the phone number is changed, the user may also want to update the identity. Therefore, similar to password change, the user also needs to change the identity in practice, although it happens less often. Thus we provide the identity update phase as follows.

Step 1. U_i enters ID_{new}, PW_{i}^{\text{new}}, and new password ID_{i}^{\text{new}}.

Step 2. The smart card verified U_i’s authenticity as step 1 of Section 4.2. If U_i is not authenticated, the card will reject the request; otherwise, the identity update process proceeds.

Step 3. As the PID_i stored in RA is related to ID_i, the identity update phase involves the interaction with RA. On this occasion, the smart card submits \{X_2, G_j, F_j, NPID_i\} to RA for requesting update identity, where \(N_j\) is a random number, \(X_1 = N_j \cdot P \oplus D, X_2 = N_j \cdot P, G_j = h(PID_i \oplus X_1 \oplus D_j \oplus PID_{\text{new}}, F_j = PID_i \oplus h(X_1 \oplus X_2), PID_{\text{new}} = h(ID_{\text{new}} \oplus b_i^i), NPID_i = PID_i \oplus h(X_1 \oplus X_2)\).

Step 4. RA shall verify the valid of U_i as follows: compute \(X_1 = xX_2, PID_j^i = f_j \oplus h(X_1^i), D_j^i = h(PID_j^i \oplus x \oplus r_i)\) where \(r_i\) is from User – list, PID_{\text{new}} = NPID_i \oplus h(X_1 \oplus X_2). If the Hoeny – List exceeds a predetermined value or the received G_i \# (PID_j^i || X_1^i || D_j^i || PID_{\text{new}}), RA rejects the request and sets Hoeny – List = Hoeny – List + 1. Once it exceeds the preset value, suspend the card.

Otherwise, RA updates PID_i with PID_{\text{new}} in User – list and sends \{M_{cs}, ND_i\} to U_i where \(M_{cs} = h(PID_{\text{new}} \oplus PID_j^i \oplus D_j^i \oplus D_{\text{new}} = h(PID_{\text{new}} \oplus X_2)\) and \(ND_i = D_{\text{new}} \oplus h(X_1^i)\).

Step 5. After receiving M_{cs} from RA, the smart card authenticates RA via testing \(M_{cs} = h(PID_{\text{new}} \oplus PID_j^i \oplus D_j^i \oplus D_{\text{new}} = h(PID_{\text{new}} \oplus X_2)\) and the equation does not hold, exit the session; otherwise, finish the identity update process: \(D_{\text{new}} = h(ID_{\text{new}} \oplus PW_{i}^{\text{new}}) \oplus b_i^i, D_{\text{new}} = ND_i \oplus h(X_1^i), E_{i}^{\text{new}} = D_{\text{new}} \oplus A_i^i \oplus D_j, C_{i}^{\text{new}} = h(A_i^i \oplus PID_i \oplus D_i \oplus D_{\text{new}}) \mod n_0.\) Finally, replace \{C_i, E_i, DP\} with \{C_{i}^{\text{new}}, E_{i}^{\text{new}}, D_{i}^{\text{new}}\}.

4.6. Re-Register Phase. If U_i’s smart card is suspended, then he/she shall reregister to RA:

Step 1. U_i → RA: \{A_i, PID_i, re – register\}.

Step 2. RA firstly finds PID_i in User – list and checks whether U_i’s card is suspended. If so, RA accepts the request and performs the register phase as Section 4.1.
5. Security Analysis

In this section, we analyze the security of our scheme via two popular methods. The results demonstrate that our protocol is secure and effective for the cloud computing environment.

5.1. Formal Analysis Based on BAN Logic. In this section, we apply the BAN logic [40] which is a widely accepted way to analyze the design logic and security of the authentication scheme. Its particular notions to depict protocols are shown in Table 2.

In BAN logic, the goals of our authentication scheme are defined as follows:

(i) Goal 1: $U_i \equiv CS_j \equiv (U_i \xrightarrow{SK} CS_j)$.
(ii) Goal 2: $U_i \equiv (U_i \xrightarrow{SK} CS_j)$.
(iii) Goal 3: $CS_j \equiv U_i \equiv (U_i \xrightarrow{SK} CS_j)$.
(iv) Goal 4: $CS_j \equiv (U_i \xrightarrow{SK} CS_j)$.

According to the proof steps in BAN logic, we redepict our scheme into an idealized form:

(i) $Message_1$: $U_i \rightarrow CS_j$: $(X_2, G_i, F_i, Z_i, U_i \xrightarrow{SK} RA)$, $U_i \xrightarrow{SK} RA$.
(ii) $Message_2$: $CS_j \rightarrow RA$: $(Message_1, Y_2, K_j, PSI_D)$, $CS_j \rightarrow RA$.
(iii) $Message_3$: $RA \rightarrow CS_j$: $(X_2, V_{cs}, (Q_{cs})D_j)$, $CS_j \rightarrow RA$.
(iv) $Message_4$: $CS_j \rightarrow U_i$: $(Y_2, (Q_{cs})D_j)$, $CS_j \rightarrow RA$.

Then, some assumptions are defined as follows:

(i) $H_1$: $U_i \equiv \{X_2\}$.
(ii) $H_2$: $CS_j \equiv \{Y_2\}$.
(iii) $H_3$: $RA \equiv \{X_2\}$.
(iv) $H_4$: $RA \equiv \{Y_2\}$.
(v) $H_5$: $RA \equiv CS_j \rightarrow RA$.
(vi) $H_6$: $CS_j \equiv CS_j \rightarrow RA$.
(vii) $H_7$: $RA \equiv U_i \xrightarrow{SK} RA$.
(viii) $H_8$: $U_i \equiv U_i \xrightarrow{SK} RA$.
(ix) $H_9$: $CS_j \equiv U_i \equiv U_i \xrightarrow{SK} CS_j$.
(x) $H_{10}$: $CS_j \equiv U_i \equiv U_i \xrightarrow{SK} CS_j$.

Based on the definition above, we perform the BAN logic proof as follows:

From $Message_2$ ($Message_3$ includes $Message_1$), it is easy to get $S_1$: $RA \equiv \{Message_1, Y_2, K_j, PSI_D\}$.

Then according to $H_{2}, S_1$, RULE(1), we get $S_2$: $RA \equiv U_i \equiv \{X_2, G_i, F_i, Z_i, U_i \xrightarrow{X_i} RA\}$.

According to $H_3$ and RULE(4), we get $S_3$: $RA \equiv$ $\{X_2, G_i, F_i, Z_i, U_i \xrightarrow{X_i} RA\}$.

And according to $S_2, S_3$ and RULE(2), we get $S_4$: $RA \equiv U_i \equiv \{X_2, G_i, F_i, Z_i, U_i \xrightarrow{X_i} RA\}$.

From the Message, we also get $S_5$: $CS_j \equiv \{Y_2, K_j, PSI_D\}$.

Then according to $H_{2}, S_1$, RULE(1), we get $S_6$: $CS_j \equiv RA \equiv \{Y_2, K_j, PSI_D\}$.

According to $H_3$ and RULE(4), we get $S_7$: $CS_j \equiv$ $\{Y_2, K_j, PSI_D\}$.

And according to $S_2, S_3$ and RULE(2), we get $S_8$: $CS_j \equiv RA \equiv \{Y_2, K_j, PSI_D\}$.

From $Message_1$, it is easy to get $S_9$: $RA \equiv \{X_2, V_{cs}, (Q_{cs})D_j\}$.

And according to $S_2, S_3$ and RULE(2), we get $S_{10}$: $RA \equiv \{X_2, V_{cs}, (Q_{cs})D_j\}$.

According to $H_3$ and RULE(4), we get $S_{11}$: $RA \equiv$ $\{X_2, V_{cs}, (Q_{cs})D_j\}$.

And according to $S_2, S_3$ and RULE(2), we get $S_{12}$: $RA \equiv CS_j \equiv \{X_2, V_{cs}, (Q_{cs})D_j\}$.

From $Message_4$, it is easy to get $S_{13}$: $U_i \equiv \{Y_2, (Q_{cs})D_j\}$.

Then according to $H_3$, $S_1$, RULE(1), we get $S_{14}$: $U_i \equiv RA \equiv \{Y_2, (Q_{cs})D_j\}$.

According to $H_3$ and RULE(4), we get $S_{15}$: $U_i \equiv$ $\{Y_2, (Q_{cs})D_j\}$.

And according to $S_2, S_3$ and RULE(2), we get $S_{16}$: $U_i \equiv RA \equiv \{Y_2, (Q_{cs})D_j\}$.

As $SK = h(Y_2 || X_2 || N_j \cdot Y_2)$, and combining $S_{12}, S_{16}$, we get $S_{17}$: $U_i \equiv CS_j \equiv U_i \xrightarrow{SK} CS_j$ (Goal 1).

Similarly, as $SK = h(Y_2 || X_2 || N_j \cdot X_2)$, with $S_{14}, S_8$, we get $S_{18}$: $CS_j \equiv U_i \equiv U_i \xrightarrow{SK} CS_j$ (Goal 3).

Finally, according to $H_2, S_{17}$, and RULE(3), we get $S_{19}$: $U_i \equiv (U_i \xrightarrow{SK} CS_j)$ (Goal 2).

And according to $H_{10}, S_{18}$, and RULE(3), we get $S_{20}$: $CS_j \equiv (U_i \xrightarrow{SK} CS_j)$ (Goal 4).

In conclusion, our scheme achieves Goals 1–4, which promises (1) $U_i$ and $CS_j$ have got authenticated mutually and (2) they negotiate the same session key $SK$.

5.2. Informal Analysis. Looking at the history of protocol designing, due to its simplicity and effectiveness, the heuristic method “still plays an important role” in cryptanalysis of protocols [20], though it does not have a theoretical form and relies on human experience heavily. Therefore, this section gives the security analysis via the heuristic method.

User Anonymity. As we mentioned in Section 3.2, user anonymity contains two aspects, we prove our user anonymity attribute from two points.

(1) $\mathcal{A}$ has no chance to acquire $ID_i$. In our scheme, the identity is transmitted in a form of $F_i$ where $F_i = PID_i \oplus h(X_i)$, $PID_i = h(ID_i || b_i)$, it is obvious that $\mathcal{A}$ has two challenges in computing $ID_i$: firstly, computing $PID_i$ from $F_i$; then guessing $ID_i$ from $PID_i$. The one only with $N_i$ or $x_i$ can compute $h(X_i)$ successfully, while $\mathcal{A}$ has no way to get this two parameters. Furthermore, even with $PID_i$, $\mathcal{A}$ still cannot conduct a guessing attack to compute $ID_i$ for $\mathcal{A}$ does not know $b_i$. 


A consists of a dynamic parameter where intercepted parameters contains their preset shared secret parameter. As a matter of fact, once getting $R_A$, $K_1$ wireless communications and mobile computing, dynamic parameter authenticates in step 2 of Section 4.2. Then $S_K$ secure. In our scheme, the session key $j$ was exposed, the previous session is still the two sessions are sent by a same user.

Therefore our scheme achieves user anonymity.

**Forward Secrecy.** Forward secrecy requires that even the long-term secret key was exposed, the previous session is still secure. In our scheme, the session key $SK = h(Y_2 \parallel X_2 \parallel X_3)$ where $X_3 = N_iY_2 = N_iN_j\parallel N_j\parallel P$. With the help of $x$, and the intercepted parameters $X_2$ and $Y_2$ from the open channel, if $A$ wants to compute $X_3$, then he/she has to solve the ECDDH problem which cannot be finished in the polynomial time. As a matter of fact, once getting $x$, $A$ almost has the same capacity with RA. If RA cannot compute $SK$ in the scheme, $A$ is probably not to do. This again confirms our previous view: for the security consideration, we shall not let RA know SK. All in all, our enhanced scheme provides forward security.

**Mutual Authentication.** Mutual authentication is the most basic requirement of a user authentication scheme. In our scheme, RA firstly authenticates $U_i$ through $G_i$ which contains their preset shared secret parameter $D_i$ and the dynamic parameter $X_1$ generated by the public-key algorithm in step 2 of Section 4.2. Then RA authenticates $CS_j$ through $K_j$ with their shared secret value $S_{key,j}$. In a short, RA authenticates $U_i$ and $CS_j$ after this process.

If both of $U_i$ and $G_i$ are authenticated, RA further computes $Q_{ei}$ and $V_{ei}$. Then in step 3 of Section 4.2, $CS_j$ checks the validity of RA via $V_{ei}$. If RA is authentic, then $CS_j$ believes RA’s judgment on $U_i$, which means that $CS_j$ also trusts the legitimacy of $U_i$. In conclusion, $CS_j$ authenticates RA and $U_i$ via this process.

On receiving $Q_{ei}$, $U_i$ verifies RA with $Q_{ei}$ in step 4 of Section 4.2. If RA is authenticated, then $U_i$ also believes the validity of $CS_j$. Therefore, $U_i$ authenticates RA and $CS_j$.

In conclusion, our scheme achieves mutual authentication.

**Privileged Insider Attack.** To avoid privileged insider attack, when $U_i$ registers to RA, he/she does not submit the identity or password directly, but a transform of them: $\{A_i, PID_i\}$. Thus the identity and password are protected by $b_i$. Even the administrator of RA cannot get $b_i$ to conduct an offline dictionary attack to guess the value of $ID_i$ and $PW_i$. Therefore, our scheme is secure against privileged insider attack.

**Offline Dictionary Attack.** As we mentioned when analyzing Amin et al.‘s scheme [3] in Section 3.2, there are two common offline dictionary attacks. So we consider two kinds of adversary here.

Suppose an adversary $A$ acquires $\{C_i, E_i, DP, n_i\}$ in the smart card, then $A$ may conduct an offline dictionary attack as follows.

**Step 1.** Guess $ID_i$ and $PW_i$ to be $ID_i^*$ and $PW_i^*$, respectively.
Step 2. Compute $b^*_i = DP \oplus h(ID^*_i \parallel PW^*_i)$ Section 4.1.

Step 3. Compute $A^*_i = h(PW^*_i \parallel b^*_i)$.

Step 4. Compute $PID^*_i = h(ID^*_i \parallel b^*_i)$.

Step 5. Compute $D^*_i = E_i \oplus A^*_i$.

Step 6. Compute $C^*_i = h(A^*_i \parallel PID^*_i \parallel D^*_i) \mod n_o$.

Step 7. Verify the correctness of $PW_i$ and $ID_i$ by checking if $(C^*_i = C_i)$. 

Step 8. Repeat steps 1 ~ 7 until the correct valued of $PW_i$ and $ID_i$ are found.

Now we suppose $\mathcal{A}$ has found such a pair of $(ID^*_i, PW^*_i)$ after the above steps. While, due to the properties of equation $C_i$, even the pair satisfies the equations, it is likely that they are not equal to $(ID_i, PW_i)$, since there are $|\mathcal{P}_{pw} \times \mathcal{D}_{id}|/n_o = 2^{32}$ candidates of $(ID_i, PW_i)$ pair when $n_o = 2^{32}$ and $|\mathcal{P}_{pw}| = |\mathcal{D}_{id}| = 2^{32}$ [30]. Consequently, $\mathcal{A}$ needs to verify $(ID^*_i, PW^*_i)$ online, but it will be restrained by Honey List. Once the failure numbers of user login exceeds the preset value, the smart card will be suspended. Accordingly, our scheme is secure against such an attack scenario.

Suppose an adversary $\mathcal{A}$ not only extracts the message in smart card, but also eavesdrops $(G_i, F_i, Z_i, X_i)$ from the open channel, then $\mathcal{A}$ attempts to guess $PW_i$ and $ID_i$. In this occasion, $\mathcal{A}$ wants to use $G_i$ as the verification parameter to check the correctness of the guessed value of $(PW_i , ID_i)$. As $G_i$ consists of $PID_a, SID_j, X_j$ and $D_j$, and according to our attack steps 1~5 above, $\mathcal{A}$ can compute the value of $PID^*_j$ and $D^*_j$. Then $\mathcal{A}$ computes $SID^*_j \oplus h(D^*_j \parallel X_j)$. Now $\mathcal{A}$ only needs to compute $X_j$. While without $x$ or $N_j$, computing $X_j$ is equivalent to solving the EC DL problem which cannot be finished in the polynomial time. As a result, our scheme can prevent such an adversary.

In conclusion, the proposed scheme is secure against dictionary attack.

**Verifier-Stolen Attack.** In our scheme, RA only needs to maintain the User-list whose elements ([$PID_i, r_i, Hoeny\_List$]) are not security-related. Furthermore, even $\mathcal{A}$ steals User-list, he/she will learn nothing useful information to conduct an attack. Thus our scheme is resistant to verifier-stolen attack.

**Replay Attack.** We prevent the replay attack via the random numbers to prevent replay attack. We take one of the message flow $(G_i, F_i, Z_i, X_i)$ as an example to explain: suppose $\mathcal{A}$ eavesdrops $(G_i, F_i, Z_i, X_i)$, then replays it to $CS_i$. While $\mathcal{A}$ does not know $N_i$, he/she cannot compute the correct session key though the replayed message can pass the verification of $RA$. Consequently, $\mathcal{A}$ gain no benefits from such an attack. Equally, it makes no sense for $\mathcal{A}$ to replay other message flows. Accordingly, our scheme is secure against replay attack.

**User Impersonation Attack.** According to the definition user impersonation attack in [24], $\mathcal{A}$ does not acquire the smart card (this condition is included in “offline dictionary attack”) here. As we analyzed above, $\mathcal{A}$ with smart card can neither guess $ID_i$ and $PW_i$ nor replay $(G_i, F_i, Z_i, X_i)$ to impersonate $U_i$, let alone the adversary without smart card. So there is only one possible method left: constructing $(G_{\alpha}, F_{\alpha}, Z_{\alpha}, X_{\alpha})$. To construct this message, $\mathcal{A}$ chooses $N_{\alpha}$, computes $(X_{\alpha}, ID_{\alpha}, PW_{\alpha})$, and finally sends it to RA. However, after RA gets $c_{\alpha}$ and $PID_{\alpha}$, $RA$ may fail to find such a $PID_{\alpha}$ in $User\_list$ or computes a $D_{\alpha}$ unequal to $D_{\alpha}$, both the two conditions lead to the failure in the authentication of $U_{\alpha}$. As a result, $RA$ finds that $\mathcal{A}$ is not a legitimate user, the attack fails.

**Server Impersonation Attack.** According to the above analysis, $\mathcal{A}$ can neither conducts the replay attack to impersonate $CS_i/RA$ nor finds ways to compute $Skey_{i/x}$, so $\mathcal{A}$ cannot impersonate $CS_i/RA$.

6. **Performance Analysis**

Some schemes like [41] which involve only two participants is essentially indistinguishable from the traditional client-server architecture and does not apply to cloud computing environments with multiple servers. Some schemes like [42] which are more concerned with authentication issues between wearable and smart phone belong to entity authentication rather than the user authentication discussed in this article. These schemes are not comparable to ours. Therefore, we only compare those having similar system architectures and application scenarios including [3, 25, 26].

As shown in Table 3, our security performance is obviously superior to other protocols: the proposed scheme

<table>
<thead>
<tr>
<th>Login(ms)</th>
<th>Auth.(ms)</th>
<th>Communication cost</th>
<th>Login</th>
<th>Auth.</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
<th>$S_8$</th>
<th>$S_9$</th>
<th>$S_{10}$</th>
<th>$S_{11}$</th>
<th>$S_{12}$</th>
<th>$S_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maitra et al. (2016) [25]</td>
<td>$6T_{\mathcal{H}} = 0.04$</td>
<td>$5T_{\mathcal{H}} + 7T_{\mathcal{H}} = 0.12$</td>
<td>640 bits</td>
<td>1280 bits</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>Kumari et al. (2017) [26]</td>
<td>$2T_{\mathcal{H}} + 7T_{\mathcal{H}} \approx 2.3$</td>
<td>$6T_{\mathcal{H}} + 15T_{\mathcal{H}} \approx 7.0$</td>
<td>2176 bits</td>
<td>9088 bits</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>Amin et al. (2018) [3]</td>
<td>$6T_{\mathcal{H}} = 0.04$</td>
<td>$17T_{\mathcal{H}} = 0.12$</td>
<td>640 bits</td>
<td>1920 bits</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>Our scheme</td>
<td>$2T_{\mathcal{H}} + 7T_{\mathcal{H}} \approx 1.0$</td>
<td>$4T_{\mathcal{H}} + 12T_{\mathcal{H}} = 2.0$</td>
<td>1408 bits</td>
<td>4096 bits</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

$T_{\mathcal{H}}$ denotes the time of modular exponentiation operation, $T_{\mathcal{H}}$ denotes hash computation, $T_{e}$ denotes symmetric encryption/decryption, $T_{M} > T_{E} > T_{H} > T_{S}$ ($T_{M} = 1.169\mu s, T_{E} = 0.508\mu s, T_{H} = 0.693\mu s, T_{S} = 0.543\mu s$ [20]). Let $n_{o}$ be 32-bit long. Let $ID_{\alpha}, PW_{\alpha}, h(\cdot)$, output of symmetric encryption, timestamp, random numbers be 128-bit long; let $P, x, y$ be 1024-bit long. $\checkmark$ means the property is satisfied; $\times$ means the property is not satisfied. Note that the evaluation criteria in [24] are also applied to cloud computing environment when regarding $CS_i$ as the sensor node, while the considerations on designing authentication scheme of this two environment are quite different due to their different network attributes.
achieves all security requirements, while others have more or less security flaws. More specifically, both the schemes of Maitra et al. [25] and Amin et al. [3] fail to achieve user anonymity and forward secrecy and cannot resist against offline dictionary attack, etc. The best one is Kumari et al.’s scheme [26] which can only provide 11 items of security requirements. In terms of computation or communication performance, as we mentioned in Section 4, the schemes only involving one-way hash operation are inevitably cost less in communication and computation than those deploying the public-key algorithm, but they certainly cannot guarantee the security of authentication. Therefore, among the compared schemes, the schemes of Amin et al. [3] and Maitra et al. [25] certainly cost less communication time and load for they only involve some one-way hash operations. However, sacrificing security to achieve high performance is inadvisable in authentication protocols. As a matter of fact, certain cost is unavoidable for security. Then compared with Kumari et al.’s scheme [26] which is equipped with public-key algorithm, our scheme costs 1ms in login phase and 2ms in authentication phase, while theirs is 2.3ms and 7ms, respectively, our computation overhead is better. Furthermore, our communication cost (1408 bits in login phase and 4096 bits in authentication phase) is also lower than theirs (2176 bits and 9088 bits). In conclusion, our scheme with all security attributes is more suitable for cloud computing environment.

7. Conclusion

The rapid development of cloud computing makes people’s lives more convenient, but also brings huge security concerns. In order to ensure user’s privacy and account security in the cloud environment, a large number of authentication schemes were proposed, but they were subsequently pointed out having one or more flaws. In order to explain the subtleties of designing an authentication protocol in the cloud environment, this paper took Amin et al.’s protocol as a study case to provide ideas for designing secure protocol for cloud environment through elaborating the security weaknesses existing in the protocol and its corresponding solutions. In addition, based on the analysis, we designed a secure authentication protocol, used the BAN logic and heuristic analysis method to prove the security of the protocol. When comparing it with related protocols, we found our scheme has obvious advantages.

Data Availability

There are not any data used in my paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the National Key Research and Development Program of China (No. 2017YFB0809090); the BUPT Excellent Ph.D. Students Foundation under Grant No. CX2017206; and the National Natural Science Foundation of China under Grant No. 61472016.

References


