Received: 12 February 2017 | Revised: 26 March 2017 | Accepted: 17 April 2017
DOI: 10.1002/dac.3336

RESEARCH ARTICLE

WILEY

A lightweight password-based authentication protocol using smart card

Chenyu Wang[1] | Ding Wang[2] | Guoai Xu[1] | Yanhui Guo[1]

[1]Beijing University of Posts and Telecommunications, Beijing, China

[2]School of Electronics Engineering and Computer Science, Peking University, Beijing, China

Correspondence
Ding Wang, School of Electronics Engineering and Computer Science, Peking University, Beijing, China.
Email: wangdingg@mail.nankai.edu.cn

Funding information
National Natural Science Foundation of China (NSFC), Grant/Award Number: 61401038

Summary

With its simplicity and feasibility, password-based remote user authentication becomes a popular way to control remote access to network. These years, numerous password-based authentication schemes have been proposed. Recently, Maitra et al proposed a smart card–based scheme which claims to be resistant to various attacks. Unfortunately, we found some important flaws in this scheme. Therefore, in this paper, we will demonstrate that the scheme of Maitra et al is not secure enough as claimed: neither resisting against off-line password guessing attack and insider attack nor preserve forward secrecy. To overcome those flaws, we put forward an improved new scheme which not only is resistant to all known attacks but also provides many attractive attributes, such as user revocation and re-register. Also, we compared the scheme with other related schemes, the result proved the superiority of our scheme. Particularly, we show a new way (beyond the conventional Deffie-Hellman approach) to achieve forward secrecy. Furthermore, we put some efforts into exploring the design principle of authentication schemes.

KEYWORDS

2-factor remote user authentication, discrete logarithm problem, offline-password guessing attack, RAS cryptosystem, smart card

1 | INTRODUCTION

With the rapid proliferation of Internet, online services become an indispensable part of people's life. However, due to the openness of the Internet, people are bound to be faced with the attendant security threats when they enjoy the convenience of the Internet. In fact, it is not surprising to a variety of reports of user information disclosure any more. Most recently, Ashley Madison, which is a web about extramarital love affair, suffered from an attack where its 32 million Ashley Madison users' information got exposed; Jingdong, which is an e-commerce platform, divulged nearly 12G users' privacy data; Gmail, which is an email product provided by Google, had its 5 million passwords accounts breached. Thus, it is vital to authenticate a remote user with password when he/she intends to access the server via an insecure communication channel, and this security mechanism is called password-based remote user authentication.

The remote authentication is a significant security mechanism to guarantee the legitimacy of the user to enjoy resources in various open environments, such as cloud computing, E-health, and wireless sensor.[1-4] The history of password-based user authentication scheme can be traced back to 1981, when Lamport[5] firstly advised a password-based remote authentication scheme. Since then, authentication schemes armed with passwords were widely accepted. And many password-based authentication schemes[6-9] sprang up. However, these schemes have an inherent weakness: A password table must be maintained by a server, which therefore results in 2 serious problems: (1) Most schemes are vulnerable to stolen-verifier attack or off-line dictionary attack, and (2) it costs a lot to protect and maintain a password table. To overcome these pitfalls, Chang and Wu[10] firstly introduced the smart card as another factor besides password into authentication schemes, which contributes to the 2-factor authentication scheme. In such scheme, the users are not only required to know the correct password but also the corresponding smart card, then he/she can access to remote resource by interacting with the server. The smart card–based authentication scheme usually involves 2 participants: the users (the number may be a lot) and the server (only a single). Furthermore, it includes 4 basic phases: registration phase, login phase, verification phase, and password-change phase. In registration phase, the user firstly submits the server his/her personal messages, then he/she will receive a smart card with important parameters from the server. If a registered user wants to access to the server, he/she will send a access request in login phase. Then in verification phase, the 2 participants are required to authenticate each other. If they both get authenticated, the authentication process finished successfully, then, the user can enjoy the services from the server. Furthermore, once the user wants to change his/she password, he/she can initiate the password change phase to change password locally or remotely. In short, a well-defined smart card–based password authentication scheme should ensure that only when a user who does not only submit the correct password but also owns corresponding smart card can the access request be allowed. Thanks to Chang and Wu, smart card got popular in most 2-factor authentication schemes and some notable ones including.[11-14] Compared with the previous single-factor password-based schemes, these schemes are low cost and have better cryptographic capacity. Till now, smart card–based password authentication scheme becomes one of the most common or convenient mechanisms to ensure the security of network in 2-factor authentication protocols. Furthermore, these years, many schemes use biometric characteristic as an additional factor to provide the authentication,[15-18] while the biometric-based authentication scheme has its inherent drawback: high deployment costs and unrecoverable. Anyway, biometric characteristic is beyond our work in this paper.

With in-depth study, scholars have reached a consensus on several key points about designing smart card–based password authentication scheme:

1. The information in the smart card can be extracted. With smart card, both the adversary and the legitimate user can get the parameters stored in smart card.[19-21]
2. To achieve user anonymity and resistance of off-line password guessing attack, a lightweight public key algorithm is necessary.[22]
3. In 2015, Wang et al[23] indicated that there is an unavoidable trade off between changing password locally and resisting to smart card loss attack without new technique. Surprisingly, Wang and Wang[20] in 2016 for the first time integrated "honeywords" and "fuzzy-verifiers" to settle this issue successfully.
4. Forward secrecy requires server side to conduct more than 2 exponentiation operations.[22]

1.1 | Motivations and contributions

Although have been developed for 20 years, password-based authentication schemes are still far from satisfactory. What is more, many schemes even violate some basic design principles thus making elementary mistakes. This situation forces us to try to devise a secure but lightweight scheme which can be resistant to known attacks and provide ideal attributes. Furthermore, to avoid elementary mistakes in the area of 2-factor authentication scheme, a detailed explanation of the design thinking and points is necessary. So, in this paper, we first choose a recent scheme as a study case and analyze its weaknesses and design thinking. On the basis of the analysis, we proposed an improved scheme and proved that our scheme can be resistant to various attacks with lower computing and storage overhead. In a word, our contribution can be summarized as below:

1. We analyze a recent proposed scheme: the scheme of Maitra et al,[24] which claimed to be resistant to various attacks. Unfortunately, we demonstrate that, under their assumptions of the capabilities of the adversary, this scheme is not secure to resist offline-password attack and insider attack, and fails to achieve forward secrecy, etc.
2. On the basis of the scheme of Maitra et al, we explain the design thinking and points of 2-factor authentication scheme then propose an enhanced scheme.
3. We prove that our scheme can be resistant to the known attacks with lower computing and storage than other related schemes. Particularly, we show a new way (beyond the conventional Deffie-Hellman approach) to achieve forward secrecy.

1.2 | Construction of the paper

The remainder of this paper is organized as follows: Section 2 introduces the preliminaries. The cryptanalysis of the scheme of Maitra et al is given in Section 3. Then, we propose our scheme in Section 4. Section 5 gives a security and performance analysis to our scheme. Section 6 gives a conclusion.

2 | PRELIMINARIES

This section firstly describes the computational problems, then lists the notations, and finally introduces the capacities of adversary and the evaluation criteria of the authentication scheme.

2.1 | Computational problems

This section introduces the discrete logarithm problem, hash function, and Rivest-Shamir-Adleman (RSA) cryptosystem; they are the key to design the scheme of Maitra et al and our scheme.

2.1.1 | The discrete logarithm problem

- Discrete logarithm problem: g is the generator of cyclic group Z_p^*, for given $(g, g^\alpha \bmod p)$; it is hard to compute $\alpha(\alpha \in Z_p^*)$ within a polynomial time.
- Computational Diffie-Hellman problem: g is the generator of the cyclic group Z_p^*, and $\alpha, \beta \in Z_p^*$, for given $(g^\beta, g^\alpha \bmod p)$; it is hard to compute $g^{\alpha\beta}$ within a polynomial time.

2.1.2 | RSA cryptosystem

1. Key generation.
   - Choose 2 distinct prime numbers p and q; compute $n = p \cdot q$.
   - Compute $r = (p-1)(q-1)$.
   - Choose an integer e such that $1 < e < r$, e and r are coprime.
   - Compute d as $d \equiv e^{-1} \bmod r$.
   - Then the public key is (n, e), and the private key is (n, d).
2. Encryption. $c \equiv m^e \bmod n$, where m is an integer and c is the ciphertext.
3. Decryption. $c^d \equiv (m^e)^d \equiv m \bmod n$.

2.1.3 | One-way hash function

- The length of x is variable; y has a fixed-length.
- Given x, it is easy to get y; while given y, it is difficult to compute x.

2.2 | Notations and abbreviations

The notations used in the schemes are shown in Table 1.

2.3 | The capacities of adversary

In cryptanalysis of 2-factor authentication scheme, the adversary $\mathcal{A}$ is also supposed to be armed with some capacities[20,23,25,26] as shown in Table 2:

TABLE 1 Notations and abbreviations

Symbol	Description
U_i	*ith* user
S	remote server
$\mathcal{A}$	the adversary
x	the secret key of S
ID_i	identity of U_i
PW_i	password of U_i
$\oplus$	XOR operation
$\|\|$	concatenation operation
$h(\cdot)$	1-way hash function
$\rightarrow$	a common channel
$\Rightarrow$	a secure channel

2.4 | Evaluation criteria

In the history of the 2-factor authentication protocol, a mass of new protocols were proposed, while shortly were pointed insecure to known attacks or lacking some crucial attribute. One of the important reasons is that the evaluation criteria is not uniform. The authors tend to design their own criteria set which their new protocol can satisfy, while ignoring those criteria which their protocol fails to meet. To settle this issue, Madhusudhan and Mittal[27] in 2012 put forward a series of evaluation criteria, including several security requirements and desire attributes. In fact, after that, most authors design their evaluation criteria according to them: Select some of the indicators in Mad-Mit criteria set to form their own evaluation criteria. However, there is no protocol that meets all the indicators in the Mad-Mit criteria set simultaneously so far. In 2015, Wang et al[23] demonstrated that those evaluation indicators are not independent of each other. For example, "Inside attack" is actually equivalent to "No password reveal"; "Password dependent" is included in "Smart card loss attack." Thus, in 2016, Wang and Wang[20] designed a new evaluation criteria. These evaluation indicators are independent of each other and cover all the security requirements and ideal attributes involved in the current 2-factor protocols. So in this paper, we will employ this set of evaluation criteria (shown in Table 3). It should be noted that, here, the attribute "Password friendly" not only requires to allow the user choose their password freely but also change their password locally; the attribute "No smart card loss attack" refers to those attacks that the adversary owns the smart card, while in "Resistance to know attacks," the adversary does not obtain the card; the attribute "Sound repairability" means that a user revokes the account and re-register without changing his/her identity; the attribute "No clock synchronization" requires no-time-clock–based operations.

TABLE 2 The capacities of the adversary[20]

1	The adversary $\mathcal{A}$ can control the open channel fully, ie, $\mathcal{A}$ can intercept, modify, delete, and resend the messages over the open channel.
2	The adversary $\mathcal{A}$ can list all pairs of (ID_i, PW_i) from $\mathcal{D}_{pw} * \mathcal{D}_{id}$ in a polynomial time, where $\mathcal{D}_{pw}$ refers the password space and $\mathcal{D}_{id}$ denotes the identity space.
3	The adversary $\mathcal{A}$ can either acquire the password of the user via malicious terminal or extract the parameters from smart card.
4	When evaluating forward secrecy, the adversary can get server's private key.
5	The adversary $\mathcal{A}$ can extract the information stored in smart card.

TABLE 3 Evaluation criteria

1	No password verifier–table	2	Password friendly
3	No password exposure	4	No smart card loss attack
5	Resistance to know attacks	6	Sound repairability
7	Provision of key agreement	8	No clock synchronization
9	Timely typo detection	10	Mutual authentication
11	User anonymity	12	Forward secrecy

3 | REVIEW OF THE SCHEME OF MAITRA ET AL

In 2016, Maitra et al revealed that the scheme of Lee et al cannot resist against forgery attack, password guessing attack, etc, thus proposed an improved scheme claiming to be resistant to those attacks. While in this section, we will demonstrate that the scheme of Maitra et al is still vulnerable to various attacks. Furthermore, we analyze the inherent reason for these attacks.

3.1 | The scheme of Maitra et al

We first briefly introduce the scheme of Maitra et al.[24] As the password change phase has little relation to our work, we omit it.

1. *Initialization phase*. S selects a generator g of a multiplicative group G of prime order p and a secret long key x where $x \in Z_q^*$, then, the corresponding public key $y = g^x \bmod p$. There is also a hash function $h(.)$: $0, 1^* \rightarrow 0, 1^l$, where l is the bit length of function output.
2. *Registration phase*.

   Step 1. $A \Rightarrow S$: $\{PW_i, ID_i\}$.
   Step 2. S chooses a random number n_i and calculates $A_i = h(ID_i \oplus h(x))$, $PWR_i = h(PW_i \oplus n_i)$, $L_i = A_i \oplus PWR_i$, $B_i = h(A_i || PWR_i)$, and $n = n_i \oplus h(ID_i \oplus PW_i)$, and adds $\{h(h(A_i))\}$ into *user_list* to store.
   Step 3. $S \Rightarrow A$: a smart card with $\{L_i, B_i, n, h(.)\}$

3. *Login and authentication phase*.

   Step 1. U_i puts the smart card into a card reader and enters ID_i and PW_i.
   Step 2. The smart card computes $n_i' = n \oplus h(ID_i \oplus PW_i)$, $PWR_i' = h(PW_i \oplus n_i')$, $A_i' = L_i \oplus PWR_i'$, and $B_i' = h(A_i' || PWR_i')$, if $B_i \neq B_i'$, ends the session, chooses a random number r_1 and timestamp T, and computes $C_1 = A_i'^{r_1} \bmod p$, $C_2 = g^{r_1} \bmod p$, $D = h(T \oplus A_i') \bmod (p-1)$, $DID = ID_i \oplus h(y^{r_1} \bmod p)$, $E_i = A_i'^{D} \bmod p$, and $F_i = E_i(C_1)^D \bmod p$.
   Step 3. $A \rightarrow S$: $\{DID, C_1, C_2, F_i, T\}$.
   Step 4. S first tests the freshness of T, then computes $ID_i^* = DID \oplus h(C_2^x \bmod p)$, $A_i^* = h(ID_i^* \oplus h(x))$, checks whether $h(h(A_i^*))$ is in the *user_list*, if not, ends the session, otherwise, computes $D_i^* = h(T_i \oplus A_i^*) \bmod (p-1)$; if $F_i(C_1^{D^*})^{-1} \neq A_i^{*D^*} \bmod p$, ends the session, otherwise, computes $G_i = h(T_s \oplus A_i^*) \bmod p$ and $K_i = C_1^{G_i} \bmod p$, where T_s is the timestamp in server side.
   Step 5. $S \rightarrow A$: $\{K_i, T_s\}$.
   Step 6. The smart card checks the validity of T_s, then computes $G_i' = h(T_s \oplus A_i') \bmod (p-1)$, $K_i' = C_1^{G_i'} \bmod p$, and tests whether $K_i = ?K_i'$, and if it fails, ends the session and the authentication fails. If it is true, both S and U_i accept the session key $SK = h(h(A_i') \oplus T_s) = h(h(A_i^*) \oplus T_s)$.

3.2 | Cryptanalysis of the scheme of Maitra et al

This section demonstrates that the scheme of Maitra et al not only is not secure from off-line password guessing attack and insider attack but also cannot preserve forward secrecy, etc.

3.2.1 | Off-line password guessing attack I

Supposing an adversary $\mathcal{A}$ stole U_i's smart card and extracted $\{L_i, B_i, n\}$ from the smart card, then, he can perform off-line password guessing attack I as below:

Step 1. Guess the value of PW_i to be PW_i^* from the password dictionary space $\mathcal{D}_{pw}$ and the value of ID_i to be ID_i^* from the identity dictionary space $\mathcal{D}_{id}$.
Step 2. Compute $n_i' = n \oplus h(ID_i^* \oplus PW_i^*)$, $PWR_i' = h(PW_i^* \oplus n_i')$, $A_i' = L_i \oplus PWR_i'$, $B_i' = h(A_i' || PWR_i')$, where L_i and n were extracted from the smart card.
Step 3. Verify the correctness of PW_i and ID_i by checking if $B_i' = ?B_i$, where B_1 was extracted from the smart card.

Step 4. Repeat Steps 1 to 3 until the correct values of PW_i and ID_i are found.

After getting PW_i and ID_i, the adversary $\mathcal{A}$ can impersonate of U_i through multiple ways, such as getting a smart card and inputting PW_i and ID_i or computing $\{DID, C_1, C_2, F_i, T\}$ directly and sending them to the server. $\mathcal{A}$ also can compute $A_i' = L_i \oplus PWR_i'$, $G_i' = h(T_s \oplus A_i') \bmod (p-1)$, $K_i' = C_1^{G_i'}$. So $\mathcal{A}$ can impersonate of S to communicate with U_i. Furthermore, a man-in-the-middle attack is feasible too. So once this attack is performed successfully, the whole security of the system would be compromised.

The time complexity: $\mathcal{O}(|\mathcal{D}_{pw}| * |\mathcal{D}_{id}| * (2T_H + 3T_I)$. T_H is the running time for hash computation, and T_I is the running time of exclusive-or operation. $|\mathcal{D}_{pw}|$ denotes the number of passwords in $\mathcal{D}_{pw}$, and $|\mathcal{D}_{id}|$ denotes the number of identities in $\mathcal{D}_{id}$. $|\mathcal{D}_{pw}|$ or $|\mathcal{D}_{id}|$ is very limited in practice,[26,28] usually $|\mathcal{D}_{id}| \leqslant |\mathcal{D}_{pw}| \leqslant 10^6$, so the attack is efficient.

In the above attack, $\mathcal{A}$ just extracts the message in the smart card then can perform the attack. According to our analysis, it is obvious that the attack is quite practical and efficient, and the attack occasion can be found in many papers.[15,21,23,29]

The inherent reason for this attack is that, on the one hand, the adversary can get a verification parameter to verify the correctness of his guessing value. In this scheme, the verification parameter is B_i; on the other hand, the parameters that consist B_i can be denoted with the guessed PW_i^*, ID_i^* (other parameters such as L_i and n are accurately known by $\mathcal{A}$). Despite B_i results in the attack, B_i is also the key parameter to help the user change password locally. This is exactly the trade off between the security and usability demonstrated by Wang et al.[23] Luckily, Wang and Wang,[20] for the first time, integrated "honeywords" and "fuzzy-verifiers" to settle this conflict. According to Wang and Wang,[20] the issue can be tackled as follows: Let the verification $B_i = h(A_i' \| PWR_i')$ mod n_0), where $2^4 \leqslant n_0 \leqslant 2^8$ and n_0 determines the capacity of the pool of the (ID, PW). So now, there are $|\mathcal{D}_{pw}| * |\mathcal{D}_{id}| \setminus n_0 \approx 2^{32}$ candidates of (ID_i, PW_i) pair for adversary to guess when $n_0 = 2^8$ and $|\mathcal{D}_{pw}| = |\mathcal{D}_{id}| = 2^6$. For these candidates, the adversary can only guess the correct one from online guessing. While there is also a way called "honeywords" to timely detect whether the smart card is breached to avoid such online dictionary guessing.

3.2.2 | Off-line password guessing attack II

Supposing an adversary $\mathcal{A}$ not only stole U_i's smart card and extracted $\{L_i, B_i, n\}$ from the smart card but also eavesdropped $\{DID, C_1, C_2, F_i, T\}$ from the open channel, then, he could perform off-line password guessing attack II as below:

Step 1. Guess the value of PW_i to be PW_i^* and ID_i to be ID_i^*.

Step 2. Compute $n_i' = n \oplus h(ID_i^* \oplus PW_i^*)$, $PWR_i' = h(PW_i^* \oplus n_i')$, $A_i' = L_i \oplus PWR_i'$, $D' = h(T \oplus A_i') \bmod (p-1)$, $E_i' = A_i'^{D} \bmod p$, and $F_i' = E_i(C_1)^D \bmod p$

Step 3. Verify the correctness of PW_i and ID_i by checking if F_i'=?F_i, where F_i was extracted from the open channel.

Step 4. Repeat Steps 1 to 3 until the correct value of PW_i and ID_i are found.

Once the adversary $\mathcal{A}$ gets PW_i and ID_i, the whole security of the system is compromised. As the way to other attacks such as user impersonation attack are similar to what we described in "off-line password guessing attack I," so we omit the detail attacking steps.

The time complexity: $\mathcal{O}(|\mathcal{D}_{pw}| * |\mathcal{D}_{id}| * (3T_H + 4T_I + 2T_E + T_M)$. T_E is the time of modular exponentiation operation and T_M is the time of modular multiplication operation. Furthermore, the attack occasion can be found in many papers.[21,23,30,31] Thus, the attack is efficient but practical.

The inherent reason for this attack is that, on the one hand, the adversary can find a verification to check the correction of the guessing value, which is inevitable to the demand of authenticating the user for the server in any authentication scheme; on the other hand, the password is the only unknown value to the adversary, which means the adversary can get other parameters consisting of the verification except for the password or the identity. Avoiding the later issue is where the researchers should strive to. Generally, in a sound 2-factor authentication scheme, the verification parameters should include a random number (or its transformation) at least. Furthermore, the random number (or its transformation) cannot be exposed to the open channel. To avoid such attack, the lightweight public-key algorithm is a necessary but not sufficient condition to settle the problem, as explained in Ma et al.[22]

3.2.3 | Forward secrecy

Supposing $\mathcal{A}$ knew S's secret key x, as well as got $\{DID, C_1, C_2, F_i, T\}$, then, he can calculate the session key SK as follows:

Step 1. Compute $ID_i' = DID \oplus h(C_2 \bmod p)$ and $A_i' = h(ID_i' \oplus h(x))$.

Step 2. Interrupt $\{K_i, T_s\}$.

Step 3. Compute $SK = h(h(A_i') \oplus T_s)$, at this point $\mathcal{A}$ gets SK successfully.

The time complexity of the attack above is $\mathcal{O}(|\mathcal{D}_{pw}| * |\mathcal{D}_{id}| * (5T_H + 3T_I)$. So the attack above is quite efficient.

In this scheme, the session key consists of A_i (where $A_i = L_i \oplus PWR_i' = h(ID_i \oplus h(x)))$ and an open timestamp T_s. As T_s is an open parameter and can be easily obtained by $\mathcal{A}$, the only key parameter is A_i, while A_i is an unchanged value and also can be calculated by $\mathcal{A}$. So this scheme certainly cannot preserve forward secrecy. Usually, to achieve forward secrecy, it is better that if the SK consists of 2 fresh random number. Furthermore, the 2 fresh parameters cannot be exposed to the open channel or be stored in the smart card. This requirement also proves that "more than two exponentiation operations conducted on the server side is necessary to achieve forward secrecy."[22]

3.2.4 | Insider attack

In this scheme, the user U_i submits a pair of PW_i and ID_i to the server S without any transformation or protection, thus, the server S can get the PW_i and ID_i, performing an insider attack to impersonate the user U_i.

Insider attack is a basic issue, and it can be settled easily: Do some transformations to camouflage the PW_i, such as $h(PW_i \| a)$ (a is a random number).

4 | PROPOSED SCHEME

Inspired by the way to deal with the conflict between the security and usability in Wang and Wang,[20] we make some amendments to the scheme of Maitra et al and reduce unnecessary computation for the sake of performance, then put forward an improved new remote authentication scheme which overcomes these weaknesses in the scheme of Maitra et al. Compared with the scheme of Wang and Wang,[20] our scheme exploits a way of integrating the discrete logarithm problem with RSA cryptosystem to achieve forward security. Furthermore, the scheme of Wang and Wang[20] focus on providing a model or a template for 2-factor authentication protocols, particularly in the way to settle the conflict between security and usability. While our scheme is built on the scheme of Maitra et al, it aims to improve this scheme to meet the evaluation indicators listed in Section 2.4. In short, our scheme includes 5 phases: initialization phase, registration phase, login phase, authentication phase, and password change phase. As the initialization phase is similar to the phase in the scheme of Maitra et al as described in Section 3.1, we omit it here. The process of our scheme is shown in Figure 1.

4.1 | Registration phase

If a new user intends to access the resource on the server, he has to register to the server firstly:

1. The user U_i chooses a password PW_i and an identity ID_i. Then, the system generates a random number b and computes $PWR_i = h(PW_i \| b)$.
2. $U_i \Rightarrow S$: $\{PWR_i, ID_i\}$.
3. After receiving $\{PWR_i, ID_i\}$ at T_{rg} (the time, and it is secure to brute-force guessing), the server S calculates $A_i = h(h(ID_i) \oplus h(PWR_i) \bmod n_0)$, where n_0 is an integer such that $2^4 \leqslant n_0 \leqslant 2^8$, and checks whether ID_i has been in the *User_List*. If not, S creates a new entry for U_i as $\{ID_i, y_i, T_{rg}, Hoeny\_List\}$, where y_i is a unique random number to U_i chosen by S, *Hoeny_List* is a list to record the number of login failures and is initialized to 0; otherwise, S updates T_{rg} and y_i in the *User_List*. Then S calculates $K_i = h(ID_i \| x \| y_i \| T_{rg})$, $L_i = K_i \oplus PWR_i$.
4. $S \Rightarrow U_i$: a smart card with $\{A_i, L_i, n_0, p, g, y, h(.)\}$.
5. U_i enters b into the smart card.

4.2 | Login phase and authentication phase

Once the user registers to the server successfully, he can send the access request to the server when he wants to enjoy the service as follows:

1. U_i inserts the card into a card reader, inputs ID_i' and PW_i'.
2. The smart card computes $PWR_i' = h(PW_i' \| b)$, $A_i' = h(h(ID_i') \oplus h(PWR_i') \bmod n_0)$, and if $A_i' \neq A_i$, exits the session. Otherwise, the card generates a random number r_1 and computes $C_1 = g^{r_1} \bmod p$, $C_2 = y^{r_1} \bmod p$, $D_i = ID_i \oplus h(C_1 \| C_2)$, and $K_i' = L_i \oplus PWR_i'$, $M_1 = h(D_i \| C_1 \| C_2 \| K_i')$.
3. $U_i \rightarrow S$: $\{C_1, D_i, M_1\}$.
4. Upon getting $\{C_1, D_i, M_1\}$, S computes $C_2' = (C_1)^x \bmod p$ and $ID_i' = D_i \oplus h(C_1 \| C_2')$. Then S searches the *User_List* to find the ID_i such that $ID_i = ID_i'$. If there is not an ID_i that satisfies the condition, S rejects the access request and sets *Hoeny_List* to be *Hoeny_List* + 1. Once the value of *Honey_List* exceeds the predetermined threshold (such as 10), S thinks that the smart card has been breached, thus suspends the card till U_i re-registers. Otherwise, S derives y_i and T_{rg} from the *User_List* and computes $K_i = h(ID_i \| x \| y_i \| T_{rg})$ and $M_1' = h(D_i \| C_1 \| C_2' \| K_i)$. Then S verifies U_i by testing whether $M_1' =?M_1$ and, if the equation is false, ends the session, sets *Hoeny_List* to be *Hoeny_List* + 1, and once the value of *Hoeny_List* exceeds the predetermined threshold (such as 10), suspends the card till U_i re-registers; otherwise, S initializes the RSA algorithm as described in Section 2.1.2, generates the public key (n, e) and the private key (n, d), and then computes $M_2 = h(ID_i \| K_i \| C_1 \| C_2' \| n \| e)$.
5. $S \rightarrow U_i$: $\{M_2, n, e\}$.
6. On obtaining $\{C_3, M_2, T_s\}$, the smart card computes $M_2' = h(ID_i \| K_i' \| C_1 \| C_2 \| n \| e)$ then tests whether $M_2' =?M_2$. If $M_2' \neq M_2$, exit the session. Otherwise the smart card generates a random number r_2 ($0 \leqslant r_2 \leqslant n$) and computes $C_3 = r_2^e \bmod n$ and $M_3 = h(ID_i \| C_3 \| K_i' \| r_2 \| C_2)$.
7. $U_i \rightarrow S$: $\{M_3, C_3\}$.
8. When getting $\{M_3, C_3\}$, S computes $r_2' = C_3^d \bmod n$ and $M_3' = h(ID_i \| C_3 \| K_i \| r_2' \| C_2')$. If $M_2' \neq M_2$, exit the session. Otherwise, both the user U_i and the server S computes their session key as $SK = h(ID_i \| C_1 \| C_2 \| C_3 \| K_i' \| r_2)$ and $SK = h(ID_i \| C_1 \| C_2' \| C_3 \| K_i \| r_2')$, respectively.

4.3 | Password change phase

When the user U_i wants to change his password, he needs perform the following steps:

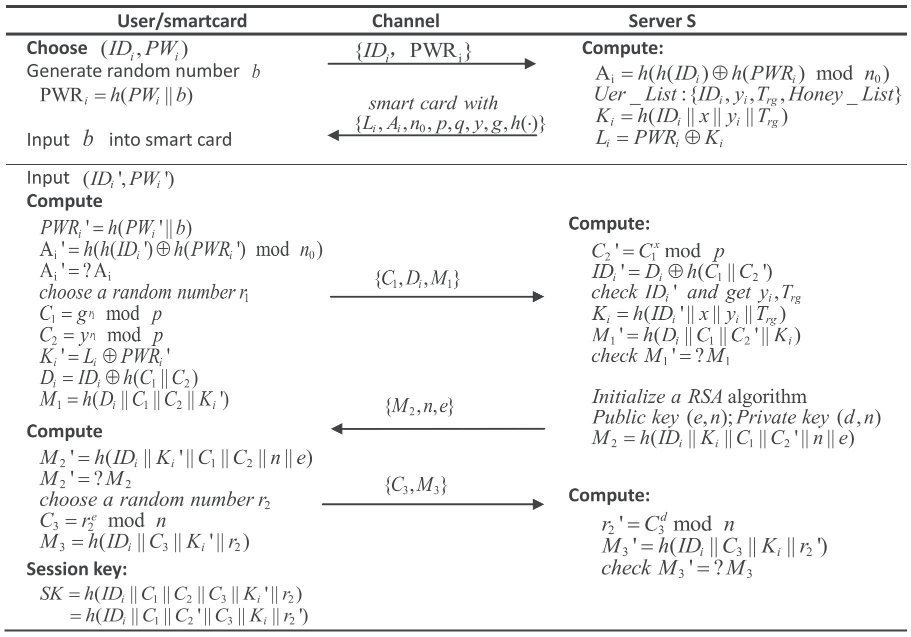
User/smartcard
Channel
Server S
Choose (ID_i, PW_i)
Generate random number b
$\mathrm{PWR}_i = h(PW_i \| b)$
Input b into smart card
$\{ID_i, \mathrm{PWR}_i\}$
smart card with $\{L_i, A_i, n_0, p, q, y, g, h(\cdot)\}$
Compute:
$\mathrm{A}_i = h(h(ID_i) \oplus h(PWR_i) \bmod n_0)$
$Uer\_List: \{ID_i, y_i, T_{rg}, Honey\_List\}$
$K_i = h(ID_i \| x \| y_i \| T_{rg})$
$L_i = PWR_i \oplus K_i$
Input (ID_i', PW_i')
Compute
$PWR_i' = h(PW_i' \| b)$
$\mathrm{A}_i' = h(h(ID_i') \oplus h(PWR_i') \bmod n_0)$
$\mathrm{A}_i' = ?\mathrm{A}_i$
choose a random number r_1
$C_1 = g^{r_1} \bmod p$
$C_2 = y^{r_1} \bmod p$
$K_i' = L_i \oplus PWR_i'$
$D_i = ID_i \oplus h(C_1 \| C_2)$
$M_1 = h(D_i \| C_1 \| C_2 \| K_i')$
$\{C_1, D_i, M_1\}$
Compute:
$C_2' = C_1^x \bmod p$
$ID_i' = D_i \oplus h(C_1 \| C_2')$
check ID_i' and get y_i, T_{rg}
$K_i = h(ID_i' \| x \| y_i \| T_{rg})$
$M_1' = h(D_i \| C_1 \| C_2' \| K_i)$
check $M_1' = ?M_1$
Initialize a RSA algorithm
Public key (e, n); Private key (d, n)
$M_2 = h(ID_i \| K_i \| C_1 \| C_2' \| n \| e)$
$\{M_2, n, e\}$
Compute
$M_2' = h(ID_i \| K_i' \| C_1 \| C_2 \| n \| e)$
$M_2' = ?M_2$
choose a random number r_2
$C_3 = r_2^e \bmod n$
$M_3 = h(ID_i \| C_3 \| K_i' \| r_2)$
$\{C_3, M_3\}$
Compute:
$r_2' = C_3^d \bmod n$
$M_3' = h(ID_i \| C_3 \| K_i \| r_2')$
check $M_3' = ?M_3$
Session key:
$SK = h(ID_i \| C_1 \| C_2 \| C_3 \| K_i' \| r_2)$
$= h(ID_i \| C_1 \| C_2' \| C_3 \| K_i \| r_2')$

FIGURE 1 The proposed scheme

1. U_i inserts the smart card into a card reader and inputs ID_i', PW_i', and new password PW_i^{new}.
2. The smart card computes $PWR_i' = h(PW_i' \| b)$, $A_i' = h(h(ID_i') \oplus h(PWR') \bmod n_0)$, check whether $A_i' =? A_i$ and, if the equation does not hold, exits the session. Otherwise, the smart card computes $PWR_i^{new} = h(PW_i^{new} \| b)$, $A_i^{new} = h(h(ID_i') \oplus h(PWR_i^{new}) \bmod n_0)$ and $L_i^{new} = L_i \oplus PWR_i' \oplus PWR_i^{new}$.
3. Replace L_i with L_i^{new} and A_i with A_i^{new}, respectively.

4.4 | Revocation phase

If the user U_i finds that his/her smart card is lost or is breached, he/she can revoke the account without changing the identity as follows:

1. U_i inserts the smart card into a card reader, inputs ID_i', PW_i'.
2. The smart card computes $PWR_i' = h(PW_i' \| b)$, $A_i' = h(h(ID_i') \oplus h(PWR') \bmod n_0)$, check whether $A_i' =? A_i$ and, if the equation does not hold, exits the session. Otherwise, the card generates a random number r_1, computes $C_1 = g^{r_1} \bmod p$, $C_2 = y^{r_1} \bmod p$, $D_i = ID_i \oplus h(C_1 \| C_2)$, $K_i' = L_i \oplus PWR_i'$, $M_1 = h(D_i \| C_1 \| C_2 \| K_i')$.
3. $U_i \rightarrow S$: $\{C_1, D_i, M_1, Revoke\_request\}$.
4. Upon getting $\{C_1, D_i, M_1, Revoke\_request\}$, S computes $C_2' = (C_1)^x \bmod p$, $ID_i' = D_i \oplus h(C_1 \| C_2)$. Then S searches the $User\_List$ to find the ID_i such that $ID_i = ID_i'$. If there is not an ID_i that satisfies the condition, S rejects the access request, and sets $Hoeny\_List$ to be $Hoeny\_List + 1$. Once the value of $Honey\_List$ exceeds the predetermined threshold (such as 10), S thinks that the smart card has been breached, thus suspends the card till U_i re-registers. Otherwise, S derives y_i and T_{rg} from the $User\_List$, computes $K_i = h(ID_i \| x \| y_i \| T_{rg})$, $M_1' = h(D_i \| C_1 \| C_2' \| K_i)$. Then S verifies U_i by testing whether $M_1' =?M_1$ and, if the equation is false, ends the session, sets $Hoeny\_List$ to be $Hoeny\_List + 1$, and once the value of $Honey\_List$ exceeds the predetermined threshold (such as 10), suspends the card till U_i re-registers; otherwise, S sets y_i to be $NULL$, thus next time, U_i cannot login successfully.

4.5 | Re-registration phase

If the account of U_i does not work properly, then U_i may want to re-register as follows:

1. $U_i \Rightarrow S$: $\{PWR_i, ID_i\}$.
2. S first checks whether the ID_i is in the Uer_List and whether the account of the U_i is revoked or his/her card is suspended. If so, S executes the registration as Section 4.1.

5 | SECURITY AND PERFORMANCE ANALYSIS

In this section, we first prove that our scheme can withstand all known attacks well, then, we compare our scheme with other related schemes in functionality and performance; the result demonstrates the superiority of our scheme.

5.1 | Security analysis

We prove our scheme can be resistant to known attacks by heuristic analysis of the scheme. This section lists almost all known attacks to the smart-based remote user authentication schemes and demonstrates how our scheme performs well in resisting against those attacks.

1. **Off-line password guessing attack.** Suppose the adversary $\mathcal{A}$ extracted the information in the smart card and eavesdropped the message between the user U_i and the server S, he may conduct the attack as following steps:

   Step 1. Guess PW_i to be PW_i^* and ID_i to be ID_i^*.
   Step 2. Compute $PWR_i' = h(PW_i^*\|b)$, $A_i^* = h(h(ID_i^*) \oplus h(PWR^*) \bmod n_0)$.
   Step 3. Verify the correctness of PW_i and ID_i by checking if $A_i^* =?A_i$.

   Even the adversary finds such PW_i' and ID_i' that satisfy the equation, he/she still is not sure whether they actually are same to PW_i and ID_i. Then the only way is to conduct an online guessing attack to test the value of those "correct" PW_i' and ID_i', while this attack is prevented by *Honey_List*. Each time the login fails, the value of *Honey_List* will add 1. Once the value of *Hoeny_List* exceeds the predetermined threshold (such as 10), the card will be suspended till U_i re-registers. Thus, the adversary cannot perform such attack by this process.

   Besides the methods above, there is usually another way as follows to execute the off-line password guessing attack:

   Step 1. Guess PW_i to be PW_i^* and ID_i to be ID_i^*.
   Step 2. Compute $PWR_i' = h(PW_i^*\|b)$, chose a random number $r_1{}^*$, computes $C_1^* = g^{r_1{}^*} \bmod p$, $C_2^* = y^{r_1{}^*} \bmod p$, $K_i^* = L_i \oplus PWR_i'$, $D_i^* = ID_i^* \oplus h(C_1^*\|C_2^*)$, and $M_1^* = h(D_i^*\|C_1^*\|C_2^*\|K_i^*)$.
   Step 3. Verify the correctness of PW_i and ID_i by checking if $M_1^* =?\ M_1$. While there are 3 uncertain parameters in M_1^*: PW_i^*, ID_i^*, r_1^*. The PW_i^* and ID_i^* is in a finite set, but the random number r_1^* is too strong to be guessed.

   Therefore, our scheme has good resistance to off-line password guessing attack.
2. **Smart card loss attack.** Suppose the adversary $\mathcal{A}$ stole the user U_i's smart card and extracted L_i, A_i, b in smart card. According to our analysis in "off-line password guessing attack," although with smart card and L_i, A_i, b, $\mathcal{A}$ cannot conduct an off-line password guessing attack. What is more, there are no other way to get the PW_i. Without knowing PW_i, even with smart card, the $\mathcal{A}$ still neither can construct M_1 to impersonate the user to the server nor performs other attacks. Therefore, our scheme can be resistant to smart card loss attack.
3. **User anonymity.** User anonymity protects the user activities from not being tracked by the adversary to analyze user habits. Usually, user anonymity refers to 2 aspects: (1) Do not expose the identity notion to an open channel and (2) keep the identity untraceable. While in our new protocol, on the one hand, there is no identity notions transmitted in the open channel; on the other hand, the identity ID_i is transmitted to the server in the form of D_i=$ID_i \oplus h(C_1\|C_2)$, where C_1 and C_2 are changed with the random number r_1. Each time the user logs in, the r_1 is different, so the D_i is also changed every time. Thus, the identity ID_i is untraceable. All in all, our scheme preserves user anonymity.
4. **Mutual authentication.** In our scheme, S authenticates U_i through testing whether M_1' equals to M_1; U_i verifies S by testing whether M_2' equals to M_2. So our scheme achieve mutual authentication.
5. **Replay attack.** Our scheme takes advantage of random numbers to prevent replay attack, and the random number is different in every session. On the one hand, the user and the server both will check the validity of the random number everytime they received a new message. On the other hand, even if the adversary replays the eavesdropped messages from the open channel, he/she still cannot construct the session key. Thus, our scheme is resistant to replay attack.
6. **User impersonation attack.** To impersonate a legitimate user, the adversary usually has 2 ways: getting the PW_i and ID_i and constructing $\{C_1, D_i, M_1\}$. The former way is impossible for $\mathcal{A}$ according to "off-line password guessing attack" and "smart card loss attack." So the later way becomes the only choice. The difficult point is building M_1, D. To achieve this, $\mathcal{A}$ has to get the key parameters K_i and ID_i, where K_i=$L_i \oplus PWR_i$=$h(ID_i\|x\|y_i\|T_{rg})$. Even assuming ID_i can be gotten by $\mathcal{A}$[23], it is still impossible to compute K_i, because PWR_i is related to PW_i and x is a secret key which cannot be known to $\mathcal{A}$. Thus, our scheme can be resistant to user impersonation attack.
7. **Server spoofing attack.** As x is a secret number, $\mathcal{A}$ cannot recover C_2 ($C_2 = (C_1)^x \bmod p$), thus, it fails to construct M_2 (M_2'=$h(ID_i\|K_i\|C_1\|C_2\|n\|e)$) to impersonate the server.
8. **Provision of key agreement.** In our scheme, after authenticating with each other successfully, the server and the user both accept the session key SK=$h(ID_i\|C_1\|C_2\|C_3\|K_i\|r_2)$, so our scheme builds the session key successfully. Furthermore, this key is changed with the random numbers.
9. **De-synchronization attack.** On the one hand, no parameters needs to be updated in the user side and the server side; on the other hand, whenever a user wants to change the password, he/she has to get authenticated firstly; finally, our scheme dose not require the server or the user to synchronize their time clock. Thus, a de-synchronization attack cannot be executed successfully.
10. **Forward secrecy.** Assuming $\mathcal{A}$ got x, the *User_ist* and all the parameters in the open channel and smart card,

he can recover the parameters like ID_i, C_1, C_2, C_3, and K_i. While computing r_2 to $\mathcal{A}$ means to break RSA security, which is impossible. Without r_2, $\mathcal{A}$ cannot compute SK as $h(ID_i||C_1||C_2||C_3||K_i||r_2)$. So the improved scheme provides strong forward secrecy.

11. **Insider attack.** In registration phase, the user U_i submits PWR_i=$h(PW_i||b)$ to the server S, the PW_i is conceal by random number b, thus an administrator of the server cannot get PW_i, ie, our scheme can withstand insider attack.
12. **Parallel attack.** The parallel attack usually happens when $\mathcal{A}$ reuses the historical messages in the open channel to construct a new request, then impersonates the legitimate user U_i to compute SK, making the server believe that it communicates with U_i. While in our scheme, $\mathcal{A}$ has to know the parameter consisted of the messages, then he can form the correct access request or the session key. While according to our analysis above, $\mathcal{A}$ is unable to get the 2 random numbers chosen by the user. Thus, our scheme can resist parallel attack.

5.2 | Performance analysis

We compare our scheme with other related scheme [24,32-38] in functionality and performance to manifest the advantages of our scheme as shown in Tables 4 and 5, respectively. Typically, according to Wang and Wang,[20], Wang et al,[23] and Maitra et al,[24] in cryptographic operations, the time complexity can be roughly expressed as $T_E > T_O \gg T_M \gg T_H > T_S$, and the lightweight operation such as "XOR" and "||" are too small that can be ignored, where T_E is the time of modular exponentiation operation, T_O is the running time of the elliptic curve point multiplication, T_M is the running time of modular multiplication/division operation, T_H is the running time for hash computation, and T_S is the running time of symmetric encryption/decryption. Thus, according to Table 5, our scheme only spends more total time than Kumari, Khan et al[32] and Kumari, Li et al.[33] These 2 schemes are only based on hash operation, which face with many security threats as shown in Table 4.

In conclusion, through the functionality comparison, it is clear that our scheme has obvious advantages: It satisfies all

TABLE 4 Functionality comparisons

Design Goals	Kumari[32]	Kumari[33]	Jiang[34]	Li[35]	Islam[36]	Marimuthu[37]	Maitra[24]	Xie[38]	Ours
No password verifier–table	√	√	√	√	√	√	√	√	√
Password friendly	√	√	×	√	√	√	√	×	√
No smart card loss attack	×	×	√	×	×	×	×	√	√
Resistance to know attacks	√	√	√	√	√	√	√	√	√
Sound repairability	×	×	×	×	√	×	×	√	√
User anonymity	√	×	×	×	×	√	√	×	√
Mutual authentication	√	√	√	√	√	√	√	√	√
Provision of key agreement	√	√	√	√	√	×	√	√	√
No clock synchronization	×	√	×	×	×	×	√	×	√
Forward secrecy	×	×	×	√	√	□	×	√	√
No password exposure	√	√	×	×	√	√	×	√	√
Timely typo detection	√	√	×	√	√	√	√	×	√

Note: √ means the property is satisfied;

× means the property is not satisfied;

□ means the property is not related to the scheme.

TABLE 5 Performance comparisons

Schemes	User side	Server side	Total
Kumari [32]	$9T_H$	$9T_H$	$18T_H$
Kumari [33]	$12T_H$	$9T_H$	$20T_H$
Jiang [34]	$3T_E + T_M + 4T_H$	$3T_E + 5T_H$	$6T_E + T_M + 9T_H$
Li [35]	$4T_E + T_M + 4T_H$	$5T_E + 5T_H$	$9T_E + T_M + 9T_H$
Islam [36]	$5T_E + T_M + 5T_H$	$2T_E + 4T_H$	$7T_E + T_M + 9T_H$
Marimuthu [37]	$6T_E + 9T_H$	$6T_E + 9T_H$	$12T_E + 18T_H$
Maitra [24]	$6T_E + T_M + 8T_H$	$4T_E + T_M + 12T_H$	$10T_E + 2T_M + 20T_H$
Xie [38]	$3T_O + 6T_H$	$3T_O + 3T_S + 6T_H$	$6T_O + 3T_S + 12T_H$
Ours	$3T_E + 10T_H$	$2T_E + 10T_H$	$5T_E + 20T_H$

the listed design goals, while other related schemes [24,35-37] have weaknesses more or less; in performance comparisons, our scheme costs less time than most of schemes.[24,34-38] Therefore, our scheme is more secure but more efficient.

6 | CONCLUSION

Preserving security and efficiency simultaneously is a challenging problem in designing smart card–based authentication schemes. Over the past 2 decades, massive schemes were proposed while later proved to be with various weaknesses. To make some changes to this situation, we first choose a recent scheme as a study case to analyze its weaknesses and design thinking. On the basis of these analysis, we proposed an improved new scheme. Particularly, we show a new way (beyond the conventional Deffie-Hellman approach) to achieve forward secrecy. Furthermore, the security and performance analysis results manifest the advantages of our scheme not only satisfies all 12 security requirements but also costs less time than most of related schemes. Our analysis in design thinking and fail reasons can provide references for analysis and design of other protocols.

ACKNOWLEDGEMENT

The authors thank the anonymous reviewers and the Editor for the constructive comments and generous feedback. This research was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61401038.

REFERENCES

1. Xia Z, Wang X, Zhang L, Qin Z, Sun X, Ren K. A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Trans Inform Foren Secur*. 2016;11(11):2594-2608.
2. Fu Z, Ren K, Shu J, Sun X, Huang F. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Trans Inform Foren Secur*. 2016;27(9):2546-2559.
3. Liu Q, Cai W, Shen J, Fu Z, Liu X, Linge N. A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. *Secur Commun Netw*. 2016;9(17):4002-4012.
4. Wang D, Wang P. On the anonymity of two-factor authentication schemes for wireless sensor networks: attacks, principle and solutions. *Computer Networks*. 2014;73(C):41-57.
5. Lamport L. Password authentication with insecure communication. *Commun ACM*. 1981;24(11):770-772.
6. Shimizu A. A danamic password authentication method using a one-way function. *Syst Comput Japan*. 1991;22(7):32-40.
7. Shieh S, Yang W, Sun H. An authentication protocol without trusted third party. *IEEE Commun Lett*. 1997;1(3):87-89.
8. Chen T, Lee W. A new method for using hash functions to solve remote user authentication. *Comput Electr Eng*. 2008;34(1):53-62.
9. Hwang T. Password authentication using public-key encryption. *Proc IEEE Int Carnahan Conf Security Technol*. 1983:141-4.
10. Chang CC, Wu TC. Remote password authentication with smart cards. *IEE Proc-Comput Dig Techniques*. 1991;138(3):165-168.
11. Huang X, Xiang Y, Chonka A, Zhou J, Deng RH. A generic framework for three-factor authentication: preserving security and privacy in distributed systems. *IEEE Trans Parallel Distrib Syst*. 2011;22(8):1390-1397.
12. He D, Zeadally S, Xu B, Huang X. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Trans Inform Foren Secur*. 2015;10(12):2681-2691.
13. He D, Zeadally S, Kummar NWW. Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures. *IEEE Trans Inform Foren Secur*. 2016;11(9):2052-2064.
14. Yu J, Ren K, Wang C, Huang X. Enabling cloud storage auditing with verifiable outsourcing of key updates. *IEEE Trans Inform Foren Secur*. 2016;11(6):1362-1375.
15. Odelu V, Das A, Goswami A. A secure biometrics-based multi-server authentication protocol using smart cards. *IEEE Trans Inform Foren Secur*. 2015;10(9):1953-1966.
16. Yuan C, XS. Fingerprint liveness detection based on multi-scale LPQ and PCA. *China Commun*. 2016;13(7):60-65.
17. He D, Wang D. Robust biometrics-based authentication scheme for multiserver environment. *IEEE Syst J*. 2015;9(3):816-823.
18. Das A. Analysis and improvement on an efficient biometric-based remote user authentication scheme using smart cards. *IET Inform Secur*. 2015;5(3):145-151.
19. Li X, Qiu W, Zheng D, Chen K, Li J. Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards. *IEEE Trans Ind Electron*. 2010;57(2):793-800.
20. Wang D, Wang P. Two birds with one stone: two-factor authentication with security beyond conventional bound. *IEEE Trans Depend Secur Comput*. 2016. https://doi.org/10.1109/TDSC.2016.2605087
21. Jiang Q, Ma J, Wei F. On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Syst J*. 2016. https://doi.org/doi:10.1109/JSYST.2016.2574719
22. Ma C, Wang D, Zhao S. Security flaws in two improved remote user authentication schemes using smart cards. *Int J Commun Syst*. 2014;27(10):2215-2227.
23. Wang D, He D, Wang P, Chu C. Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment. *IEEE Trans Depend Secur Comput*. 2015;12(4):428-442.
24. Maitra T, Obaidat M, Amin R, Islam S, Chaudhry S, Giri D. A robust elgamal-based password-authentication protocol using smart card for client-server communication. *Int J Commun Syst*. 2016. https://doi.org/doi:10.1002/dac.3242
25. Huang X, Xiang Y, Bertino E, Zhou J, Xu L. Robust multi-factor authentication for fragile communications. *IEEE Trans Depend Secur Comput*. 2013;11(6):568-581.
26. Wang D, Zhang Z, Wang P. Targeted online password guessing: an underestimated threat. *Proc. ACM CCS*. 2016:1242-1254.
27. Madhusudhan R, Mittal R. Dynamic id-based remote user password authentication schemes using smart cards: a review. *J Netw Comput Appl*. 2012;35(4):1235-1248.
28. Wang D, Wang P. On the implications of Zipf's law in passwords. *Proc ESORICS*. 2016:111-131.
29. Wei F, Ma J, Ma C, Li X. A two-factor authenticated key exchange protocol based on rsa with dynamic passwords. *Int J Embedded Syst*. 2015;7(3):257-265.
30. Li X, Niu J, Liao J, Liang W. Cryptanalysis of a dynamic identity-based remote user authentication scheme with verifiable password update. *Int J Commun Syst*. 2015;28(2):374-382.

31. Li X, Xiong Y, Ma J, Wang W. An enhanced and security dynamic identity based authentication protocol for multi-server architecture using smart cards. *J Netw Comput Appl*. 2012;35(2):763-769.

32. Kumari S, Khan M, Li X. An improved remote user authentication scheme with key agreement. *Comput Electr Eng*. 2014;40(6):1997-2012.

33. Kumari S, Li X, Wu F, Das A, Odelu V, Khan M. A user anonymous mutual authentication protocol. *KSII T Internet Inf*. 2016;10(9):4508-4528.

34. Jiang Q, Ma J, Li G, XL. Improvement of robust smart-card-based password authentication scheme. *Int J Commun Syst*. 2015; 28(2):383-393.

35. Li X, Niu J, Khan M, Liao J. An enhanced smart card based remote user password authentication scheme. *J Netw Comput* 2013;36(5):1365-1371.

36. Islam SH. Design and analysis of an improved smartcard-based remote user password authentication scheme. *Int J Commun Syst*. 2016;29(11):1708-1719.

37. Marimuthu K, Saravanan R. A secure remote user mutual authentication scheme using smart cards. *J Inform Secur Appl*. 2014;19:282-294.

38. Xie Q, Wong D, Wang G, Tan X, Chen K, Fang L. Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model. *IEEE Trans Inform Foren Secur*. 2017;12(6):1382-1392.

How to cite this article: Wang C, Wang D, Xu G, Guo Y. A lightweight password-based authentication protocol using smart card. *Int J Commun Syst*. 2017;e3336. https://doi.org/10.1002/dac.3336