SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:1552–1559
Published online 22 March 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.768

SPECIAL ISSUE PAPER

Robust password changing and DoS resilience for human-centric password authentication

Xiangxue Li[1,2]*, Haifeng Qian[1], Yu Yu[3], Jian Weng[4] and Ziping Wang[5]

[1] Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China
[2] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[3] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China
[4] Department of Computer Science, JiNan University, Guangzhou 510632, China
[5] Department of Information Science and Systems, Morgan State University, Maryland 21251, U.S.A.

ABSTRACT

In password-based or two-factor (password and smart card) authentications, password changing is one of common techniques used to improve the security of the systems protected by the password. However, the password-changing operations in existing password authentications either depend on the login phase or violate the common practice that an old password should not be valid for subsequent login after being updated. On the other hand, password mistyping is very common in reality, which may be random or be skewed by the adversary via technical means or social engineering manipulation [i.e., a kind of denial-of-service (DoS) attack]. In human-centric authentication mechanisms, password changing and DoS resilience are not marginal issues. The paper addresses the requirements of robust password changing in authentication and presents $\mathcal{SPCA}$, a password authentication scheme with robust password changing, DoS resilience, and card-compromise security. Thus, the proposal can be viewed as a suitable candidate instantiation for authentication services of human-centric security, by embedding in the computer and software systems. $\mathcal{SPCA}$ also achieves other appealing features, such as self-healing ability and strong privacy protection, which may be useful for human-centric applications. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS

password changing; DoS resilience; self-healing

***Correspondence**

Xiangxue Li, Department of Computer Science and Technology, East China Normal University, Dongchuan Road 500, Shanghai 200241, China.
E-mail: xxli@cs.ecnu.edu.cn

1. INTRODUCTION

To meet various security demands in computer networks, a number of security measures are well studied and employed [1–3]. Because smart cards are small enough, comparatively cheap, well standardized, and very tamper proof, many researchers explored smart card-based security [4,5].

Entity authentication is the act of confirming the truth of an identity of an entity. Generally, there are three factors of authentication: something the users have (say, smart card), something the users know (say, password), or something the users are. One can use any factor of authentication alone or in combination with other authentication factors to have a stronger authentication, meaning that without any of the combined factors, authentication cannot take place successfully. The authentication process is embedded in the computer and software systems, by many different approaches. Two-factor authenticated key agreement schemes are amongst the most commonly used, where servers exchange keys with clients (or users) who use memorized short passwords and long cryptographic keys stored on smart cards to login [6–8]. Security of these authentication mechanisms relies on the assumption that no one can compromise the two factors simultaneously. The authentication should remain secure even if any one of the two factors is compromised [3,5–8].

Motivation. One can easily find in the literature many password-based authentication schemes that provide the functionality of password-changing operation. Password changing is indeed one of the common techniques used to improve the security of the systems protected by a

password. However, the password-changing operations in existing password authentications either depend on the successful run of a login phase, for example, [9,10], or violate the common practice that an old password should not be valid for subsequent login after being updated, for example, [11]. Refer to Section 2 for more details.

On the other hand, password mistyping is very common in reality, which may be random or be skewed by the adversary via technical means or social engineering manipulation [a kind of denial-of-service (DoS) attack] [12]. When the user keys in the password, mistyping (no matter random or being skewed by the adversary) may happen, and the card reader treats some other value, as the expected one.

Therefore, for authentication services of human-centric security, we should consider the capabilities of password changing and DoS resilience at the same time. The paper addresses the requirements for updating password and presents $\mathcal{SPCA}$, a password authentication with DoS resilience and card-compromise security. The idea behind $\mathcal{SPCA}$ is to combine the authenticated Diffie–Hellman technique [13] (which defeats the men-in-the-middle attack) and a variant of El Gamal encryption with provable security [14] (which preserves user privacy and prevents secret and sensitive information leakage from adversaries).

Main contributions. The proposal $\mathcal{SPCA}$ enjoys the following advantages.

- Explicit key confirmation: in $\mathcal{SPCA}$, each entity is assured that the other entity has actually computed the session key.
- Initiator untrecability: in terms of user privacy protection, $\mathcal{SPCA}$ provides initiator untrceability that is stronger than initiator anonymity and requires that any adversary should be not only infeasible to infer the identity of the initiator but also prevented from linking one (unknown) user interacting with the server to another transcript.
- DoS attack resilience: $\mathcal{SPCA}$ defeats DoS attack, meaning that anyone cannot impersonate a legal user to send information to the server who shall then reject the legal user to log in subsequently.
- Resistance against offline attack: $\mathcal{SPCA}$ is secure against offline password-guessing attack even when the keys stored on the card are compromised.
- Robust password changing: $\mathcal{SPCA}$ realizes robust password-changing operation and resists mistyping attack in the password-changing phase.

These appealing features make $\mathcal{SPCA}$ a suitable candidate instantiation for authentication services of human-centric security, by embedding in the computer and software systems.

Roadmap. The rest of the paper is organized as follows. Section 2 addresses the requirements of robust password changing in a two-factor authentication. The proposal $\mathcal{SPCA}$ is described in Section 3. We analyze in Section 4 the security of $\mathcal{SPCA}$, followed by the discussion of other functionalities in Section 5. Section 6 concludes the whole paper.

2. TOWARDS ROBUST PASSWORD CHANGING: THE REQUIREMENTS

In human-centric authentication mechanism, robust password changing should be a big issue, rather than a marginal property. Previous authentications, say [9–11], took little consideration on robust password changing. To ensure the usability and the security of practical applications, these requirements of password changing are reasonable and necessary:

(i) Only the user who knows the current password for his or her smart card has the capability of changing the password.
(ii) Only the user who knows the current password for his or her smart card can update the password successfully.
(iii) Once the password is updated successfully, the old password (associated with the old secret information on smart card) should be invalid for subsequent login.
(iv) The password-changing phase is independent from the login phase, as they are different phases with different missions.

These issues are obligated to provide a robust password-changing operation. The password-changing phases of most existing password authentications violate some of these requirements and thus may be vulnerable to some attacks or performance degradation. For example, the schemes in [9,10] violate requirement (iv), which results in an inefficient password-changing phase, whereas the scheme in [11] violates all the aforementioned requirements but the last one.

Note that the schemes that do not satisfy the first two requirements may be vulnerable to DoS attack. The DoS attack prevents or inhibits the normal use or management of communication facilities. This attack may act on a specific user. For instance, an adversary may perform this attack to cause the server to reject the login of a specific user. Take for example the scheme in [11], where the password-changing phase consists of a single operation of replacing on the card the value V with $V^* = V \oplus h(PW) \oplus h(PW^*)$, if the card owner wants to change his or her password from the old one PW to the new one PW^* and the keys in them. Herein, V is a value stored on the card, and h is a hash function. One subtle flaw of this noninteractive password changing results from the fact that the password cannot be expected to be read reliably into the computer or other card readers [12].

For example, password mistyping is very common in reality, which may be random or be skewed by the adversary via technical means or social engineering manipulation [12]. When the user keys in the password PW^*, mistyping

(no matter random or being skewed by the adversary) may happen, and the card reader treats some value, say PW^{**}, as the expected PW^{*}. Then, $V^{**} = V \oplus h(PW) \oplus h(PW^{**})$ would cause an authentication failure in the next session. In practice, this could happen in a malicious model. Suppose that an adversarial colleague of the user wants to make the user's card ineffective, he or she can launch the attack at the *lunch time*. Namely, he or she may run the password-changing program by inputting two random strings PW_1 and PW_2, and then the card becomes useless because the user knows neither PW_1 nor PW_2.

3. TOWARDS ROBUST PASSWORD CHANGING AND COMPROMISE SECURITY SIMULTANEOUSLY: THE PROPOSAL *SPCA*

This section describes the proposal $\mathcal{SPCA}$ that satisfies the requirements in Section 2, besides the security properties of authentication schemes [15]. Thus, the scheme can be viewed as a suitable candidate instantiation for authentication services of human-centric security, by embedding in the computer and software systems. The proposal $\mathcal{SPCA}$ consists of five stages: parameter generation, registration, precomputation, login, and password-changing phases.

3.1. Parameter generation

(1) Server $\mathcal{S}$ chooses a large prime p, generates an elliptic curve E: $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{Z}_p$ s.t. $4a^3 + 27b^2 \bmod p \neq 0$, and finds a point G of large order n on the elliptic curve E.
(2) $\mathcal{S}$ selects a random number $x \in \mathbb{Z}_n$ and computes the public key $P = x \cdot G$.
(3) $\mathcal{S}$ holds x as its secret master key and makes the parameters (p,E,G,P) public.
(4) The server also publishes hash functions h, H, $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, which will be used in other phases.

3.2. Registration

As showed in Table I, the registration phase is performed between the user $\mathcal{U}$ and the server $\mathcal{S}$.

(1) User $\mathcal{U}$: randomly choose b and α, compute $k_u = h(PW||b)$, and send ID_u, α, and k_u to $\mathcal{S}$, where PW and ID_u are the password and the identity of $\mathcal{U}$, respectively.
(2) Server $\mathcal{S}$: given k_u and ID_u from $\mathcal{U}$, compute $k_s = h(P||x)$ and $t = H(k_s, ID_u, k_u, c_u)$, where c_u is a random number. Set $\alpha_u = \alpha$, append (ID_u, α_u, c_u) to the registration table T, and send (t,P) to $\mathcal{U}$.

Eventually, the smart card contains (t,P,b,α), and the server holds the verification table $T = \{(ID_u, \alpha_u, c_u)\}$ for its users.

Table I. Registration phase of *SPCA*.

$\mathcal{U}$		$\mathcal{S}$
choose random b, α		
$k_U = h(PW\|\|b)$		
	$\xrightarrow{ID_u, \alpha, k_u}$	
		$k_s = h(P\|\|x)$
		choose random c_u
		$t = H(k_s, ID_u, k_u, c_u)$
		$\alpha_u = \alpha$
		append (ID_u, α_u, c_u) to T
	$\xleftarrow{t,P}$	
card contains (t,P,b,α)		

In $\mathcal{SPCA}$, the server $\mathcal{S}$ need not store a password table for its users. Note that if a password table is used on the server side, the adversary may find some clever way to crack the passwords. Indeed, there are various ways to obtain the password table, for example, dumping memory at a convenient time and searching the system backup tapes.

3.3. Precomputation

Before $\mathcal{U}$ and $\mathcal{S}$ take part in the login protocol, they execute the following computations to enhance the performance.

- User $\mathcal{U}$: randomly choose r_u and compute $M_1 = r_u \cdot G$ and $M_2 = r_u \cdot P$.
- Server $\mathcal{S}$: randomly choose r_s and compute $M_3 = r_s \cdot G$.

These precomputed points are only valid for one login session.

3.4. Login

After the precomputation, $\mathcal{U}$ holds $M_1 = r_u \cdot G$, $M_2 = r_u \cdot P$, and $\mathcal{S}$ holds $M_3 = r_s \cdot G$, $k_s = h(P||x)$. These quantities may be used in the following login protocol initiated by the user. Table II shows the interactions between $\mathcal{U}$ and $\mathcal{S}$.

(1) User $\mathcal{U}$: let $A = M_1$, update $\alpha = \alpha + 1$, and then compute

$$B = (ID_u||\alpha||k_u||H(A, ID_u, \alpha, k_u, t)) \oplus \mathcal{H}_1(M_2) \quad (1)$$

where $k_u = h(PW||b)$, on input PW, b, t, α and the precomputed M_1, M_2. Send (A,B) to $\mathcal{S}$.

(2) Server $\mathcal{S}$: on receiving (A,B), parse $B \oplus \mathcal{H}_1(x \cdot A)$ as $ID_u||\alpha||k_u||\sigma$. If there exists in T a record (ID_u, α_u, c_u) starting with ID_u s.t. $\alpha > \alpha_u$ and

$$\sigma = H(A, ID_u, \alpha, k_u, H(k_s, ID_u, k_u, c_u)) \quad (2)$$

execute the following (otherwise abort):

(a) Update $\alpha_u = \alpha$ in T and compute $M_4 = r_s \cdot A$, $K = \mathcal{H}_2(x \cdot A, A, M_3, M_4)$.
(b) Let $C = M_3$, $X = H(K,C)$ and send (C,X) to $\mathcal{U}$.

Table II. Login phase of *SPCA*.

$\mathcal{U}$		$\mathcal{S}$
$A = M_1$ $\alpha = \alpha + 1$ $B = (ID_u \|\| \alpha \|\| k_u \|\| H(A, ID_u, \alpha, k_u, t)) \mathcal{H}_1(M_2)$	$\xrightarrow{A,B}$	$ID_u \|\| \alpha \|\| k_u \|\| \sigma = B \oplus \mathcal{H}_1(x \cdot A)$ If $\exists (ID_u, \alpha_u, c_u) \in T$ s.t. $\alpha > \alpha_u$ $\sigma = H(A, ID_u, \alpha, k_u, H(k_s, ID_u, k_u, c_u))$ then $\alpha_u = \alpha$ $M_4 = r_s \cdot A$ $K = \mathcal{H}_2(x \cdot A, A, M_3, M_4)$ $C = M_3,\ X = H(K,\ C)$
$K^\star = \mathcal{H}_2(M_2, M_1, C, r_u \cdot C)$ $X =^? H(K^\star, C)$ $Y = H(K^\star, X)$	$\xleftarrow{C,X}$	
	$\xrightarrow{Y}$	$Y =^? H(K, X)$

(3) User $\mathcal{U}$: on receiving (C,X), compute $K^\star = \mathcal{H}_2(M_2, M_1, C, r_u \cdot C)$ and then check $X =^? H(K^\star, C)$: if not, abort; otherwise, set $K^\star$ as the session key and send $Y = H(K^\star, X)$ to $\mathcal{S}$.

(4) Server $\mathcal{S}$: if $Y = H(K,X)$, set K as the session key; otherwise, abort.

The login protocol uses α_u the counter maintained by the server. This ensures the *freshness* of each login session and protects $\mathcal{SPCA}$ from the *replay* attack. Given a valid login transcript, the adversary cannot mount offline dictionary attack as k_u, the only quantity contains knowledge about PW, and its related hash value are encrypted by a variant of El Gamal encryption with provable security [14]. Meanwhile, the authenticated Diffie–Hellman [13] embedded in the protocol is responsible for the session key's security to defeat the men-in-the-middle attack.

3.5. Password-changing operation

This phase is invoked whenever $\mathcal{U}$ wants to update his or her password from PW (and the nonce b) to PW' (and the nonce b'). Table III shows the interactions between $\mathcal{U}$ and $\mathcal{S}$.

(1) User $\mathcal{U}$: let $A = M_1$, update $\alpha = \alpha + 1$, and then compute

$$B = (ID_u \|\| \alpha \|\| k_u \|\| k_u \|\| H(A, ID_u, \alpha, k_u, k_u, t)) \oplus \mathcal{H}_3(M_2) \quad (3)$$

where $k_u = h(PW \| b)$ and $k_u = h(PW' \| b')$, on input PW, b, PW', b', t, α and the precomputed M_1, M_2. Send (A,B) to $\mathcal{S}$.

(2) Server $\mathcal{S}$: on receiving (A,B), parse $B \oplus \mathcal{H}_3(x \cdot A)$ as $ID_u \| \alpha \| k_u \| k_u \| \sigma$. If there exists in T a record (ID_u, α_u, c_u) containing ID_u s.t. $\alpha > \alpha_u$ and

$$\sigma = H\left(A, ID_u, \alpha, k_u, k'_u, H(k_s, ID_u, k_u, c_u)\right) \quad (4)$$

where $k_s = h(P \| x)$, execute the following (otherwise abort):

(a) Update $\alpha_u = \alpha$ in T and compute $K_c = \mathcal{H}_2(k_u, k_u, x \cdot A, A)$.

(b) Randomly choose c'_u and compute $t' = H(k_s, ID_u, k'_u, c'_u)$, where $k_s = h(P \| x)$.

(c) Compute $C = E_{K_c}(t')$, $X = H(K_c, C)$ and send (C,X) to U.

(3) User $\mathcal{U}$: on receiving (C,X), compute

$$K_c^\star = \mathcal{H}_2\left(h(PW \| b), h\left(PW' \| b'\right), M_2, M_1\right)$$

check $X =^? H\left(K_c^\star, C\right)$: if not, abort; otherwise, obtain $t' = H(k_s, ID_u, k'_u c'_u)$ by decrypting C with $K_c^\star$, update $t = t'$ and $b = b'$, and send $Y = H\left(K_c^\star, X\right)$ to $\mathcal{S}$.

(4) Server $\mathcal{S}$: given Y, check $Y =^? H(K_c, X)$; if not, replace (ID_u, α_u, c_u) by $(ID_u, \alpha_u, c_u, c_u)$; else, update $c_u = c'_u$.

In the login or the password-changing phases, once a user submits to the server the tuple (A,B) containing ID_u identical to that of $(ID_u, \alpha_u, c_u, c'_u)$, $\mathcal{S}$ deletes the one of c_u, and c'_u that cannot pass the verification equation (2) or (4) in Steps 2 maintains a valid triple in the registration table T and continues the remaining steps as usual. Therefore, the user can use only one of $(PW,\ b,\ t)$ and $(PW',\ b',\ t')$ to login or change the password later.

The password-changing mechanism is independent from the proposed login protocol. In some existing authentications (e.g., [9,10]), it is required that the user and the server first run the login protocol to obtain session key, which is then used to encrypt secret information for

Table III. Password-changing phase of *SPCA*.

$\mathcal{U}$		$\mathcal{S}$
$A = M_1$ $\alpha = \alpha + 1$ $B = (ID_u \Vert \alpha \Vert k_u \Vert k_u \Vert H(A, ID_u, \alpha, k_u, k_u, t))$ $\oplus \mathcal{H}_3(M_2)$	$\xrightarrow{A,B}$	$ID_u \Vert \alpha \Vert k_u \Vert k_u \Vert \sigma = B \oplus \mathcal{H}_3(x \cdot A)$ If $\exists\, (ID_u, \alpha_u, c_u) \in T$ s.t. $\alpha > \alpha_u$ $\sigma = H(A, ID_u, \alpha, k_u, k'_u, H(k_s, ID_u, k_u, c_u))$ Then $\alpha_u = \alpha$ $K_c = \mathcal{H}_2(k_u, k'_u, x \cdot A, A)$ choose random c_u $t' = H(k_s, ID_u, k_u, c_u)$ $C = E_{K_c}(t'), X = H(K_c, C)$
$K_c^\star = \mathcal{H}_2(h(PW \Vert b), h(PW' \Vert b'), M_2, M_1)$ $X =^? H(K^\star, C)$ $t = D_{K_c^\star}(C)$ $b = b'$ $Y = H(K_c^\star, X)$	$\xleftarrow{C,X}$	
	$\rightarrow^{Y}$	If $Y = H(K_c, X)$ then $c_u = c'_u$ else replace the record (ID_u, α_u, c_u) with $(ID_u, \alpha_u, c_u, c'_u)$

updating the password. Therefore, the computation and communication costs are much higher than those of $\mathcal{SPCA}$ (which does not need such a session key and thus reduces the overload of computation and communication). In some other authentications (e.g., [11]), although these two phases are independent, the password-changing operation is vulnerable to mistyping attack that leads to authentication failure in the subsequent session (refer to Section 2 for more details). In the proposed password-changing phase, the adversary without the knowledge of the correct password cannot succeed in running the password-changing phase, as will be analyzed in the coming section.

4. SECURITY ANALYSIS OF THE PROPOSAL

This section gives the security analysis of $\mathcal{SPCA}$ in terms of authentication and checks whether $\mathcal{SPCA}$ satisfies the requirements of robust password changing described in Section 2. In the succeeding text, we first give security model under which the proposal $\mathcal{SPCA}$ is analyzed.

4.1. Security model

According to different resources that the adversary has, we consider the adversary's ability. For the user in the system, an adversary might utilize the following resources to launch its attack:

- Card compromise: the adversary can compromise the user's card including its secret data.
- User compromise: the adversary can compromise the user's passwords.
- General case: the adversary has neither the user's card nor the password.

The adversary might also compromise the server and thus obtain the master key of the server. However, most of existing authentication schemes assume that the server's master key is well protected and therefore ignore this case, whereas, if the server is compromised, the adversary can do anything without the user's permission. It is desirable that given the transcript of the session, the adversary cannot infer any session key exchanged before the server is compromised. We call it as forward secrecy under the server-compromise attack. The schemes in [9,10] pay no attention to and are thus vulnerable to this attack.

Generally, the adversaries are classified into two types: static and adaptive. An adversary is *static* if it can just eavesdrop over the channel and *adaptive* if it can modify, delete, and insert the messages on the public channel. Hereafter, we focus on adaptive adversary for security analysis as it is more powerful than static one.

4.2. Authentication security of *SPCA*

The security results are stated in the following claims. Herein, we say that a scheme is secure if there exists no

adversary can impersonate either the server or the user or deduce useful information about the session keys in the login phase.

Claim 4.1. SPCA is secure against adaptive adversary under card-compromise attack.

Proof. If the smart card is compromised, the last protection for the user is the password *PW*. Without *PW*, the adversary cannot generate a valid (A,B) that can pass the verification of Step 2 in the login phase with probability greater than $\frac{1}{|\mathbb{D}|}$ for the password dictionary $\mathbb{D}$. Moreover, a used valid (A,B) generated by the user cannot be used later as the server already updated α_u, making the used (A,B) invalid in the verification to be executed.

From a valid transcript of the login phase, the adversary cannot mount offline dictionary attack because k_u, the only information that contains knowledge about *PW*, and its related hash value are encrypted by a variant of El Gamal encryption with provable security [14]. Therefore, the login phase does not leak any information for *PW*. Meanwhile, the session key's security is also achieved by embedding a fresh Diffie–Hellman problem [13] in the protocol.

The adversary cannot impersonate the server either if the smart card is compromised. This is due to the design trick that in Step 2, the server should compute $x \cdot A$ to generate the session key K. Without the knowledge of the master key x, the adversary cannot do this because of the difficulty of computing Diffie–Hellman problem [16].

Therefore, $\mathcal{SPCA}$ is secure against adaptive adversary under card-compromise attack.

Claim 4.2. $\mathcal{SPCA}$ is secure against adaptive adversary under user-compromise attack.

Proof. Without the smart card that contains (t,b,P,α), the adversary knowing the password *PW* cannot generate a valid (A,B) to initialize the session talk because the server uses t to verify (A,B) that should be generated with t. Moreover, previously used (A,B) cannot pass the verification because α_u was updated. Therefore, no adversary can impersonate the server and deduce any useful information on the session key, given the corresponding transcript of interaction because of the hardness of the Diffie–Hellman problem [16] and security protection of El Gamal encryption [14]. □

Claim 4.3. $\mathcal{SPCA}$ is secure against adaptive adversary in the general case.

This claim sounds naturally because an adversary breaking $\mathcal{SPCA}$ in the general case can also break $\mathcal{SPCA}$ under card-compromise attack and user-compromise attack. □

Claim 4.4. $\mathcal{SPCA}$ achieves forward security against adaptive adversary under user-compromise and server-compromise attacks.

Proof. The session key K of $\mathcal{SPCA}$ roots in a Diffie–Hellman tuple (G,P,M_1,M_2) and an authenticated fresh Diffie–Hellman tuple (G,M_2,M_3,M_4) [16]. Therefore, once K is generated, no one can obtain its useful information, thanks to the security of the fresh Diffie–Hellman problem. Actually, even if the server's master key, the smart card, and the password are all compromised, the adversary cannot obtain any information about the session keys previously generated. □

Claim 4.5. $\mathcal{SPCA}$ can defend DoS attack in the login phase and password-changing phase.

Proof. In the two phases of $\mathcal{SPCA}$, the messages sent in each move are authenticated with hash value (which can be viewed as a hash message authentication code). In the login phase, (A,B) contains the authenticator of A, ID_u, k_u, and α; (C,X) is an authenticated encryption ciphertext; and Y is to confirm the received messages. Therefore, it is difficult for an adversary to initialize a DoS attack without being detected. The password-changing phase is designed via the same idea that leads to the immunity of DoS attack as well. □

4.3. Robust password changing

Obviously, the proposed password-changing mechanism is independent from the proposed login protocol. Besides, it also satisfies other requirements in Section 2.

Claim 4.6. The password-changing protocol of $\mathcal{SPCA}$ achieves all the requirements of robust password changing.

Proof. In $\mathcal{SPCA}$, the password-changing phase is independent from the login phase; therefore, the security of password changing does not rely on that of session key agreement. (A,B) sent in the first move of the password-changing protocol is a ciphertext of a variant of El Gamal encryption that encrypts ID_u, α, k_u, k'_u and the authenticator that authenticates these data and the token t issued by the server. Without knowing *PW* and t, anyone cannot generate (A,B) that could pass the subsequent verifications of the server. Resending or replaying (A,B) is useless as we use a one-time counter α_u (unknown to outsiders) embedded in (A,B). Moreover, without (PW', b') and (PW, b), anyone cannot update the token t and b. Therefore, the password-changing protocol achieves security requirements (i) and (ii) in Section 2.

On the other hand, once the user ends the password-changing protocol successfully, the data on the old card and the old password *PW* become invalid as t, issued by and only by the server, has changed. This token also contains the new password and a random number c_u chosen by the server. Thus, the password-changing protocol achieves security requirement (iii) in Section 2.

As the password-changing mechanism itself provides authentication service, authentication-related properties of

the password-changing mechanism are similar to those of the login protocol. □

5. OTHER FUNCTIONALITIES

As noted in [10,17–19], protection of personal information is very important in public communication networks.

Claim 5.1. $\mathcal{SPCA}$ provides strong privacy protection for the user, achieving initiator anonymity and initiator untraceability.

Proof. In the login phase, the user sends (A,B) to the server to initialize the login session, where $A = r_u \cdot G, B = (ID_u||\alpha||k_u||H(A, ID_u, \alpha, k_u, t)) \oplus \mathcal{H}_1(M_2)$ is a ciphertext of a variant of El Gamal encryption [14] that encrypts ID_u, k_u, the authenticator $H(A, ID_u, \alpha, k_u, t)$, and other related information. Note that (C,X) and Y transmitted during the session are independent from ID_u. Under the assumption that $\mathcal{H}_1(\cdot)$ is a random oracle, B and any random string of the same length are indistinguishable if M_2 is unknown to the adversary. Thus, the variant of El Gamal encryption leads to initiator untraceability, and no one can distinguish between two users in which one is the real initiator. □

Another important feature of $\mathcal{SPCA}$ is self-healing under card-compromise attack. Recall that smart cards are vulnerable to theft/card-compromise attack. $\mathcal{SPCA}$ provides self-healing service for the user even if the smart card is compromised.

Claim 5.2. $\mathcal{SPCA}$ provides self-healing functionality.

Proof. When a smart card is compromised, a backup of the card can be used to run the password-changing protocol. After the user successfully updates the password, the data on the backup are fresh as well. Then, any adversary that uses the original data cannot login to the server successfully. Note that if the users periodically change the passwords, they need not care about the card-compromise attack, thanks to the self-healing functionality of $\mathcal{SPCA}$.

It is clear that $\mathcal{SPCA}$ also provides some other appealing features such as no password table, no time-synchronization problem, adaptively chosen password, conveniently revoking lost card [11], and mutual authentication.

6. CONCLUSION

The paper focuses on human-centric authentication services: password changing, DoS resilience, and self-healing. With smart card, we present a password authentication scheme that strengthens the requirements of robust password changing in existing authentication schemes. The proposal not only offers robust password changing, card-compromise security, and immunity to DoS attack but also achieves additional functionalities such as strong anonymity protection and self-healing ability under card-compromise attack.

ACKNOWLEDGEMENTS

This work has been supported by the National Natural Science Foundation of China (Grant Nos. 60703031, 61172085, 61103221, 61021004, 61070249, U1135004, 61170080 and 11061130539).

REFERENCES

1. Upmanyu M, Namboodiri A, Srinathan K, Jawahar C. Blind authentication: a secure crypto-biometric verification protocol. *IEEE Transactions on Information Forensics and Security* 2010; **5**(2):255–268.
2. Yu P, Baras J, Sadler B. Physical-layer authentication. *IEEE Transactions on Information Forensics and Security* 2008; **3**(1):38–51.
3. Fan C, Lin Y. Provably secure remote truly three-factor authentication scheme with privacy protection on biometrics. *IEEE Transactions on Information Forensics and Security* 2009; **4**(4):933–945.
4. Schryen G, Rich E. Security in large-scale internet elections: a retrospective analysis of elections in Estonia, The Netherlands, and Switzerland. *IEEE Transactions on Information Forensics and Security* 2009; **4**(4):729–744.
5. Sun H, Chen Y, Lin Y. oPass: a user authentication protocol resistant to password stealing and password reuse attacks. *IEEE Transactions on Information Forensics and Security* 2011. doi:10.1109/TIFS.2011.2169958.
6. Bellovin S, Merritt M. Encrypted key exchange: password-based protocols secure against dictionary attacks. In: *Proceedings of IEEE Symposium on Security and Privacy* 2002 72–84.
7. IEEE P1363.2, http://grouper.ieee.org/groups/1363/passwdPK/submissions.html
8. Lamport L. Password authentication with insecure communication. *ACM Communications* 1981; **24**(11):770–772.
9. Juang W, Chen S, Liaw H. Robust and efficient password-authenticated key agreement using smart cards. *IEEE Transactions on Industrial Electronics* 2008; **15**(6):2551–2556.
10. Li X, Qiu W, Zheng D, Chen K, *et al*. Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards. *IEEE Transactions on Industrial Electronics* 2010; **57**(2):793–800.

11. Sun D, Huai J, Sun J, Li J, Zhang J, Feng Z. Improvements of Juang *et al*.'s password authenticated key agreement scheme with smart cards. *IEEE Transactions on Industrial Electronics* 2009; **56**(6):2284–2291.
12. Kolesnikov V, Rackoff C. Password mistyping in two-factor authenticated key exchange. *Proceedings of ICALP'08* 2008; Lecture Notes in Computer Science Volume **5126**:702–714.
13. Krawczyk H. SIGMA: the 'SiGn-and-Mac' approach to authenticated Diffie–Hellman and its use in the IKE protocols. *Proceedings of Crypto'03* 2003; Lecture Notes in Computer Science Volume **2729**:400–425.
14. Shoup V. Sequences of games: a tool for taming complexity in security proofs, available at http://www.shoup.net/papers/games.pdf.
15. Boyd C, Mathuria A. *Protocols for Authentication and Key Establishment*. Springer, Berlin Heidelberg 2003.
16. Joux A, Nguyen K. Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups. *Journal of Cryptology* 2003; **16**(4):239–247.
17. Hughes D, Shmatikov V. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security* 2004; **12**(1):3–36.
18. Wu C, Lee W, Tsaur W. A secure authentication scheme with anonymity for wireless communications. *IEEE Communications Letters* 2008; **12**(10):722–723.
19. Tang C, Wu D. Mobile privacy in wireless networks-revisited. *IEEE Transactions on Wireless Communications* 2008; **7**(3):1035–1042.