Cryptanalysis and security enhancement of a remote user authentication scheme using smart cards

WANG Ding¹ ² (∇), MA Chun-guang¹

1. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
2. Automobile Management Institute of PLA, Bengbu 233011, China

Abstract

With the broad implementations of the electronic business and government applications, robust system security and strong privacy protection have become essential requirements for remote user authentication schemes. Recently, Chen et al. pointed out that Wang et al.’s scheme is vulnerable to the user impersonation attack and parallel session attack, and proposed an enhanced version to overcome the identified security flaws. In this paper, however, we show that Chen et al.’s scheme still cannot achieve the claimed security goals and report its following problems: (1) It suffers from the offline password guessing attack, key compromise impersonation attack and known key attack; (2) It fails to provide forward secrecy; (3) It is not easily repairable. As our main contribution, a robust dynamic ID-based scheme based on non-tamper resistance assumption of the smart cards is presented to cope with the aforementioned defects, while preserving the merits of different related schemes. The analysis demonstrates that our scheme meets all the proposed criteria and eliminates several grave security threats that are difficult to be tackled at the same time in previous scholarship.

Keywords cryptanalysis, authentication protocol, smart card, non-tamper resistant, forward secrecy

1 Introduction

In distributed networks, it is of great concern to protect the systems and the users’ privacy and security from malicious adversaries. Accordingly, user authentication becomes an essential security mechanism for remote systems to assure one communicating party of the validity of the corresponding party by acquisition of corroborative evidence. Generally speaking, techniques for user authentication are basically based on one or more of the following categories: (1) what a user knows, such as passwords; (2) what a user has, such as smart cards; (3) what a user is, such as fingerprints. Due to the advantages of simplicity, convenience, low cost, adaptability, portability and cryptographic capacity, the first two techniques to identify a user have been widely adopted in remote user authentication schemes.

Recently, dozens of password authentication schemes using smart cards have been proposed [1–14]. However, as noted in Refs. [7–9], there is no common set of desirable security properties that has been widely adopted for the construction of this type of schemes. What’s more, recently presented schemes are still having various security flaws being overlooked, and many of these schemes were demonstrated to be problematic and broken shortly after they were first put forward [1–6,10–13]. Problems in those protocols imply that the design and security analysis of such protocols remain a hard problem, which outlines the need for further research. In 2011, Chen et al. [11] proposed a new list of requirements for evaluating the goodness of password authentication scheme using smart cards, which consists of the following four criteria:

1) C1. Security: the scheme should withstand various possible malicious attacks, such as offline password guessing attack, replay attack and known key attack.
2) C2. Session key agreement: a session key should be
established to secure the subsequent data communications between the user and remote server.

3) C3. Mutual authentication: not only can the server verify the legal users, but also the users can check the validity of the corresponding server.

4) C4. Forward secrecy: even if long-term private key of one or more entities are compromised, the secrecy of previous session keys is not affected.

In Chen et al.'s [11] paper, they demonstrated that a recently proposed scheme by Wang et al. [12] is vulnerable to the guessing attack, forgery attack and denial of service attack. To eliminate these defects, they proposed an enhancement over Wang et al.'s scheme, and claimed that the new version is free from various attacks and can satisfy all the criteria listed above.

Unfortunately, in this paper, we will point out that Chen et al.'s scheme still suffers from the offline password guessing attack, key compromise impersonation attack and known key attack. Furthermore, their scheme is not easily repairable and still fails to provide forward secrecy. That is, Chen et al.'s scheme cannot fulfill the above requirements C1 and C4. To eliminate all the identified flaws, a robust authentication scheme based on the secure one-way hash function and the well-known large integer factorization problem is proposed. The fact that our scheme achieves the above four criteria under a weaker assumption, i.e. the non-tamper resistance assumption of the smart cards, is the highlight of our paper.

The remainder of this paper is organized as follows: in Sect. 2, we briefly review Chen et al.'s authentication scheme. Sect. 3 describes the weaknesses of Chen et al.'s scheme. Our proposed scheme is presented in Sect. 4, and its security analysis is given in Sect. 5. The comparison of the performance of our scheme with the other related schemes is shown in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Review of Chen et al.'s scheme

In this section, we examine the smart card based password authentication scheme proposed by Chen et al. [11] in 2011. Their scheme consists of four phases, namely the registration phase, the login phase, the verification phase and password update phase. For ease of presentation, we employ some intuitive abbreviations and notations listed as follows:

- $I_i$: Identity of user $U_i$;
- $P_i$: Password of user $U_i$;
- $x$: The secret key of remote server $S$;
- $y$: A random value corresponding to user $U_i$;
- $p, q, n$: $p$ and $q$ are two large prime numbers, and $n=pq$;
- $e, d$: $e$ is a prime number, such that $ed=1 \mod (p-1)$;
- $h()$: A cryptographic un-keyed hash function;
- $h_k()$: A cryptographic keyed hash function with secret key $k$;
- $\oplus$: The bitwise XOR operation;
- $\Rightarrow$: The string concatenation operation;
- $\rightarrow$: A common communication channel.

2.1 Registration phase

The registration phase involves the following operations:

**Step R1** $U_i$ chooses his/her identity $I_i$, password $P_i$, and a random number $b$.

$$U_i \Rightarrow S: \{I_i, h(h(b \oplus P_i))\}.$$ 

**Step R2** On receiving the registration message from $U_i$, the server $S$ computes $P=h(I_i \oplus x)$, $R=P \oplus h(b \oplus P_i)$ and $V=h_s(h(b \oplus P_i))$.

**Step R3** $S \Rightarrow U_i$: A smart card containing security parameters $\{V, R, h()\}$.

**Step R4** $U_i$ enters $b$ into his/her smart card.

2.2 Login phase

When $U_i$ wants to login to $S$, the following operations will be performed:

**Step L1** $U_i$ inserts his/her smart card into the card reader, and inputs $I_i$ and $P_i$.

**Step L2** The smart card computes $P = R \oplus h(b \oplus P_i)$ and $V' = h_s[h(h_b \oplus P_i)]$, and checks whether $V'$ equals the stored $V$. If the verification fails, the smart card terminates this session.

**Step L3** The smart card generates a random number $r$, and then computes $C_i= P \oplus h(r \oplus b)$ and $C_i=h_s[h (r \oplus b) \mid T_s]$, where $T_s$ is the current timestamp on user side.

**Step L4** $U_i \rightarrow S: \{I_i, C_1, C_2, T_s\}$.

2.3 Verification phase

After receiving the login request message from user $U_i$,
at time $T_u$, the server $S$ performs the following operations:

**Step V1** If either the format of $I_i$ is invalid or $T_u - T_s > \Delta T$, $S$ rejects the login request. If $(T_u - T_s) > \Delta T$, $S$ also rejects the login request.

**Step V2** $S$ computes $P = h(I_i \oplus x)$, $C_1' = P \oplus C_1$ and $C_2' = h_r(C_1' \| T_s)$, and then compares the computed $C_2'$ with the received $C_2$. If they are not equal, $S$ rejects the request.

**Step V3** $S$ computes $C_3 = h_r(C_1' \oplus T_s \| P)$, where $T_s$ denotes the current timestamp on server side.

**Step V4** $S \rightarrow U_i$: $\{C_3, T_s\}$.

**Step V5** Upon receiving the reply message $\{C_3, T_s\}$ at time $T_u$, $U_i$ checks the validity of $T_u$. If $(T_u - T_s) > \Delta T$ or $T_u = T_s$, $U_i$ terminates the session.

**Step V6** $U_i$ computes $C_3' = h_r(h_r(b \oplus b') \oplus T_s \| P)$, and then compares the computed $C_3'$ with the received $C_3$. If they are not equal, $U_i$ terminates the session.

After authenticating each other, $U_i$ and $S$ use the same session key $K = C_1' = h_r(b \oplus b')$ to secure subsequent data communications.

2.4 Password change phase

This phase is invoked whenever $U_i$ wants to change the old password $P_i$ to the new password $P_{i, new}$.

**Step P1** $U_i$ insert the smart card into card reader and enters $I_i$ and $P_i$ and requests to change password.

**Step P2** The smart card computes $P' = h_r(b \oplus P_i)$ and $V' = h_r(h_r(b \oplus P_i)$, and checks whether $V'$ equals the stored $V$. If the verification fails, the smart card rejects the request.

**Step P3** The smart card computes $R_{new} = h_r(h_r(b \oplus P_{i, new})$ and $V_{new} = h_r(h_r(b \oplus P_{i, new})$, and then updates $R$ and $V$ with $R_{new}$ and $V_{new}$, respectively.

3 Cryptanalysis of Chen et al.’s scheme

The general adversary model to evaluate the security of authentication protocols using smart cards assumes an attacker with full control over the communication channel between the user and the remote server [7,15]. In other words, all the messages exchanged can be blocked, intercepted, deleted, or modified by the attacker, and the attacker can also insert his/her own fabricated messages. Secondly, protocols must assume that the attacker can temporarily gain access to a legitimate user’s smart card, which is reasonable in practice. What’s more, since recent research results have demonstrated that the secret data stored in the common smart card could be extracted by some means, such as monitoring the power consumption [16–17] or analyzing the leaked information [18], the smart card should be supposed to be non-tamper resistant, i.e., all the secret data stored in the smart card can be revealed. And this assumption has already been widely made in recently published works [5–10,13–14].

In the following discussions of the security flaws of Chen et al.’s scheme, based on the three assumptions stated above, we assume that an attacker can extract the secret values $\{V, R, b\}$ stored in the legitimate user’s smart card, and the attacker can also intercept or block the login request message $\{I_i, C_1, C_2, T_s\}$ from $U_i$ to $S$ and the reply message $\{C_3, T_s\}$ from $S$ to $U_i$.

3.1 Offline password guessing attack

In Chen et al.’s scheme, a user is allowed to choose his/her own password $P_i$ at will during the registration and password change phases; the user usually tends to select a password, e.g., his phone number or birthday, which is easily remembered for his convenience. Hence, these easy-to-remember passwords, called weak passwords [19], have low entropy and thus are potentially vulnerable to offline password guessing attack, which is the most serious threat that an admired password protocol must be able to effectively cope with.

Let us consider the following scenarios. In case a legitimate user $U_i$’s smart card is somehow obtained (e.g. stolen or picked up) by an adversary, and the stored secret values such as $R$, $V$ and $b$ can be extracted by side-channel attacks. With a previously eavesdropped message $\{C_2, I_i, C_3, T_s\}$, the adversary can acquire $U_i$’s password $P_i$ by performing the following malicious attack procedure:

**Step 1** Guesses the value of a candidate password to be $P_i'$ from the password space $\mathcal{D}$.

**Step 2** Computes $P = R \oplus h(h_r(b \oplus P_i'))$, where the value of $b$ and $R$ are revealed from the smart card.

**Step 3** Computes $V' = h_r(h_r(b \oplus P_i'))$.

**Step 4** Verifies the correctness of $P_i'$ by checking if $V'$ equals the revealed $V'$.

**Step 5** Goes back to Step 1 of this phase until the correct value of $P_i$ is found.

Since the size of the password dictionary, i.e. $|\mathcal{D}|$, is very limited in practice, the above attack procedure can be
completed in polynomial time. Moreover, the above attack we describe is very effective because it only requires the abilities of an eavesdropping attacker, i.e. a passive attacker, and involves no special cryptographic operations. Hence, Chen et al.’s scheme cannot resist the offline password guessing attack.

After guessing the correct value of \( P_i \) and with the revealed \( b \) and \( R \), the adversary can compute the valid \( P = R \oplus h(b \oplus P_i) \). With this computed \( P \) and revealed \( b \), the attacker can fabricate and send a valid login request message \( \{ I_i, C_i, C_j, T_a \} \) to the server \( S \), where \( I_i \) is previously intercepted from the insecure channel and \( T_a \) denotes current timestamp of the attacker. Hence, the attacker can successfully impersonate the legitimate user \( U_i \) to login to \( S \) without the help of \( U_i \)’s smart card unless \( U_i \) re-registers to \( S \).

Moreover, once the adversary obtains the correct value of \( P_i \), with the intercepted \( I_i \), he/she can change the password to a new one, which causes the password change phase becoming insecure. Even if the adversary returns the changed smart card to the original user \( U_i \), \( U_i \) will not be able to login to the remote server \( S \). This leads to a denial of service attack.

### 3.2 Key compromise impersonation attack

In the case of key compromise impersonation [15,20], what concerns us is whether the knowledge of a communicating party \( A \)’s private key allows a malicious adversary \( M \) not only to impersonate \( A \) to others but also to impersonate other uncorrupted entities to \( A \). Schemes that can thwart this kind of reverse impersonation are said to resist key compromise impersonation attack.

Suppose the long-time secret key \( x \) of the server \( S \) is leaked out by accident or intentionally stolen by the adversary \( M \). Without loss of generality, we assume the login request \( \{ I_i, C_i, C_j, T_a \} \) of one previous session, namely \( j \)th session, is intercepted by \( M \). Once the value of \( x \) is obtained, with the intercepted \( I_i \) and \( C_i \), \( M \) can impersonate the legitimate user \( U_i \) through the following method:

1. **Step 1** Computes the parameter \( P = h(I_i \oplus x) \).
2. **Step 2** Computes \( j \)th session key \( K_j = h(r_j \oplus b) = C_i \oplus P \).
3. **Step 3** Fabricate \( C_j' = h_p(h(r_j \oplus b) \| T_a) = h_p(C_i \oplus P \| T_a) \), where \( T_a \) is the current timestamp on client side.
4. **Step 4** Sends \( \{ I_i, C_i', C_j', T_a \} \) to server \( S \).

**Step 5** Ignores the reply from \( S \) and computes the session key \( K_j = h(r_j \oplus b) = C_i \oplus P \).

Since the correct value of \( P \) has been obtained by adversary \( M \), it is easy to see that the following relationship holds true: \( C_i' = h_p(C_i' \| T_a) = h_p(C_i \oplus P \| T_a) \). Therefore, server \( S \) will accept \( M \) login request and sends back a reply. By generalizing the above attack, \( M \) can easily imitate any user to login to \( S \) at any time without employing any cryptographic techniques.

It should be noted that, with the knowledge of \( P = h(I_i \oplus x) \), an adversary can also fabricate the valid \( C_i \) and compute session key \( K \) in a similar way as described above. That is, the adversary can also successfully launch a server masquerading attack.

Hence, Chen et al.’s scheme cannot withstand the key compromise impersonation attack.

### 3.3 Known key attack

As noted in Ref. [20], the resistance to known key attack is a basic security requirement of key agreement schemes.

What concerns us is the realistic possibility that some session-specific information, e.g. a session key or the ephemeral secrets that lead to the derivation of the session key, will leak to an adversary. This can happen in various ways, such as the simple mishandling of information, the malicious action of an insider and a temporary break-in. In this case, the exposed session is definitely insecure, but what a well-designed key-agreement protocol should guarantee is, such an exposure will only affect the specific compromised session. Other sessions, established by the same or other parties, should not be endangered by this leakage.

Suppose a legitimate user \( U_i \)’s \( j \)th session key, denoted by \( K_j = h(r_j \oplus h) \), is leaked. Once an adversary \( M \) obtains \( K_j \), with \( C_j \) previously intercepted during the verification phase of \( j \)th session, he/she can successfully derive the secret parameter \( P = h(I_i \oplus x) = C_i' \oplus K_j \). With this critical parameter \( P \) and following the attack procedure introduced in Sect. 3.2, \( M \) can successfully impersonate the legitimate user \( U_i \) to login to \( S \) unless \( U_i \) re-registers.

### 3.4 No provision of forward secrecy

As with resistance to key-compromise impersonation attack, the feature of forward secrecy is also concerned
with limiting the effects of ultimate failures, in case the disclosure of server’s long-term private keys [19].

Let us consider the following scenarios. Suppose the server S’s long-term private key x is leaked out by accident or intentionally stolen by an adversary M. Once x is obtained, with previously intercepted jth authentication message \( \{I_j, C_j, T_j\} \), M can derive the session key \( SK_j \) of S and \( U_i \)’s jth encrypted communication as follows:

**Step 1** computes the parameter \( P = h(I_j \oplus x) \).

**Step 2** computes jth session key \( SK_j = h(r_j \oplus b) = C_j \oplus P \).

Once the session key \( SK_j \) is obtained, the entire jth session will become completely insecure. Consequently, the property of forward secrecy is not provided in Chen et al.’s scheme, while the provision of forward secrecy is a basic requirement for a secure key agreement scheme.

### 3.5 Failure of preserving user anonymity

As violation concern of user privacy is promptly raised among individuals, human-rights organizations and government agencies, especially in some authentication scenarios, e.g. electronic voting or secure online-order placement, it is very important to preserve the privacy of a user. In addition, the leakage of the user specific information may also cause an unauthorized entity to track the user’s login history and current location [6,21]. Therefore, assuring anonymity does not only preserve user privacy but also makes remote user authentication protocols more secure.

In Chen et al.’s scheme, the user’s identity I is static in all the login phases, which may leak the identity of the logging user once the login messages were eavesdropped, hence user anonymity is not preserved.

### 3.6 Poor reparability

In 2009, Chung et al. [22] pointed out that Wang et al.’s scheme [14] is not easily repairable once the critical parameter \( P \) corresponding to \( U_i \) is compromised. Unfortunately, we find Chen et al.’s scheme inherits this defect and thus is also not easily repairable when the parameter \( P \) is obtained by an adversary. As described in the previous sections, there are a variety of ways that may lead to the leakage of \( P \), such as the compromise of the \( U_i \)’s password or the leakage of a session key.

Impersonation attacks cannot be restricted and stopped even if \( U_i \) has detected that \( P \) has been compromised and then used a new password to re-register with S. As the value of \( P \) is determined only by \( U_i \)’s identifier \( I_i \) and S permanent secret key x, S cannot change \( P \) for \( U_i \) unless \( I_i \) or x can be changed. However, since x is commonly used for all users rather than specifically used for only \( U_i \), it is unreasonable and inefficient if x should be changed to recover the security of \( U_i \) only. Additionally, it is also impractical to change \( I_i \), which should be tied to \( U_i \) in most application systems. Thus, Chen et al.’s scheme is not easily repairable.

### 3.7 Redundancy in the verification phase

Since the structure of \( C_2 \) and \( C_3 \) is completely different, the threat of parallel session attack and reflection attack is completely eliminated, and thus there is no need to check whether \( T_u = T_u \) in Step V1 of the verification phase. So is the case with Step V5 of the verification phase.

### 4 Our proposed scheme

In this section, we present an enhanced remote user authentication scheme to satisfy all the four criteria listed in Sect. 1 and the abbreviations and notations used in the following sections are listed in Sect. 2. According to our analysis, three principles for designing a sound password-based remote user authentication scheme are presented. First, user anonymity, especially in some application scenarios, (e.g., e-commerce), should be preserved, because from the identity \( I_i \), some personal information may be leaked about the user. Second, to achieve forward secrecy and known key security, the formation of a session key shall include both the long-term private keys and session-specific transient secrets. Finally, the password change process should be performed locally without the hassle of interaction with the remote authentication server for the sake of security, user friendliness and efficiency [6]. Accordingly, our new scheme proceeds as follows.

#### 4.1 Registration phase

The server S generates two large primes \( p \) and \( q \) and computes \( n = pq \), then chooses a prime number \( e \) and an integer \( d \), such that \( ed \equiv 1 \mod (p-1)(q-1) \). Finally, the server S makes the values of \( n \) and \( e \) public, while \( p \), \( q \) and
d are only known to server S. The registration phase involves the following operations:

**Step R1**: \( U_i \) chooses his/her identity \( I_i \), password \( P_i \), and a random number \( b \).

**Step R2**: \( U_i \rightarrow S: \{ I_i, h(b \oplus P_i) \} \).

**Step R3**: On receiving the registration message from \( U_i \), the server \( S \) chooses a random value \( y \), and computes \( \text{new} = h(I_i \| h(b \oplus P_i)) \oplus d \) and \( I_i = h(b \oplus P_i) \| I_i \| h(y) \). \( B_i = y \oplus I_i \oplus h(b \oplus P_i) \) and \( D_i = h(h(d) \| y) \oplus d \). Server \( S \) chooses the value of \( y \), corresponding to user \( U_i \) to make sure \( D_i \) is unique for each user. The server \( S \) stores \( y \oplus h(d) \| d \) and \( I_i \oplus h(d) \| y \) corresponding to \( D_i \) in its backend database.

**Step R4**: \( S \rightarrow U_i: \{ C_1, C_2, A_i, S, n, e, h(\cdot) \} \).

**Step R5**: \( U_i \) enters \( b \) into his/her smart card.

### 4.2 Login phase

When \( U_i \) wants to login to \( S \), the following operations will be performed:

**Step L1**: \( U_i \) inserts his/her smart card into the card reader, inputs \( I_i \) and \( P_i \).

**Step L2**: The smart card computes \( y = B_i \oplus I_i \oplus h(b \oplus P_i) \) and \( A_i = h(b \oplus P_i) \| I_i \| h(y) \). Smart card verifies the validity of \( A_i \) by checking whether \( A_i \) equals the stored \( A_i \). If the verification fails, the smart card computes \( d = N_a \oplus h(I_i \| h(b \oplus P_i)) \), \( I_{\text{old}} = h(I_i \| y) \| h(d) \| N_a \| T_u \), \( C_1 = N_a \mod n \) and \( C_2 = h(I_i \| h(d) \| y) \| T_u \| N_a \), where \( T_u \) is current timestamp. Otherwise, the session is terminated.

**Step L3**: \( U_i \rightarrow S: \{ I_{\text{old}}, C_1, C_2, T_u \} \).

### 4.3 Verification phase

After receiving the login request message from user \( U_i \) at time \( T_w \), server \( S \) performs the following operations:

**Step V1**: The server \( S \) checks the validity of timestamp \( T_u \) by checking \( (T_u - T_w) \leq \Delta T \), where \( T_w \) is the current timestamp of server \( S \) and \( \Delta T \) is the permissible time interval for a transmission delay.

**Step V2**: The server \( S \) decrypts the random number \( N_a \) from \( C_1 \) using its private key \( d \), then computes \( D_i = h(I_{\text{old}} \oplus h(d) \| N_a \| T_u) \) and finds \( D_i \) corresponding to \( D_i \) in its database. If \( D_i \) is not found, \( S \) rejects. Otherwise, \( S \) extracts \( y \| h(d) \| d \) and \( I_i \| h(d) \| y \) corresponding to \( D_i \) from its database. Now the server \( S \) computes \( y \| h(d) \| d \) and \( I_i \| h(d) \| y \) because the server \( S \) knows the value of \( d \).

**Step V3**: The server \( S \) computes \( C_i = h(I_i \| h(d) \| y) \| T_u \| N_a \) and compares \( C_i \) with the value of received \( C_i \). This equivalency authenticates the legitimacy of the user \( U_i \) and the login request is accepted else the connection is terminated.

**Step V4**: The server \( S \) computes the session key \( K = h(I_i \| h(d) \| y) \| T_u \| T_s \| N_a \) and \( C_i = h(h(d) \| I_i \| y) \| T_u \| N_a \| K) \), where \( T_s \) is the current timestamp at server side.

**Step V5**: \( S \rightarrow U_i: \{ C_3, T_s \} \).

**Step V6**: Upon receiving the reply message \( \{ C_3, T_s \} \) at time \( T_u \), \( U_i \) checks the validity of \( T_s \). If \( (T_u - T_s) > \Delta T \), \( U_i \) terminates the session.

**Step V7**: \( U_i \) computes \( SK' = h(I_i \| h(d) \| y) \| T_u \| T_s \| N_a \) and \( C_i' = h(h(d) \| I_i \| y) \| T_u \| N_a \| K') \), and then compares the computed \( C_i \) with the received \( C_i \). If they are not equal, \( U_i \) terminates the session.

**Step V8**: After authenticating each other, \( U_i \) and \( S \) use the same session key \( K' = h(I_i \| h(d) \| y) \| T_u \| T_s \| N_a \) to secure subsequent data communications.

### 4.4 Password change phase

This phase is invoked whenever \( U_i \) wants to change the old password \( P_i \) to the new password \( P_i' \).

**Step P1**: \( U_i \) inserts the smart card into the card reader and enters \( I_i \) and \( P_i \), and requests to change password.

**Step P2**: The smart card computes \( y = B_i \oplus I_i \oplus h(b \oplus P_i) \), \( A_i' = h(b \oplus P_i) \| I_i \| h(y) \). Smart card verifies the validity of \( A_i' \) by checking whether \( A_i' \) equals the stored \( A_i \). If the verification fails, the smart card rejects the request. If the failure times exceed a predefined value, smart card locks immediately.

**Step P3**: The smart card computes \( N_a' = N_a \oplus h(I_i \| h(b \oplus P_i) \| h(b \oplus P_i') \| h(b \oplus P_i') \| h(b \oplus P_i') \| h(b \oplus P_i') \) and then updates \( N_a \), \( A_i \), and \( B_i \) with \( N_a', A_i' \) and \( B_i' \), respectively.

### 4.5 User revoking phase

When user \( U_i \) wants to revoke a lost smart card, the client still can use the previous password and the identity to re-register. This time, the server \( S \) only needs to choose a new random number \( y \) corresponding to \( U_i \), and then computes the new \( N_a, A_i, B_i \) and \( D_i \). The other parts of this phase are the same with that of the registration phase.
5 Security analysis

The security of our proposed authentication scheme is based on the secure hash function and the difficulty of the large integer factorization problem. In this section, we analyze the security features provided by our scheme under the assumption that the secret information stored in the smart card can be revealed, i.e., \( N_s, A_i \), and \( B_i \) can be obtained by a malicious privileged user. Accordingly, a malicious privileged user \( U_i \) having his own smart card can find out the value of \( h(d) \) as \( h(d) = N_i \oplus h[I_i \| h(b \oplus P_i)] \) because the malicious user \( U_i \) knows his own \( I_i \) and \( P_i \) corresponding to his own smart card.

1) User anonymity: suppose that the attacker has intercepted \( U_i \) authentication messages \((I_{CID}, C_1, C_2, C_3)\).

2) Offline password guessing attack: suppose that a malicious privileged user \( U_i \) has got another legitimate user, say \( U_k \) smart card, and the secret information \( b, N_k, A_k \) and \( B_k \) can also be revealed under our assumption of the non-tamper resistant smart card. Even after gathering this information and obtaining \( h(d) = N_i \oplus h(b \oplus P_i) || I_i \), the attacker has to at least guess both \( I_k \) and \( P_k \) correctly at the same time, because it has been demonstrated that our scheme can provide user identity confidentiality. It is impossible to guess these two parameters correctly at the same time in polynomial time, and thus the proposed scheme can resist offline password guessing attack with smart card security breach.

3) Stolen verifier attack: in the proposed protocol, only the server \( S \) knows private secret \( d \) and stores \( y_i \oplus h(h(d)[||d] \) and \( I_i \oplus h(d)|||y_i \) corresponding to \( D_i \) in its database. Although a malicious privileged user can compute \( h(d) \) in the way described above, he/she does not have any technique to find out the value of \( d \), nor can he/she calculates \( y_i \) corresponding to other legitimate user. Therefore, the proposed protocol is secure against stolen verifier attack.

4) User impersonation attack: as \( I_{CID}, C_1, C_2 \) and \( C_3 \) are all protected by secure one-way hash function, any modification to these parameters of the legitimate user \( U_i \) authentication messages will be detected by the server \( S \) if the attacker cannot fabricate the valid \( C_2^* \) and \( C_3^* \). Because the attacker has no way of obtaining the values of \( I_i, P_i \) and \( y_i \) corresponding to user \( U_i \), she cannot fabricate the valid \( C_2^* \) and \( C_3^* \). Therefore, our protocol is secure against user impersonation attack.

5) Server masquerading attack: in the proposed protocol, a malicious server cannot compute the session key \( K = h(I_i \| h(d) \| y_i \| T_u \| T_s \| N_s) \) and \( C_2 = h(I_i \| h(d) \| y_i \| T_u \| N_u) \) because the malicious server does not know the values of \( I_i \) and \( y_i \) corresponding to user \( U_i \), and has to solve the large integer factorization problem to retrieve \( N_u \). Therefore, the proposed protocol is secure against server masquerading attack.

6) Replay attack and parallel session attack: our scheme can withstand replay attack because the authenticity of authentication messages \((I_{CID}, C_2, C_3)\) is verified by checking the freshness of timestamp \( T_u \) and/or \( T_s \). On the other hand, the presented scheme resists parallel session attack, in which an adversary may masquerade as a legitimate party by replaying a previous authentication message intercepted from another session. Since authentication messages in different sessions employ different fresh timestamps and random numbers, the messages intercepted from other sessions will be definitely found invalid. Therefore, the resistance to replay attack and parallel session attack can be guaranteed in our protocol.

7) Mutual authentication: in our dynamic ID-based scheme, the server authenticates the user by checking the validity of \( C_2 \) in the access request. We have shown that our scheme can preserve user anonymity, so user \( I_i \) is only known to the server \( S \) and the user \( U_i \) itself. We have proved that our scheme can resist user impersonation attack. Therefore, it is impossible for an adversary to forge messages to masquerade as \( U_i \) in our scheme. To pass the authentication of server \( S \), the smart card first needs \( U_i \) identity \( I_i \) and password \( P_i \) to get through the verification in Step L2 of the login phase. In this Section, we have shown that our scheme can resist offline password guessing attack. Therefore, only the legal user \( U_i \) who knows correct \( I_i \) and \( P_i \) can pass the authentication of server \( S \). On the other hand, the user \( U_i \) authenticates server \( S \) by explicitly checking whether the other party communicating with can obtain the correct session key \( K = h(I_i \| h(d) \| y_i \| T_u \| T_s \| N_u) \)
and compute the valid $C_3$ or not. Since the malicious server does not know the values of $N_i$, $I_i$ and $y_i$ corresponding to user $U_i$, only the legitimate server can compute the correct session key $K$ and $C_3$. From the above analysis, we conclude that our scheme can achieve mutual authentication.

8) Denial of service attack: assume that an adversary has got a legitimate user $U_i$’s smart card. However, in our scheme, the security parameters stored in the server $S$ do not need to be updated during every authentication phase. Furthermore, the smart card checks the validity of user identity $I_i$ and password $P_i$ before the password update procedure. Since the smart card computes $A_i^j = h(h(b \oplus P_i) || I_i) \oplus h(y_i)$ and compares it with the stored value of $A_i$ in its memory to verify the legality of the user before the smart card accepts the password update request, it is not possible for the adversary to guess out $I_i$ and $P_i$ correctly at the same time in polynomial time. Accordingly, once the number of login failure exceeds the predefined system value, the smart card will be locked immediately. Hence, the proposed protocol is free from denial of service attack.

9) Key compromise impersonation attack: suppose the long-time private key $d$ of the server $S$ is leaked out by accident or intentionally stolen by the adversary $M$. Without loss of generality, we assume the authentication messages $\{I_{CIDS}, C_1^i, C_1^j, C_2^i, C_2^j, T_i^j\}$ of one previous session, namely $j$th session, is also intercepted by $M$. Once the value of $d$ is obtained, with the intercepted $C_1^i$, $M$ can derive $N_i$. However, $M$ has no way to obtain the critical parameter $y_i$ as $y_i$ is always concealed in non-invertible hashed form. Without $y_i$, $M$ cannot fabricate valid $C_2$ and $K$, and thus the threat of impersonation attack is eliminated.

10) Known key attack: since neither the structure of session key $K$ is the same with any other authentication message, nor $K$ functions as part of any other authentication message, the leakage of $K$ does not affect other unexposed sessions.

11) Forward secrecy: in our scheme, the session key $K$ is computed with the contribution of identity $I_i$ and security parameter $y_i$, thus the attacker cannot compute the previously generated session keys without the knowledge of the correct value of $I_i$ and $y_i$ corresponding to user $U_i$, even the attacker knows the server $S$ long time private key $d$. As a result, our scheme provides the property of forward secrecy.

### 6 Performance evaluation

In this section, we compare the performance and security features of our proposed scheme with that of related schemes which are also based on non-tamper resistance assumption of the smart cards to evaluate our scheme. The comparison results are depicted in Tables 1 and 2, respectively.

#### Table 1 Performance comparison among relevant authentication schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total computation cost</td>
<td>$2T_S+17T_E$</td>
<td>$8T_E$</td>
<td>$6T_E+8T_R$</td>
<td>$1T_S+2T_E+8T_R$</td>
<td>$3T_S+10T_E$</td>
<td>$6T_S+5T_E$</td>
<td>$3T_E+6T_R$</td>
</tr>
<tr>
<td>Communication cost/bit</td>
<td>1664</td>
<td>768</td>
<td>2816</td>
<td>866</td>
<td>256</td>
<td>2650</td>
<td>1664</td>
</tr>
<tr>
<td>Storage overhead/bits</td>
<td>2560</td>
<td>384</td>
<td>3200</td>
<td>256</td>
<td>3200</td>
<td>3200</td>
<td>1280</td>
</tr>
</tbody>
</table>

#### Table 2 Security features comparison among relevant authentication schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserving user anonymity</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to offline password guessing attack</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to stolen verifier attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to user impersonation attack</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to server masquerading attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to parallel session attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to denial of service attack</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to password disclosure to server</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to known key attack</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resistance to key compromise impersonation attack</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Session key agreement</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Forward secrecy</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Xu et al.’s scheme is found vulnerable to user impersonation attack by Song in 2010 [3].
2. An attacker can simply modify the last exchange in the password change phase, which leads to a denial of service attack on their scheme [14].
3. Tsai et al.’s scheme is found no provision of forward secrecy by Wu et al. in 2012 [9].
An efficient authentication scheme must take computation cost, communication overhead and storage cost into consideration. We mainly focus on the efficiency of login and verification phases since these two phases are the main body of an authentication scheme and are executed much more frequently than the other phases. Without loss of generality, the identity $I$, password $P$, random numbers, timestamp values and output of secure one-way hash function are all recommended to be 128-bit long, while $n$, $e$ and $d$ are all 1 024 bit long. Let $T_h$, $T_E$, $T_S$ and $T_X$ denote the time complexity for hash function, exponential operation, symmetric cryptographic operation and XOR operation, respectively. Since the time complexity of XOR operation is negligible as compared to the other three operations, we do not take $T_X$ into account. Typically, time complexity associated with these operations can be roughly expressed as $T_E > T_H > T_S$ [23–24].

In our scheme, the parameters [$N_i$, $A_i$, $B_i$, $n$, $e$, $d$] are stored in the smart card, thus the storage cost is 2 560 (= $4 \times 128 + 2 \times 1 024$) bit. The communication overhead includes the capacity of transmitted message involved in the authentication process, which is 1 664 (= $5 \times 28 + 1 \times 1 024$) bit. During the login and verification phase, the total computation cost of the user and server is $2T_E + 17T_H$. The proposed scheme is more efficient than Xu et al.’s scheme, Tsai et al.’s scheme and Chen et al.’s scheme [4], and enjoys nearly the same performance as Kim et al.’s scheme. As compared to Chen et al.’s scheme [11] and Song et al.’s scheme, to resolve all the identified defects, the decrease of some efficiency is unavoidable.

In particular, since smartcards are usually characterized as resource-constrained and low-powered devices, computation cost at the user end is always regarded as a key criterion for smartcard-based schemes. In our scheme, the computation cost at user end is $T_E + 9T_H$ during the login and verification phases. Clearly, $T_E$ is the most time-consuming operation and contributes the main overhead at user end. What needs further investigation is that, in practice, the encryption exponent $e$ of the exponential operation is often very limited, such as $e=3$ or $e=7$, and a widely accepted, secure enough encryption exponent is $e=2^{16}+1$ [24]. As the studies in Refs. [25–27] suggest, when the encryption exponent $e$ is much smaller than the modular $n$, the time taken for pure computation is significantly shorter than that of common exponential operation with big exponents. Let’s take the embedded RISC processors, which are dominant in smart card products, for example. On a SmartMIPS-4KSc 33 MHz processor, a 1 024 bit common modular exponentiation takes 320 ms, while this result is only 2.28 ms when $e=7$ [26]. Both theory and practice have shown the feasibility and acceptability to conduct one exponential operation with small exponent in resource-limited environments.

Table 2 gives a comparison of the admired features of our proposed scheme with the other relevant authentication schemes. Our proposed scheme can resist all sophisticated attacks, which indicates the satisfaction of criterion C1, while the other six latest schemes suffer from several security vulnerabilities; our proposed scheme provides criterion C4, i.e. forward secrecy, while all the other schemes except Xu et al.’s fail to achieve this feature; our proposed scheme and Tsai et al.’s scheme preserve user anonymity, while the rest of the schemes do not provide this property. Since the criteria C2 and C3 are basic requirements for practical authentication schemes, all the seven schemes achieve these two criteria. In addition, our improved scheme is easily reparable. It is clear that our scheme meets more criteria as compared to other relevant authentication schemes using non-tamper resistant smart cards.

It should be noted that, in our scheme, server $S$ maintains an account-database, which contains users’ security parameters for authentication. If the adversary performs any unauthorized modifications on the account-database, the data will become inconsistent and the system may be crumbled. Thus, special security measures should be taken to eliminate such risks. Fortunately, the countermeasure is not complicated: $S$ can routinely and frequently make offsite backup of the account-database and check the consistency, and restore the account-database by using the offsite backup when necessary. Thus, there is a trade-off between performance and functionality in our scheme, while this trade-off is inevitable for authentication schemes with provision of sound reparable [22]. What’s more, it is important to notice that, although our scheme and the relevant schemes discussed are all based on timestamp mechanism to resist replay attacks, they can be easily transformed into a nonce-based design [24], with which the clock synchronization problem can be eliminated.
7 Conclusions

It is a challenge to design a practical password-based authentication scheme using non-tamper resistant smart cards, because the designers are faced with the difficult task of reconciling security, functionality and efficiency requirements and sometimes must make design decisions that appear well motivated but have unintended consequences. More recently, Chen et al. showed that Wang et al.’s secure remote user authentication scheme cannot defend against various attacks and then proposed an improved version. However, in this paper, we have demonstrated that, besides some practical pitfalls, Chen et al.’s scheme suffers from the offline password guessing attack, key compromise impersonation attack and known key attack, and fails to provide forward secrecy. As to our main contribution, a robust dynamic ID-based authentication scheme is proposed to remedy these identified flaws, the security and performance analysis demonstrate that our presented scheme achieves more security requirements with high efficiency and thus our scheme is more secure and efficient for practical application environments.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (61170241, 61073042).

References

From p. 98


(Editor: WANG Xu-ying)