A Privacy-Flexible Password Authentication Scheme for Multi-Server Environment

Jue-Sam Chou [1]*, Yalin Chen [2], Chun-Hui Huang [3], [3]

[1] Department of Information Management, Nanhua University Chiayi 622 Taiwan, R.O.C

jschou@mail.nhu.edu.tw

Tel: 886+ (0)5+272-1001 ext.56226

*: corresponding author

[2]Institute of information systems and applications, National Tsing Hua University

d949702@oz.nthu.edu.tw

[3] Department of Information Management, Nanhua University Chiayi 622 Taiwan, R.O.C

g6451519@mail.nhu.edu.tw

Abstract

Since Kerberos suffers from KDC (Key Distribution Center) compromise and impersonation attack, a multi-server password authentication protocol which highlights no verification table in the server end could therefore be an alternative. Typically, there are three roles in a multi-server password authentication protocol: *clients*, *servers*, and a *register center* which plays the role like KDC in Kerberos. In this paper, we exploit the theoretical basis for implementing a multi-server password authentication system under two constraints: *no verification table* and *user privacy protection*. We found that if a system succeeds in privacy protection, it should be implemented either by using a public key cryptosystem or by a register center having a table to record the information shared with corresponding users. Based on this finding, we propose a privacy-flexible system to let a user can employ a random-looking *dynamic identity* or employ a *pseudonym* with the register center online or offline to login a server respectively according to his privacy requirement. Compared with other related work, our scheme is not only efficient but also the most conformable to the requirements that previous work suggest.

Keywords: *password authentication*, *impersonation attack*, *user privacy protection*, *Kerberos*, password guessing attack, *smart card lost attack*

1. Introduction

Password authentication has been widely used as an authentication mechanism in a client-server architecture over networks for many years. In the mechanism, a client generally registers his account and password to a server through a secure channel. At later time, the client can remotely access the server if he can prove his identity by

offering the correct password. In such system for convenience, a client may hope to access multiple servers by applying just one pair of account and password. For this purpose, Kerberos, which was originated by Massachusetts Institute of Technology (MIT) in the 1980s, was proposed and now becomes a popular solution adopted by current industry. Its version 5 appeared as RFC 1510 in 1993 was made obsolete by RFC 4120 in 2005. In 2007, MIT formed the Kerberos Consortium to foster continuous development. Many computer operating systems have used Kerberos, such as Windows 2000, FreeBSD, Apple's Mac OS X, Red Hat Enterprise Linux 4, Sun's Solaris, IBM's AIX, and HP's Open VMS, etc.

Kerberos [1] is based on symmetric Needham-Schroeder protocol and takes use of a trust third party termed as a key distribution center (KDC) to maintain a database of secret keys. In it, each entity – either a client or a server – shares a secret key known only between itself and the KDC. In fact, the secret key for a client is usually the hash result of his password, as in the system of UNIX. Knowledge of this secret key serves to prove the entity's identity. However, some drawbacks of Kerberos are often concerned by researchers. *Firstly*, Kerberos is subject to a single point of failure; it requires continuous availability of a center server. Under this requirement, when the center server is down, no one can log in. *Secondly*, since all secret keys are stored in the KDC, a compromising of this authentication infrastructure will allow an attacker to launch an impersonation attack.

In recent year, some studies [2-14] (called multi-server password authentication protocols) each proposed an alternative approach of Kerberos, which emphasize no verification (or password) table usage in the server. This causes the server to get rid of stolen-verifier attack. In each of these systems, there exists a trusted register center (RC) for distributing a secret key to each entity (a server or a client) in the registration phase. The secret key is usually a computed result of both RC's private master key and the entity's identity. All the requirements RC needs is only to store his master key. Hence, such an approach can eliminate the necessity of a verification table usage in RC. We think this no verification table design principle would be a countermeasure to cope with the second drawback of Kerberos. In addition to the no verification table concept, some other multi-server authentication studies [3-8, 9-12] further advocated *RC-offline authentication*. When RC (like the role of KDC in Kerberos) is down, a client can be allowed to login a server (authentication with RC offline). We think that this advocacy would be a countermeasure to the first drawback of Kerberos (suffering the single point of failure). The other desirable features in this study area include: *no assumption for servers to be trustworthy*, *increasing servers freely*, *changing password freely*, and preventing the attacks like *offline password guessing*, *insider server spoofing*, and *smart-card loss attack*. The *smart-card loss attack* [15] indicates

that any attacker can launch an *offline password guessing* or *impersonation attack* under the circumstance that a smart card is lost and the stored data is extracted (by an attacker by using some means as introduced in [16, 17]).

More recently, for responding to the increasing demand of user privacy protection demanded in many applications, some multi-server password authentication protocols [3, 13, 14] further cooperate an untraceability function into their schemes. Untraceability is a privacy notion which can prevent an attacker from inferring the user's identity in a transcript and from linking any particular user to a specific transcript. The concept of *dynamic identity* (DID) is considered to be able to fulfill this untraceability purpose since it cannot be recognized and linked to any particular user by a network eavesdropper.

In this study, we are the first to exploit the theoretical basis for the implementation of a multi-server password authentication system under two constraints: *no verification table* and *user privacy protection*. We found that if a system succeeds in privacy protection, it should be implemented through either introducing a public key cryptosystem (PKC) or employing a table in RC to record the information shared with corresponding users. From many observed facts, we see that our argument holds. For example, studies [3, 13] are two failed privacy protection systems since they introduce neither PKC nor any table in RC (The detailed analyses will be given in Section 2.3.). Hence, in this study we propose a privacy-flexible password authentication protocol using PKC. In our system, we let a user can employ a DID (encrypted by RC public key) or employ a *pseudonym* (a random string authorized by RC) with RC online or offline to login a server respectively according to his privacy requirement.

The remainder of this paper is organized as follows. In Section 2, we give some preliminaries, including the design principle of multi-server password authentication protocols, the theoretical exploration on the implementation of privacy protection systems, and the reviews of some related privacy protection schemes. In Section 3, we show our scheme. And its merits and security features analyses are given in Section 4. In section 5, we demonstrate the performance comparisons among our scheme and other related work. Then, we show the discussion in Section 6. Finally a conclusion is given in Section 7.

2. Preliminaries

In this section, we introduce some preliminaries, including the design principle of multi-server password authentication protocols in Section 2.1, theoretical exploration on implementing systems with user privacy protection and table-free constraint in Section 2.2, and related work reviews and analyses in Section 2.3.

2.1 Design principles of multi-server password authentication protocols

According to literatures [2-14], a multi-server password authentication system typically consists of a RC and some servers and clients. In it, RC is the only trusted node. The servers should not be assumed to be trustworthy and the client is usually equipped with a tamper-resistant smart card to attain a better protection for the stored authentication data. The systems highlight no usage of any verification table in RC and servers, and a client needs only one single registration while being able to login multiple servers. Based on these literatures, we show a conceptual model for an ideal multi-server password authentication system in Figure 1. In the registration phase of Figure 1, each client or server should register his identity to RC to obtain a secrecy shared between RC and itself. Here, we give this shared secrecy a generic form $h(ID, x)$ (called credential in this paper), where x is RC's private master key, ID presents the client or server's identity, and $h(\cdot)$ is a secure one-way hash function. In the figure, credentials $\mathscr{A}$ and $\mathscr{B}$ are belonging to client i and server j, correspondingly. Thus, RC needs not store each shared secrecy in its database because he can compute the secrecy on sight of ID when the entity logins.

As for the authentication phase, some studies [2, 13, 14] require RC to be always online. However, we think such a design principle will consume more resource and communication cost, and be easily subject to a single point of failure (like Kerberos). Hence, we consider a RC-offline system [2, 4, 5, 7-14] would be more practical. Under such a consideration, in the authentication phase of Figure 1, we let RC be an intermediary when client i logins server j at first time. Through RC's trust relationship, both the client and server can authenticate each other based on credential $\mathscr{A}$ and $\mathscr{B}$ correspondingly. Meanwhile, the server distributes another credential $\mathscr{C}$ to the client for the client's subsequent logins, where $\mathscr{C} = h(ID_i, xj)$ and xj is server j's private master key. That is, when the client accesses the same server next time, he and the server can base on credential $\mathscr{C}$ to perform mutual authentication without RC's involvement.

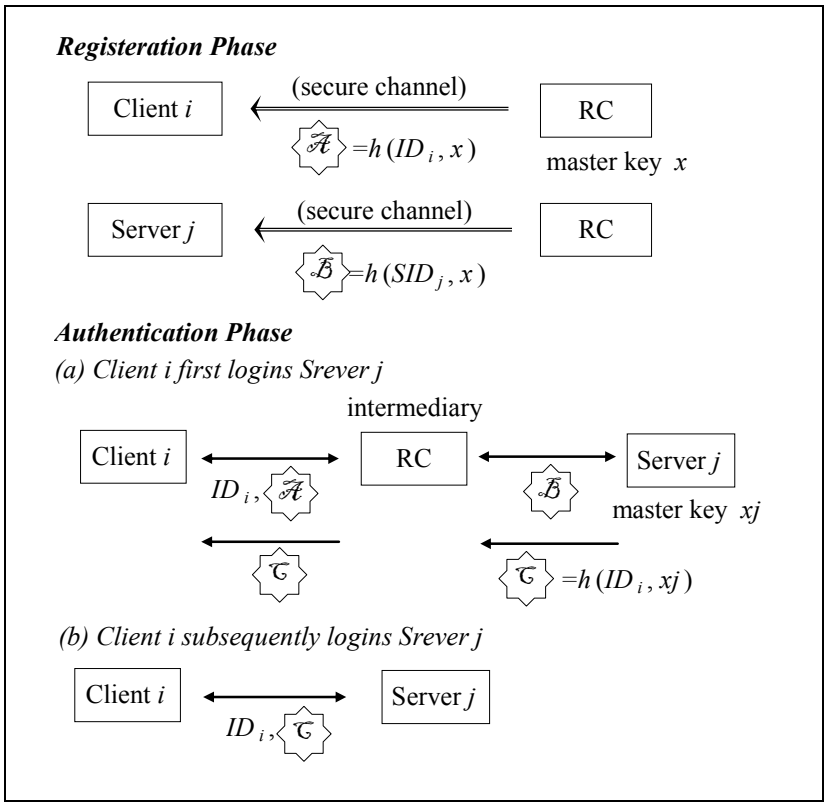


Fig. 1. Conceptual model of an ideal multi-server password authentication protocol.

2.2 Theoretical exploration on implementing systems with user privacy protection and table-free constraint

In this section, we first demonstrate what a privacy protection system should be. Then, we consider how to fulfill such a privacy protection system without RC having any authentication table.

From studies [2, 13], we know that a system can be defined as a privacy protection system if and only if it possesses two properties: (a) a user's ID (UID) should not be transferred in a clear way, and (b) a transcript can not be linked to any transcript to refer to a particular user. In other words, a user's location cannot be traced by a third party (3rdP). To design such privacy protection systems, researchers usually introduce a random-looking DID in the authentication procedure which can be conceptually treated as a UID masked with a random string rs (written as UID$\oplus rs$). Due to lack the knowledge of corresponding rs, DID cannot be distinguished by a 3rdP. It should only be identifiable by the intended recipient, i.e. RC. An intuitive implementation to accord with this requirement is to let the client and RC keep the same rs simultaneously. With this implementation, RC must maintain a table consisting of the tuple <UID, rs> for various users. Therefore, when seeing DID in an authentication request, RC can search the table to find a match, i.e., trying to match each UID$\oplus rs$

with the received DID to identify the user.

In the following, we further consider what a privacy protection system should be if RC is not allowed to have a table to store authentication-related data, such as <UID, *rs*>. Under this situation, RC will lose the capability to recognize a DID. Hence, to implement such a system, it is reasonable to consider that DID should be formed by some means where only RC can deform it while others cannot. A possible way we suggest is to introduce a symmetric or public key cryptosystem (SKC or PKC). In a SKC, the user employs a secret key to encrypt his UID with a random *rs* to form a DID and RC applies the same secret key to decrypt the DID to authenticate the user. Or in a PKC, the user employs RC's public key to encrypt his UID with a random *rs* to form a DID and RC applies his private key to decrypt the DID to authenticate the user. In both cases, a 3rdP is unable to recognize or trace a particular DID due to the security of SKC and PKC. However, if SKC is used, RC will need a table to store the symmetric keys shared with corresponding users. This conflicts with the table-free constraint. Therefore, only PKC can be used. In the following, we will use a series of statements to formally demonstrate above reasoning. (Notations: "→" indicates implication, "∧" logical and, and "∨" logical or.)

A successful privacy protection system
→RC can unmask DID with *rs* ∧ 3rdP can not unmask DID
→DID is encrypted by using PKC (*Statement I*) ∨
DID is encrypted by using SKC (*Statement II*) ∨
rs is shared between RC and user by some other means (*Statement III*)
→ *rs* is encrypted by using PKC ∨
RC has a table to record the information (symmetric keys or *rs*es) shared with corresponding users

In the above reasoning, *Statement III* encompasses all possible means (except for SKC and PKC) which can make RC be able to obtain the current *rs* for the corresponding user. Many methods can be adopted to achieve this goal. As an example, a user and RC can both share a previous *rs*, and obtain current *rs* by performing PRNG (pseudo random number generator) or hash operation on previous *rs*. All such means imply that RC needs a table to record related information (such as previous *rs*, initial value, etc.) for each user. As a result from *Statement I*, *II*, and *III*, we can conclude that a successful privacy protection system implies that the system should use PKC or RC should have a table to record the related information shared with corresponding users. Therefore, we have the following argument.

Argument 1. If a system is a successful privacy protection system, then it should be implemented either by using a public key cryptosystem or by RC having a table to

record the information shared with corresponding users.

Proof: We prove this argument by contrapositive, i.e. show that if the system is neither implemented by using PKC nor by RC having a table, then the system would fail to be a privacy protection system. As we know, if the system is implemented neither by public cryptosystem encryption (on rs) nor by recording shared information (the symmetric keys or rses shared between RC and clients), then when a client sends a DID to RC, RC would have no idea about how to deform DID, since he can not obtain the corresponding symmetric keys or rs. Thus, the system fails. This completes the proof.

2.3. Related work reviews and analyses

In this section, we use *Argument 1* to investigate three multi-server privacy protection password authentication protocols, Liao-Wang's [3], Hsiang-Shih's [13], and Wang-Juang-Li's [14]. We found that these investigations results support our argument. In the following, we will show the investigations. The corresponding schemes are shown in Figure 2 through 6. In these figures, we use SC to denote the abbreviation of smart card.

2.3.1 Liao-Wang's scheme

Liao and Wang [3] developed a DID-based multi-server authentication scheme to achieve user privacy in 2009. In the setup phase of their scheme, RC chooses a random number x as its master key and y as a system common secrecy which will be shared among all registered entities. Figure 2 and 3 show the registration phase and authentication phase, respectively.

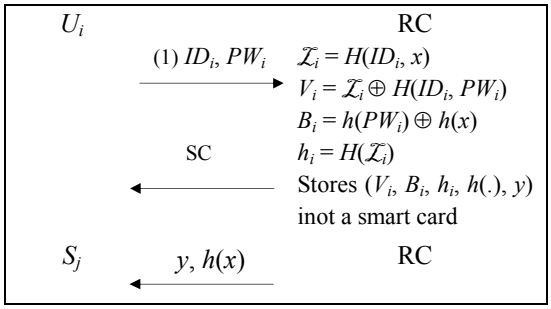
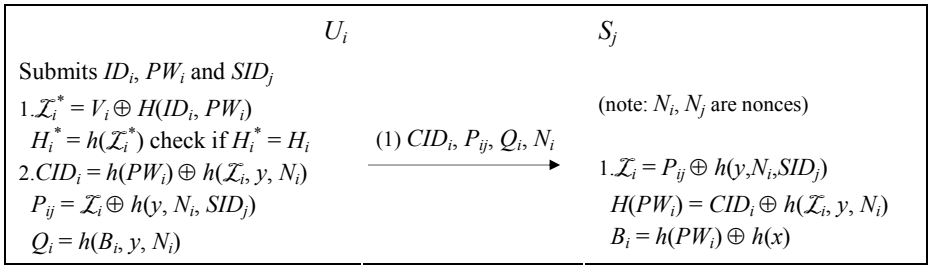


Fig. 2. Registration Phase of Liao-Wang's Scheme



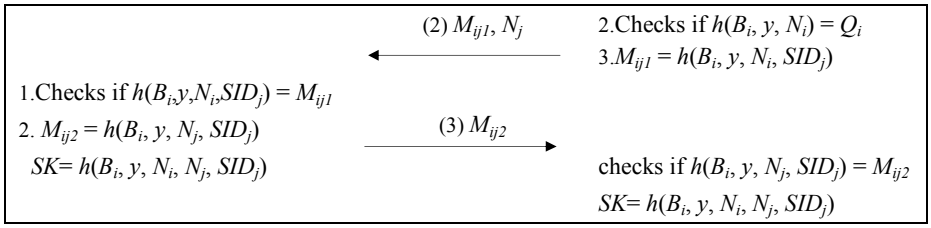


Fig. 3. Authentication Phase of Liao-Wang's Scheme

In Figure 2, we see that like in a generic multi-server authentication scheme, RC and user U_i (with identity ID_i) share credential $\mathcal{L}_i = h(ID_i, x)$, but RC and server S_j share only $h(x)$ and y. By inspecting the authentication phase of the scheme (as shown in Figure 3), we found that without the appearance of RC, the only shared information between U_i and S_j is y. In other words, the substantive basis of U_i and S_j's mutual authentication is the system common secrecy y rather than credential $\mathcal{L}_i$. This design method exposes a serious vulnerability that any entity in the system knowing y can launch an impersonation attack. Chen et al.'s study [18] and Hsiang-Shih's study [13] pointed out such an attack respectively. We show it as follows. Suppose that insider client E has eavesdropped a login request $\{CID_i, P_{ij}, Q_i, N_i\}$ that U_i sent to S_j. He can first compute $h(x) = h(PW_E) \oplus B_E$, where B_E is the secrecy stored in E's smart card, $\mathcal{L}_i = P_{ij} \oplus h(y||N_i||SID_j)$, $h(PW_i) = CID_i \oplus h(\mathcal{L}_i||y||N_i)$, and $B_i = h(PW_i) \oplus h(x)$. Once E has extracted $\mathcal{L}_i$, $h(PW_i)$, and B_i, he can impersonate U_i to login server S_j by first randomly choosing a nonce N_E, computing $CID_i^{(E)} = h(PW_i) \oplus h(\mathcal{L}_i||y||N_E)$, $P_{ij}^{(E)} = \mathcal{L}_i \oplus h(y||N_E||SID_j)$, and $Q_i^{(E)} = h(B_i||y||N_E)$, and then sending $\{CID_i^{(E)}, P_{ij}^{(E)}, Q_i^{(E)}, N_E\}$ to S_j. S_j will accept the request and regard it as being from U_i without any detection since $h(B_i, y, N_E) = Q_i^{(E)}$.

Other than the above demonstrated vulnerability, we also use *Argument 1* to examine Liao-Wang's scheme as follows. Since their scheme only uses a common secrecy y as the basis for server S_j to identify DID (represented as CID_i in Liao-Wang's scheme), it does not introduce PKC or any table in RC or servers. Any entity in the system knowing the common secrecy y would have ability to identify a user. This violates *Argument 1*. That is, it fails to achieve user privacy protection. More precisely, when S_j receives message flow (1) in the authentication phase as shown in Figure 3, any entity in the system who knows y can follow the steps as performed by S_j to identify user U_i.

Besides, we also found that Liao-Wang's scheme suffers from the smart-card lost and offline password guessing attack. Since if an attacker obtains U_i's smart card and extracts the stored data B_i ($=h(PW_i) \oplus h(x)$), he can first compute $h(x)$ in the same manner as mentioned in Chen et al.'s [18] and Hsiang-Shih's [13] attack. Then, he can guess a password PW^* and check its correctness by matching $h(PW^*) \oplus h(x)$ with B_i.

2.3.2 Hsiang-Shih's scheme

Hsiang and Shih [13] proposed an improvement on Liao-Wang's scheme. We briefly show their improvement's registration phase and authentication phase in Figure 4 and 5, respectively.

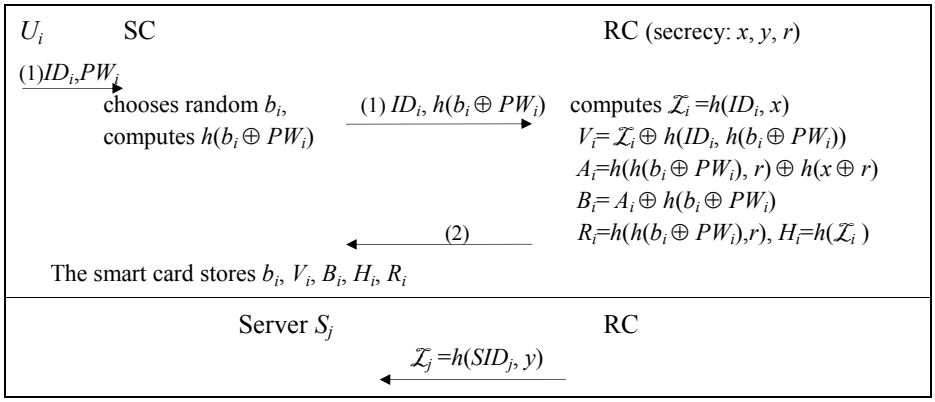
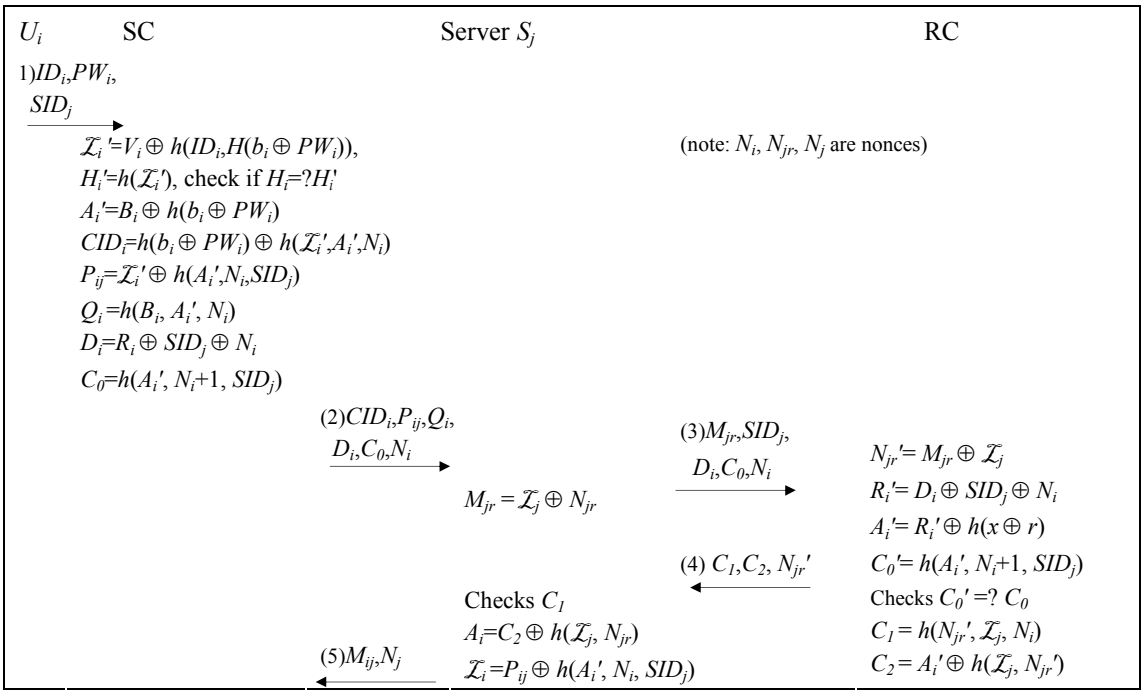


Fig. 4 Registration Phase of Hsiang-Shih's Scheme

Below, we use *Argument 1* to investigate the privacy protection in the improvement. Since it neither employs any table in RC nor introduces PKC to form DID, we therefore infer that it is not a successful privacy protection system. We show the evidence as follows. By eavesdropping on the transferred message, we can deduce all U_i's secrets stored in his smart card. For instance, from messages (2):$\{CID_i, P_{ij}, Q_i, D_i, C_0, N_i\}$, (3):$\{M_{jr}, SID_j, D_u, C_0, N_i\}$, and (4):$\{C_1, C_2, N_{jr}'\}$, transmitted among U_i, S_j and RC (as shown in Figure 5), we can deduce A_i by computing $A_i = C_2 \oplus h(M_{jr})$, and then obtain R_i, $\mathcal{Z}_i$, $h(b_i \oplus PW_i)$ by calculating $R_i = D_i \oplus SID_j \oplus N_i$, $\mathcal{Z}_i = P_{ij} \oplus h(A_i, N_i, SID_j)$, and $h(b_i \oplus PW_i) = CID_i \oplus h(\mathcal{Z}_i, A_i, N_i)$, respectively. Hence, we break the scheme.



Checks M_{ij}	$h(b_i \oplus PW_i) = CID_i \oplus h(\mathcal{L}_i, A_i, N_i)$ $B_i = A_i \oplus h(b_i \oplus PW_i)$ Check $h(B_i,A_i,N_i) =? Q_i$ $M_{ij} = h(B_i,N_i,A_i,SID_j)$

Fig. 5. Login and Authentication Phase of Hsiang-Shih's Scheme

In addition, we also demonstrate a smart-card lost and offline password guessing attack on Hsiang-Shih's scheme as follows. When an attacker obtains U_i's smart card and extracts the stored data $\{b_i, V_i, B_i, H_i (=h(\mathcal{L}_i)), R_i\}$, he can guess a password PW^* and compute $\mathcal{L}_i^* = V_i \oplus h(ID_i, h(b_i \oplus PW^*)$. He then computes and compares to see if $h(\mathcal{L}_i^*)$ is equal to H_i.

2.3.3 Wang-Juang-Lei's scheme

In 2009, Wang, Juang and Lei [14] proposed a multi-server privacy protection password authentication scheme based on quadratic residue encryption (also known as Rabin's PKC). In their system setting phase, RC selects two large primes p and q as private key and computes $n = pq$ as public key. In the user resister phase, RC computes $\mathcal{L}_i = h(ID_i, x)$ and distributes a smart card storing $\{ID_i, \mathcal{L}_i, h(.), n\}$ to U_i via a secure channel, and then U_i updates $\mathcal{L}_i$ by computing $\mathcal{L}_i' = \mathcal{L}_i \oplus h(PW_i)$. Other than this, RC also computes and distributes credential $\mathcal{L}_j = h(SID_i, x)$ to server S_j. We use Figure 6 to demonstrate the login and authentication phase of their scheme.

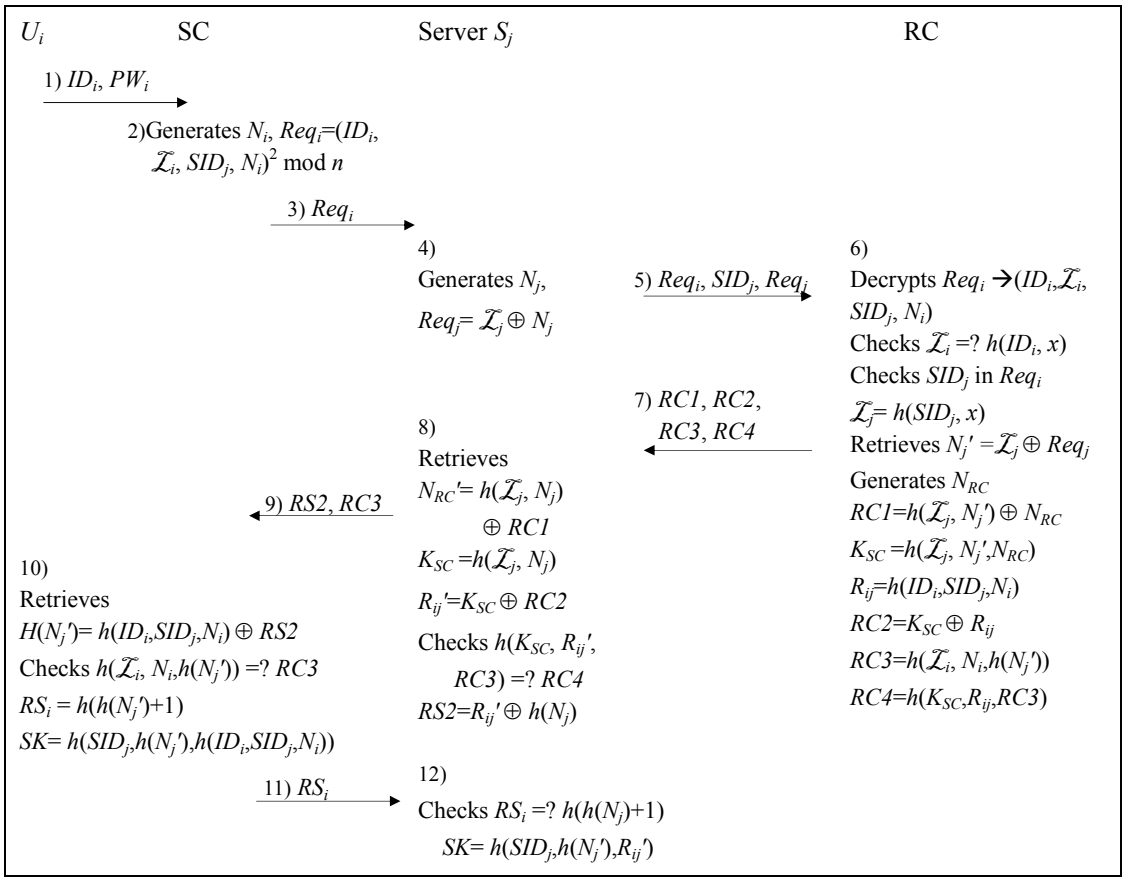


Fig 6. Login and Authentication Phase of Wang-Juang-Lei's Scheme

From the figure, we can see that user privacy can be preserved because Req_i containing user's ID is protected by Rabin's PKC; i.e., only RC knowing n's factorizing can decrypt Req_i. Moreover, Req_i is randomized by random nonce N_i and thus ensures the untraceability property. However, despite this, we found that there still exist two drawbacks. (But this does not violate *Argument 1* since our argument is an imply clause.) First, RC always needs to be online; this makes the system less efficient and vulnerable to the single point of failure like Kerberos. Second, the scheme violates the forward secrecy property. (Forward secrecy indicates that if a long-term private key is compromised, the security of any session key established earlier will not be affected [19, 20].) Since if an attacker knows the long-term secrecy (p, q), he can computes nonce N_i in Req_i (message flow (3)) and compute $R_{ij}' = h(ID_i, SID_j, N_i)$ (in step (8)). From RS2 (message flow (9)), he can obtain $h(N_j)$. Then he can compute the session key $SK = h(SID_j, h(N_j), h(ID_i, SID_j, N_i))$ (as in step (10)). That is, the session key in any earlier conversation can be extracted.

3. Proposed Scheme

In this section, we will propose a user privacy protection system. From *Argument 1*, we know that if we want our system to possess both the goals of user privacy protection and no verification table in RC and servers, the only possible approach is to introduce a PKC. Moreover, for efficiency consideration, we only let RC have a pair of public and private keys; whereas the others (including servers) do not. Since by applying the encryption of RC's public key to form a DID, a user can be identified only by RC rather than by other servers. Hence, for a user to be identified by a server without RC being online, we must introduce a *pseudonym* into the system for the user to login the server. More precisely, the user should first login and registers a *pseudonym* to the server via RC's help (online). Thereafter, the user can use the same pseudonym to login the server with RC offline any times until he wants to change it. That is, a user can decide the duration of his pseudonym which is randomly chosen by himself and authorized by RC. We characterize this design principle as flexible privacy. Our proposed scheme consists of five phases: (1) setup phase, (2) registration phase, (3) authentication with new pseudonym (RC online) phase, (4) authentication with RC offline phase, and (5) password change phase. Now, we describe it in details as follows.

(1) Setup phase

In the setup phase, RC chooses his public key and corresponding private key. It also chooses a secret master key x, a public secure one-way hash function $h(\cdot)$, and a group $<g>$ with prime order p. In addition, a server S_j also chooses xj as his secret master key. In the following, we use PKE(·) and PKD(·) to present an encryption and

decryption by using RC's public key and private key respectively.

(2) Registration phase

In the registration phase as shown in Figure 7, RC computes credential $\mathcal{Z}_i = h(ID_i, x)$ and distributes $\mathcal{Z}_i' = \mathcal{Z}_i \oplus h(PW_i, r_i)$ to client U_i, where ID_i is U_i's identity, PW_i his password, and r_i a random number chosen by U_i. Then, U_i's smart card stores $\mathcal{Z}_i'$ and r_i. On the other hand, RC distributes credential $\mathcal{Z}_j = h(SID_j, x)$ to server S_j, where SID_j is S_j's identity.

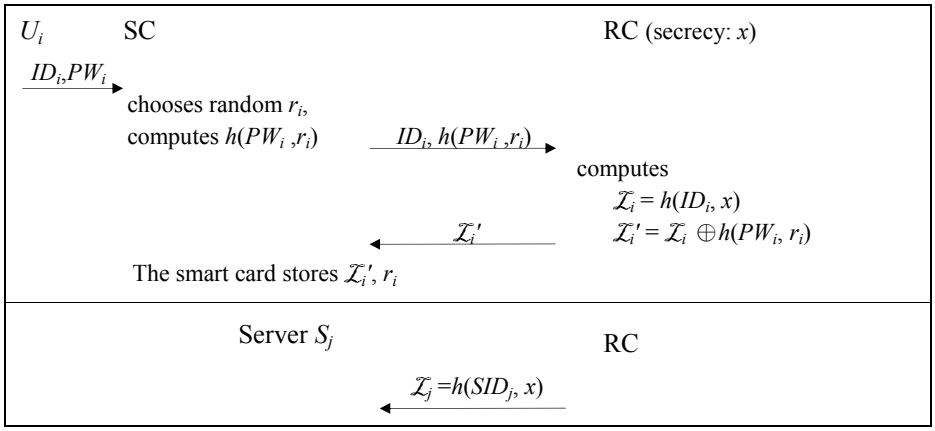
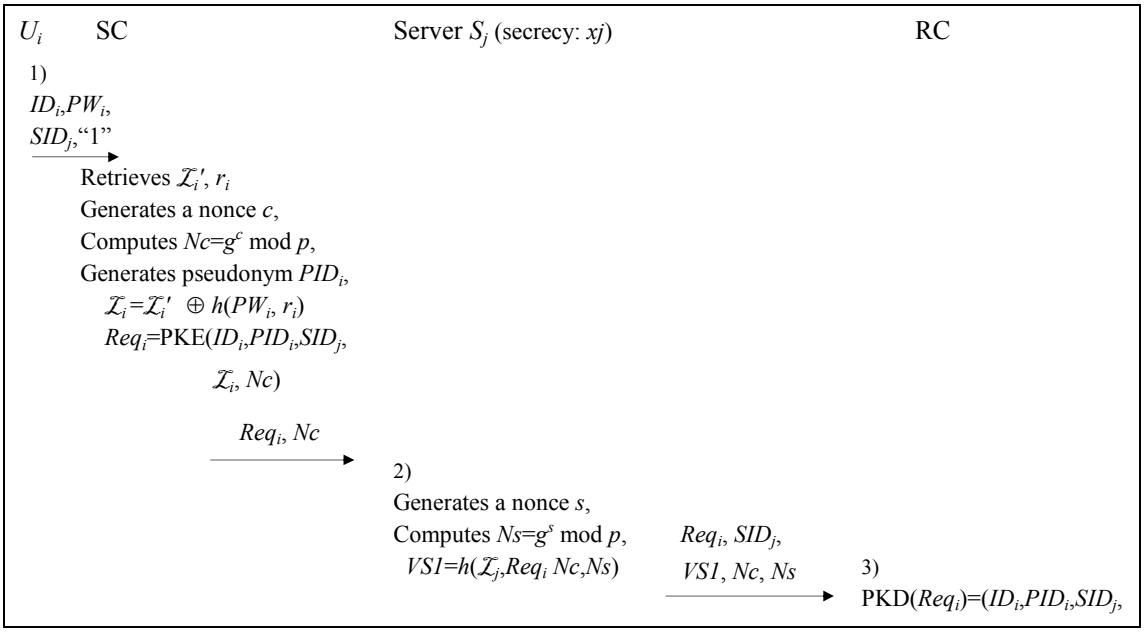


Fig. 7. The Registration Phase

(3) The authentication with new pseudonym phase

The authentication with new pseudonym phase of our scheme can be applied by U_i in two possible cases: (a) He has never logined into server S_j before and wants to generate a pseudonym for subsequent login to S_j with RC offline, and (b) He wants to renew his pseudonym relating with S_j. In this phase, U_i, S_j, and RC together perform the following steps, which are also shown in Figure 8.



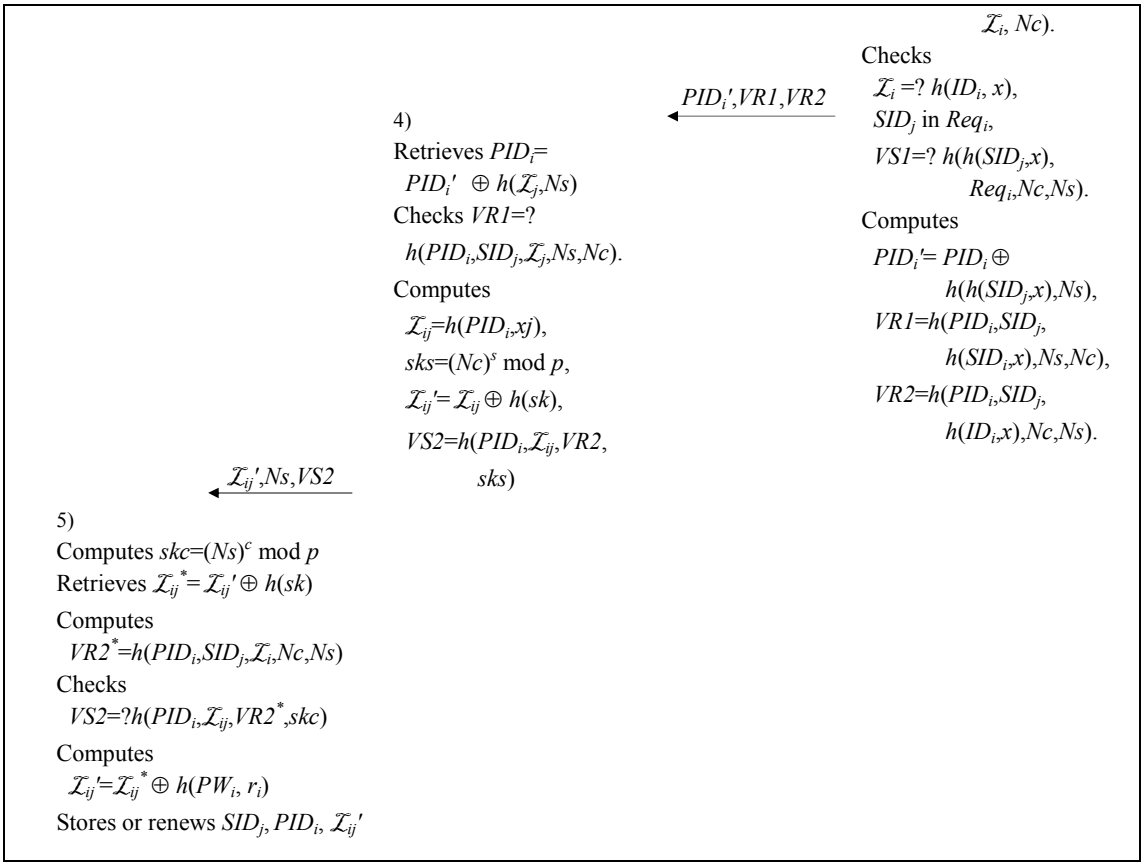


Fig. 8. Authentication with New Pseudonym Phase

Step 1. U_i keys $\{ID_i, PW_i, SID_j, \text{“1”}\}$ to his smart card, where “1” indicates U_i logins S_j with RC online. On receiving the message, smart card extracts the corresponding $\mathcal{Z}_i'$ and r_i, generates a random nonce c and pseudonym PID_i, and computes $Nc = g^c \bmod p$, $\mathcal{Z}_i = \mathcal{Z}_i' \oplus h(PW_i, r_i) = h(ID_i, x)$, and $Req_i = \text{PKE}(ID_i, PID_i, SID_j, \mathcal{Z}_i, Nc)$. Then, it sends $\{Req_i, Nc\}$ to S_j on behalf of U_i.

Step 2. On receiving $\{Req_i, Nc\}$, S_j generates a random nonce s, and computes $Ns = g^s \bmod p$ and $VS1 = h(\mathcal{Z}_j, Req_i, Ns)$. Then, it sends $\{Req_i, SID_j, VS1, Nc, Ns\}$ to RC.

Step 3. On receiving $\{Req_i, SID_j, VS1, Nc, Ns\}$, RC decrypts Req_i to obtain $\{ID_i, PID_i, SID_j, \mathcal{Z}_i, Nc\}$. It then performs three checks: (i) if $\mathcal{Z}_i = h(ID_i, x)$ holds, (ii) if the received SID_j is the same as the one in Req_i, (iii) if $VS1 = h(h(SID_i, x), Req_i, Nc, Ns)$ holds. If all three checks succeed, RC authenticates both U_i and S_j's identities, and accepts the login request. Then RC computes $PID_i' = PID_i \oplus h(h(SID_i, x), Ns)$, $VR1 = h(PID_i, SID_j, h(SID_i, x), Ns, Nc)$, and $VR2 = h(PID_i, SID_j, h(ID_i, x), Nc, Ns)$, and sends $\{PID_i', VR1, VR2\}$ to S_j.

Step 4. On receiving $\{PID_i', VR1, VR2\}$, S_j retrieves $PID_i = PID_i' \oplus h(\mathcal{Z}_j, Ns)$, and checks if $VR1 = h(PID_i, SID_j, \mathcal{Z}_j, Ns, Nc)$ holds. If it holds, S_j authenticates RC's identity and accepts pseudonym PID_i. Then, it computes a new

credential $\mathcal{Z}_{ij} = h(PID_i, xj)$ and session key $sks = (Nc)^s \bmod p$, and calculates $\mathcal{Z}_{ij}' = \mathcal{Z}_{ij} \oplus h(sks)$ and $VS2 = h(PID_i, \mathcal{Z}_{ij}, VR2, sks)$. It then sends $\{\mathcal{Z}_{ij}', Ns, VS2\}$ to U_i.

Step 5. On receiving $\{\mathcal{Z}_{ij}', Ns, VS2\}$, U_i's smart card computes $skc = (Ns)^c \bmod p$, retrieves $\mathcal{Z}_{ij}^* = \mathcal{Z}_{ij}' \oplus h(skc)$, computes $VR2^* = h(PID_i, SID_j, \mathcal{Z}_{ij}, Nc, Ns)$, and checks to see if $VS2 = h(PID_i, \mathcal{Z}_{ij}^*, VR2^*, skc)$ holds. If the check holds, U_i authenticates both RC and S_j's identities and accepts $\mathcal{Z}_{ij}^*$ to be the credential for the subsequent logins to S_j. Then, the smart card stores $\{SID_j, PID_i, \mathcal{Z}_{ij}' = \mathcal{Z}_{ij}^* \oplus h(PW_i, r_i)\}$ into its storage or updates the corresponding old parameters if SID_j already exists.

Step 6. U_i and S_j can communicate by using the session key skc ($sks = (g)^{cs} \bmod p$).

(4) Authentication with RC offline phase

We describe our authentication with RC-offline phase as follows which is also shown in Figure 9.

Step 1. U_i keys $\{ID_i, PW_i, SID_j, \text{“0”}\}$ to his smart card, where “0” indicates U_i logins S_j without RC online. On receiving the message, the smart card extracts $\{SID_j, PID_i, \mathcal{Z}_{ij}'\}$ and r_i from its storage, generates a random nonce c, and computes $Nc = g^c \bmod p$, $\mathcal{Z}_{ij} = \mathcal{Z}_{ij}' \oplus h(PW_i, r_i) = h(PID_i, xj)$, and $VU1 = h(PID_i, SID_j, \mathcal{Z}_{ij}, Nc)$. Then,it sends $\{PID_i, VU1, Nc\}$ to S_j.

Step 2. On receiving $\{PID_i, VU1, Nc\}$, S_j computes $\mathcal{Z}_{ij} = h(PID_i, xj)$ and checks to see if $VU1 = h(PID_i, SID_j, \mathcal{Z}_{ij}, Nc)$ holds. If it holds, S_j identifies PID_i as a valid pseudonym. It then generates a random nonce s, computes $Ns = g^s \bmod p$, session key $sks = (Ns)^c \bmod p$, and $VS3 = h(SID_j, PID_i, h(PID_i, xj), sks)$, and then sends $\{VS3, Ns\}$ to U_i.

Step 3. On receiving $\{VS3, Ns\}$, U_i computes session key $skc = (Nc)^s \bmod p$ and checks to see if $VS3 = h(SID_j, PID_i, \mathcal{Z}_{ij}, skc)$ holds. If it holds, U_i authenticates S_j as being valid.

Step 4. U_i and S_j can communicate to each other by using the session key skc (= $sks = (g)^{cs} \bmod p$).

U_i	SC	Server S_j
1) ID_i, PW_i, SID_j, “0”		
⟶	Retrives $PID_i, \mathcal{Z}_{ij}', r_i$ Generates a nonce c, Computes $Nc=g^c \bmod p$, $\mathcal{Z}_{ij}=\mathcal{Z}_{ij}' \oplus h(PW_i, r_i)$ Computes $VU1=h(PID_i, SID_j, \mathcal{Z}_{ij}, Nc)$	

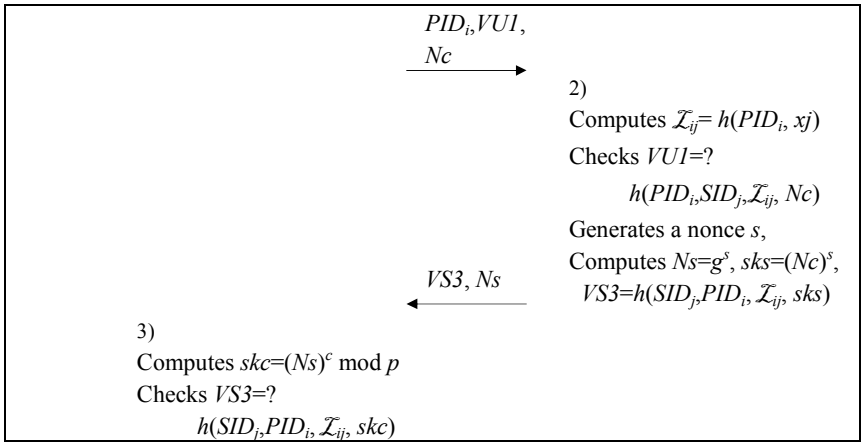


Fig. 9. Authentication with RC Offline Phase

(5) Password change phase

Finally, our scheme provides a user with the ability to change his password freely without RC or servers' involvement. We describe the process as follows which is also depicted in Figure 10. In the figure, U_i keys $\{PW_i, PW_i^{(new)}\}$ to the smart card, where PW_i is the old password and $PW_i^{(new)}$ the new one. Then, the smart card generates a random number $r_i^{(new)}$ and updates $\mathcal{Z}_i' = \mathcal{Z}_i' \oplus h(PW_i, r_i) \oplus h(PW_i^{(new)}, r_i^{(new)})$. Then, it replaces r_i with $r_i^{(new)}$ and updates each $\mathcal{Z}_{ij}'$ with $\mathcal{Z}_{ij}' \oplus h(PW_i, r_i) \oplus h(PW_i^{(new)}, r_i^{(new)})$ correspondingly.

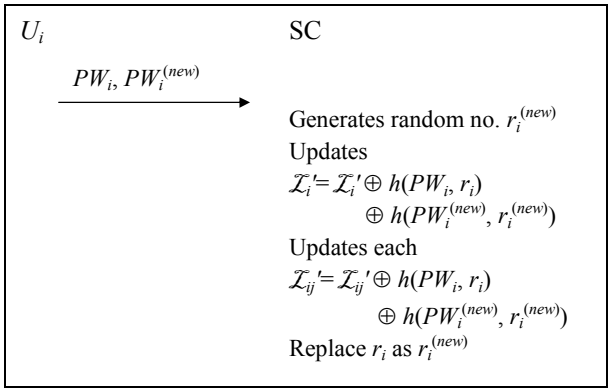


Fig. 10. Password Change Phase

4. Merits and Security Features Analyses of Our Scheme

In this section, we show the merits of our scheme in Section 4.1; then, analyze its security features in Section 4.2.

4.1. Merits

Our scheme conforms to most of the merits previous studies suggest. For instance, the possession of merits *RC-offline authentication*, *no assumption of servers to be trustworthy*, *increasing server freely*, *session key agreement*, and *changing password*

freely are obvious. In the following, we only give a detail discussion on other two merits.

(a) No verification table usage: It is obvious that not only RC but also the servers do not require a table to store user authentication data. This is because the shared secrecy, $\mathcal{L}_i = h(ID_i, x)$, between RC and client U_i is and can only be computed by RC. And, the shared secrecy, $\mathcal{L}_{ij} = h(PID_i, xj)$, between U_i and client S_j is and can only be computed by S_j. Hence, our scheme can prevent stolen-verifier attack.

(b) Flexible user privacy protection: In the authentication with new pseudonym phase, both the user's ID_i and pseudonym PID_i are first randomized by nonce N_c and then encrypted by PKE. Thus, the user privacy can be preserved on account of the security of PKE; i.e., no one except RC who possesses the private key could decrypt the encryption in polynomial time. On the other hand, in the authentication with RC offline phase, U_i can use his pseudonym $PID_i^{(j)}$ to login S_j, where $j = 1$ to n and n is the number of servers. Those $PID_i^{(j)}$s each is randomly chosen by U_i and has been authorized by RC. They are independent on the real identity *UID* and have no relationship to each other. Thus, U_i cannot be traced when he logins to different servers by using different pseudonyms. However, he could be traced when he uses the same pseudonym to login to the same server twice since this contradicts the definition of untraceability as described in Introduction (But the real identity of U_i still can not be revealed yet.). Hence, our system is a privacy-flexible one where a user can decide the lifetime of his pseudonym by himself. If he wants to use a new pseudonym, he can renew one at any time by just performing phase (3), logining to a server through RC's online authorization.

4.2 Security Features Analyses

In this section, we analyze the security features of our scheme.

(a) Mutual authentication. We first examine this feature in the login with new pseudonym phase as shown in Figure 8 (the mutual authentication among the three parties, user U_i, server S_j, and RC). For authenticating U_i, after receiving U_i's login request, S_j sends $\{Req_i, SID_j, VS1, Ns\}$ to RC. RC decrypts Req_i to obtain $\{ID_i, PID_i, SID_j, \mathcal{L}_i, Nc\}$ and verifies the validities of both $\mathcal{L}_i$ ($=h(ID_i, x)$) and $VS1$($=h(\mathcal{L}_j, Req_i, Ns)$). If they are valid, RC confirms that both U_i and S_j are authentic. Here, if we use $A \rightarrow B$ to represent A authenticates B, or equivalently, B is regarded as being authentic by A, we can demonstrate these two relations by using the two solid arrows, ① and ②, as indicated in Figure 11. RC then sends $\{PID_i'$ ($=PID_i \oplus h(\mathcal{L}_j, Ns)$), $VR1$, $VR2\}$ to S_j. S_j verifies the validity of $VR1$ ($=h(PID_i, SID_j, \mathcal{L}_j, Ns, Nc)$). If it is valid, S_j confirms that RC is authentic. This is also depicted by the solid arrow ③ in the figure. S_j then sends $\{\mathcal{L}_{ij}'$ ($=\mathcal{L}_{ij} \oplus h(sk)$),

Ns, *VS2*} to U_i. Also, U_i has to verify *VS2* (=*h*(PID_i, $\mathcal{L}_{ij}$, *VR2*, *sk*)), where *VR2* (=*h*(PID_i, SID_j, $\mathcal{L}_i$, *Nc*, *Ns*)) should be computed by RC and transmitted honestly by S_j. If *VS2* is valid, U_i confirms that RC is authentic. This is depicted by using the solid arrow ④ in the figure.

It is obvious that authenticity relationship has transitive property by way of an intermediate node when the identities of both communicating parties and their common secret are committed, *e.g.*, to a hash function. For example, if A→B and B→C, then we obtain A→C by way of B. According to this fact, from U_i→RC (arrow ④) and RC→ S_j (arrow ②), we obtain $\boldsymbol{U_i \rightarrow S_j}$ (the dashed arrow ⑤); from S_j→RC (arrow ③) and RC→U_i (arrow ①), we obtain $\boldsymbol{S_j \rightarrow U_i}$ (the dashed arrow ⑥). As a result, we can see that the mutual authentications between each pair of the three parties are satisfied in this phase.

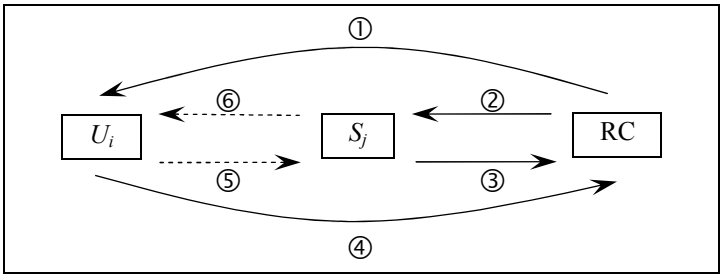


Fig. 11. Authenticity Relationship

Next, we examine the mutual authentication in the login with RC offline phase as shown in Figure 9, i.e., the mutual authentication between server S_j and an anonymous user (U_i) who uses pseudonym PID_i. Before performing RC offline login, U_i and S_j should have executed the authentication login with new pseudonym phase (through RC). Hence, they already have shared a secrecy $\mathcal{L}_{ij}$. Therefore, in this phase, S_j authenticates the anonymous user (U_i) by checking the received *VU1* to see if it contains $\mathcal{L}_{ij}$. If it does, S_j believes that the anonymous user (U_i) is valid. This is because the anonymous user (U_i) can prove that he knows secrecy $\mathcal{L}_{ij}$ which is distributed by S_j before. Similarly, U_i authenticates S_j through checking the received *VS3* to see if it contains $\mathcal{L}_{ij}$. If it does, U_i believes that S_j is the intended party because he can proves the knowledge of secrecy $\mathcal{L}_{ij}$. As a result, we can see that the mutual authentication between S_j and U_i who uses pseudonym PID_i is satisfied in this phase.

(b) Preventing offline password guessing attack. Each of the communication flows in the authentication with new pseudonym phase (phase (3)) – {Req_i, *Nc*}, {Req_i, SID_j, *VS1*, *Ns*}, {PID_i', *VR1*, *VR2*}, and {$\mathcal{L}_{ij}'$, *Ns*, *VS2*} – are independent from the login requester's password PW_i. More precisely, all the variables in the flows do not contain PW_i and thus this design can effectively prevent offline password

guessing attack. The same phenomenon also exists in the authentication with RC offline phase (phase (4)) of our scheme.

(c) Preventing smart-card-lost attack. Assume that an attacker E obtains U_i's smart card, he may extract the stored data in the card including $\mathcal{I}_i' = h(ID_i, x) \oplus h(PW_i, r_i)$, r_i, and multiple records of $<SID_j, PID_i, \mathcal{I}_{ij}' = h(PID_i, xj) \oplus h(PW_i, r_i)>$. Under this situation, firstly, we discuss the case why E cannot succeed in using the data $\mathcal{I}_i'$ and r_i to launch an offline password guessing attack in phase (3). This is because E does not have the knowledge of x and thus cannot determine $h(ID_i, x)$ and therefore $h(PW_i, r_i)$. Hence, without the value of $h(PW_i, r_i)$, the guessing value $h(PW_i^{(*)}, r_i)$, where $PW_i^{(*)}$ is the password guessed by E, will lose the basis of matching. Similarly without the knowledge of xj, E cannot succeed in using data $\mathcal{I}_{ij}'$ and r_i to launch an offline password guessing attack in phase (4) neither.

Secondly, we discuss the reason why E cannot succeed in impersonating U_i when he obtains the data stored in U_i's smart card. We consider that in phase (3), E can not employ his own secrecy ($\mathcal{I}_E'$ ($=h(ID_E, x) \oplus h(PW_E, r_E)$) and r_E) and the compromised data ($\mathcal{I}_i'$ ($=h(ID_i, x) \oplus h(PW_i, r_i)$) and r_i) to impersonate U_i. This is because without the knowledge of x and U_i's password PW_i, E is not able to extract correct $\mathcal{I}_i$ ($=\mathcal{I}_i' \oplus h(PW_i, r_i) = h(ID_i, x)$) from the two pairs ($\mathcal{I}_E'$, r_E) and ($\mathcal{I}_i'$, and r_i). (We have shown that the password can hardly be guessed in last paragraph). Similarly in phase (4), without the knowledge of xj and PW_i, E cannot extract $\mathcal{I}_{ij}$ ($=h(PID_i, xj)$), which is the basis used by S_j to identify PID_i, from his own secrecy $\mathcal{I}_{Ej}'$ ($=h(PID_E, xj) \oplus h(PW_E, r_E)$) and r_E, and from the compromised data $\mathcal{I}_{ij}'$ ($=h(PID_i, xj) \oplus h(PW_i, r_i)$) and r_i .

(d) Forward secrecy. This property indicates that the compromise of long-term keys (i.e., RC's private key, master key x, or S_j's master key xj) will not affect the security of earlier session keys. In our scheme, the session key sks ($= (Nc)^s = skc = (Ns)^c = g^{cs}$), is composed by one-time secrecy c and s which are chosen by client U_i and server S_j respectively. If an attacker compromises p, q, x, and xj, all he can obtain are Nc and Ns. However, without the secrecy c or s, he will fail in computing sks or skc.

(e) Preventing the insider-server spoofing attack. In the authentication with new pseudonym phase (phase (3)), if server S_{j+1} wants to masquerade as S_j, he will be rejected. Since without the knowledge of $\mathcal{I}_j$ ($=h(SID_j, x)$) shared only between S_j and RC, he cannot be authenticated by RC successfully. Similarly in the authentication without RC phase (phase(4)), without the knowledge of $\mathcal{I}_{ij}$ ($=h(PID_i, xj)$) shared only between S_j and U_i, S_{j+1} cannot be authenticated by U_i successfully. Hence, our scheme can prevent insider-server spoofing attack.

(f) Preventing replay attack. Sine our scheme uses fresh random nonces chosen by the two communicating parties in each session, it can prevent replay attack. In other words, an old nonce $Nc^{(old)}=g^{c(old)}$ (or $Ns^{(old)}=g^{s(old)}$) in a request (or response) can not be used to deduce a new session key $sk^{(new)}$ ($\neq (Ns^{(old)})^{c(new)} \neq (Nc^{(old)})^{s(new)}$) since $c^{(new)}$ (and $s^{(new)}$) are computationally infeasible to be extracted from the eavesdropped $Nc^{(new)}$ (and $Ns^{(new)}$) due to Discrete Logarithm problem.

5. Performance Comparisons

After the discussion of merits and security features of our scheme, in the following, we compare these features with other three related work which we have reviewed in Section 2.3.

Firstly, we list both the function and security features comparisons among these schemes in Table 1 and 2, respectively. In Table 1, Liao-Wang's and Hsiang-Shih's schemes fail in user privacy protection (we have demonstrated this in Section 2.3), and Hsiang-Shih's and Wang et al.'s schemes do not provide RC-offline authentication. As to the session key agreement, except ours is of Diffie-Hellman (DH) type, all others are non-DH key agreement schemes. Consequently, our scheme has the greatest degree of conformance to the listed functions.

Secondly, we compare the security features among these schemes and show the result in Table 2. Schemes [3, 13, 14] do not have the forward secrecy property since they are not DH type; the compromise of long-term keys will expose the content of earlier session keys. Conversely, our scheme adopting DH key agreement can achieve forward secrecy. We have analyzed this in just above Section 5. As for the feature of preventing smart-card lost attack, Liao-Wang's and Hsiang-Shih's work both suffer from the password guessing attack. We have demonstrated this in Section 2.3. From the above-mentioned, we can conclude that our scheme is the most secure among these compared schemes.

Table 1. Function comparison among our scheme and other related schemes

Features	ours	Liao-Wang's [3]	Hsiang-Shih's [13]	Wang et.al's [14]
No verification table in RC or server	○	○	○	○
User privacy protection	○	×	×	○
RC-off-line authentication	○	○	×	×
No assuming servers to be trustworthy	○	○	○	○
Increasing servers freely	○	○	○	○
Changing password freely	○	×	×	× (No offer)
DH key agreement	○	×	×	×

Table 2. Security features Comparisons

Features	ours	Liao-Wang's [3]	Hsiang-Shih's [13]	Wang et.al's [14]
Mutual authentication	○	×	×	○
Forward secrecy	○	×	×	×
Preventing insider-server spoofing attack	○	×	×	○
Preventing off-line password guessing attack	○	×	○	○
Preventing smart-card-lost attack	○	×	×	○

Table 3. Passes comparison among our scheme and other related scheme

	ours	Liao-Wang's [3]	Hsiang-Shih's [13]	Wang et.al's [14]
RC-online authentication	4	×	5	5
RC-offline authentication	2	3	×	×

Finally, we examine the number of needed passes in our scheme and related work and show the result in Table 3. In the table, Hsiang-Shih's and Wang et al's work each needs 5 passes and consumes much system overhead by only providing a RC-online protocol. Liao-Wang's scheme seems to be the most efficient by always offering RC-offline login. But without RC transferring the trust, the scheme inherently suffers from the impersonation attack. Therefore, our scheme with 4 passes in phase (3) (RC online authentication) and 2 passes in phase (4) (RC offline authentication) has the best performance in communication efficiency.

6. Discussions

In this section, we discuss some concerns.

(a). Security of secret master keys *x* and *xjs*: It can be easily seen that most multi-server password authentication schemes including ours should set the length of master key x to be sufficiently large. Otherwise, an insider attacker E can launch a password guessing attack as follows. First, E tries to read his smart card to obtain $\{\mathcal{Z}_E', r_E\}$, and then computes credential $\mathcal{Z}_E = \mathcal{Z}_E' \oplus h(PW_E, r_E) = h(ID_E, x)$. Second, he guesses x^* and checks whether $h(ID_E, x^*) = \mathcal{Z}_E$. If the master key x is truly random, then the searching space will be 2^l, where $l = |x|$. Therefore, l should be large enough to ensure the security of the master key. On the other hand, considering the security of hash function, it is generally suggested that the output length of $h(\cdot)$, i.e. $k = |h(\cdot)|$, should be at least 160 bits to resist against the birthday attack.

(b). Tradeoff between smart card access control and smart card lost password guessing attack: Smart card access control indicates that a user must provide a valid input to let the card perform his request; otherwise, the card will refuse it. For example, a smart card can be designed to store the hash result of a

cardholder's password ($h_i = h(PW_i)$). Then it will respond to a user's request only when the user keys in a right password PW_i^* satisfying $h(PW_i^*) = h_i$. Studies [3, 13] have such an access control mechanism. However, this would introduce a smart card lost and password guessing attack. Since when an attacker obtains the lost card and its storage content, he can guess the password by finding a match. On the other hand, without the basis of matching, a scheme without access control can prevent from a smart card lost and password guessing attack. In fact, many current commercial applications, such as financial IC card systems, do not adopt the strategy that the card itself verifies the user's password.

(c).Smart card storage usage: Although in our scheme, RC or servers do not keep a verification table and thus prevent the stolen-verifier attack; however, the smart card should consume additional storages to store RC-offline authentication data, such as server identity SID_j, pseudonym PID_i, and credential $\mathcal{L}_{ij}$. In contrast, Wang et al.'s scheme [14] which also bases on PKC does not require such storage consumption but in their scheme, RC needs to be involved in each user authentication. This scarifies system efficiency and easily suffers from the single point of failure. Hence, we think this is a trade off between *system's efficiency* and *card's storage usage efficiency*. Let's further estimate the storage usage in our scheme. We suppose that SID_j and PID_i each are 64-bit strings, and $\mathcal{L}_{ij}$ is a 160-bit string. Under this assumption, a tuple $<SID_j, PID_i, \mathcal{L}_{ij}>$ would totally consume 288 bits, i.e. 36 bytes. Hence, if a smart card offers extra 1KB to store the RC-offline authentication data, it can totally store 28 records for login to 28 different servers. (Accordingly, when a user wants to access the 29th server, the smart card may overwrite the least frequently used record.) This extra card's storage consumption makes our system more desirable in the consideration of system efficiency. In other words, our design is worthy in using additional card's storage in an acceptable range to gain system efficiency.

7. Conclusion

Multi-server password authentication schemes could be an alternative of Kerberos which is now widely adopted by industry. Both of the systems are of one single registration and multiple server service accesses. However, multi-server password authentication schemes emphasizing *no verification table in the server end* and *RC-offline login allowance* would be the remedies to the drawbacks of Kerberos. In addition to these two emphases, some recent multi-server password authentication schemes concern about user privacy protection as well. In this study, we are the first one to explore the theoretical basis for implementing a multi-server password authentication system with these three characteristics: *no verification table*, *RC-offline*

login allowance, and *user privacy protection*. According to *Argument 1*, we propose a privacy-flexible system possessing these three characteristics to let a user can employ a random-looking DID with RC online, or employ a pseudonym with RC offline, to login a server according to his privacy requirement. In addition to the characteristics, our scheme conforms to most of the merits that previous studies suggest, including no assumption of servers to be trustworthy, increasing server freely, Diffie-Hellman session key agreement, and changing password freely. Also, we have proved that our scheme can achieve mutual authentication and forward secrecy, and can prevent from offline password guessing attack, smart card lost attack, insider-server spoofing, and replay attack.

References

[1] Kerberos White Paper, http://www.kerberos.org/software/whitepapers.html.

[2] J. L. Tsai, "Efficient multi-server authentication scheme based on one-way hash function without verification table", *Computers & Security*, Vol. 27, No. 3-4, pp. 115-121, May-June 2008.

[3] Y. P. Liao, S. S. Wang, "A secure dynamic ID based remote user authentication scheme for multi-server environment", *Computer Standards & Interfaces*, Vol. 31, No. 1, pp. 24-29, Jan. 2009.

[4] W. J. Tsaur, C. C. Wu, W. B. Lee, "An enhanced user authentication scheme for multi-server Internet services", *Applied Mathematics and Computation*, Vol. 170, No. 1-1, pp. 258-266, Nov. 2005.

[5] W. J. Tsaur, C. C. Wu, W. B. Lee, "A smart card-based remote scheme for password authentication in multi-server Internet services", *Computer Standards & Interfaces*, Vol. 27, No. 1, pp. 39-51, Nov. 2004.

[6] I. C. Lin, M. S. Hwang, L. H. Li, "A new remote user authentication scheme for multi-server architecture", *Future Generation Computer Systems*, Vol. 19, No. 1, pp. 13-22, Jan. 2003.

[7] J. H. Lee, D. H. Lee, "Efficient and Secure Remote Authenticated Key Agreement Scheme for Multi-server Using Mobile Equipment", *Proceedings of International Conference on Consumer Electronics*, pp. 1-2, Jan. 2008.

[8] L. Hu, X. Niu, Y. Yang, "An Efficient Multi-server Password Authenticated Key Agreement Scheme Using Smart Cards", *Proceedings of International Conference on Multimedia and Ubiquitous Engineering*, pp. 903-907, Apr. 2007.

[9] C. C. Chang, J. Y. Kuo, "An efficient multi-server password authenticated key agreement scheme using smart cards with access control", *Proceedings of International Conference on Advanced Information Networking and Applications*, Vol. 2, No. 28-30, pp. 257-260, Mar. 2005.

[10] R. J. Hwang, S. H. Shiau, “Password authenticated key agreement protocol for multi-servers architecture”, *Proceedings of International Conference on Wireless Networks*, Vol. 1, No. 13-16, pp. 279-284, Jun. 2005.

[11] C. C. Chang, J. S. Lee, “An efficient and secure multi-server password authentication scheme using smart cards”, *Proceedings of International Conference on Cyberworlds*, No. 18-20, pp. 417-422, Nov. 2004.

[12] W. S. Juang, “Efficient multi-server password authenticated key agreement using smart cards”, *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 1, pp. 251-255, Feb. 2004.

[13] H. C. Hsiang, W. K. Shih, “Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment”, *Computer Standards & Interfaces*, Vol. 31, No. 6, pp. 1118-1123, Nov. 2009.

[14] R. C. Wang, W. S. Juang, C. L. Lei, “User Authentication Scheme with Privacy-Preservation for Multi-Server Environment”, *IEEE Communications Letters*, Vol. 13, No. 2, pp. 157-159, Feb. 2009.

[15] I. E. Liao, C. C. Lee, M. S. Hwang, “A password authentication scheme over insecure networks”, *Journal of Computer and System Sciences*, Vol. 72(4), Jun. 2006, 727-740.

[16] P. Kocher, J. Jaffe, B. Jun, “Differential power analysis”, *Proceedings of Advances in Cryptology- CRYPTO 99*, 1999, 388-397.

[17] T. S. Messerges, E. A. Dabbish, R. H. Sloan, “Examining smart-card security under the threat of power analysis attacks”, *IEEE Transactions on Computers*, Vol 51(5), 2002, 541-552.

[18] Y. Chen, C. H. Huang, J. S. Chou, “Comments on two multi-server authentication protocols”, http://eprint.iacr.org/2008/544, December 2008.

[19] D. Z. Sun, J. P. Huai, J. Z. Sun, J. X. Li, J. W. Zhang, Z. Y. Feng, “Improvements of Juang et al.’s Password-Authenticated Key Agreement Scheme Using Smart Cards”, *IEEE Transactions on Industrial Electronics*, Vol. 56(6), Jun. 2009, 2284-2291.

[20] X. Li, W. Qiu, D. Zheng, K. Chen, J. Li, “Anonymity Enhancement on Robust and Efficient Password-Authenticated Key Agreement Using Smart Cards”, *IEEE Transactions on Industrial Electronics*, Vol. 57(2), Feb. 2010, 793-800.