Journal of Computer and System Sciences 80 (2014) 195–206

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture

Kaiping Xue*, Peilin Hong, Changsha Ma

The Information Network Lab of EEIS Department, USTC, Hefei, 230027, China

ARTICLE INFO

Article history:
Received 12 November 2012
Received in revised form 25 June 2013
Accepted 12 July 2013
Available online 25 July 2013

Keywords:
Authentication and key agreement
Identity protection
Multi-server architecture
Traceability
Smart card

ABSTRACT

Traditional password based authentication schemes are mostly considered in single-server environments. They are unfit for the multi-server environments from two aspects. Recently, base on Sood et al.'s protocol (2011), Li et al. proposed an improved dynamic identity based authentication and key agreement protocol for multi-server architecture (2012). Li et al. claim that the proposed scheme can make up the security weaknesses of Sood et al.'s protocol. Unfortunately, our further research shows that Li et al.'s protocol contains several drawbacks and cannot resist some types of known attacks. In this paper, we further propose a lightweight dynamic pseudonym identity based authentication and key agreement protocol for multi-server architecture. In our scheme, service providing servers don't need to maintain verification tables for users. The proposed protocol provides not only the declared security features in Li et al.'s paper, but also some other security features, such as traceability and identity protection.

1. Introduction

With the rapid growth of modern computer networks, increasing numbers of systems contain a certain quantity of service providing servers around the world and provide services via the Internet. It's important to verify the legitimacy of a remote user in a public environment before he/she can access the service. But traditional password based authentication schemes are mostly considered in single-server environments. They are unfit for the multi-server environments from two aspects. On the one hand, users need to register in each server and to store large sets of data, including identities and passwords. On the other hand, servers are required to store a verification table containing user identities and passwords. [1] firstly proposed a remote authentication scheme using smart card based on Elgamal's public key cryptosystem [2], which doesn't need to maintain verification tables. After that, numerous smart card based single-server authentication schemes using one-way hash functions had been proposed [3–11]. However, it is still hard for a user to use different smart cards to login and access different remote servers. This is because users still need to remember numerous sets of identities and passwords. In order to resolve this problem, several schemes have been proposed to the study of authentication and key agreement in the multi-server environment [12,13,20–25], all of which claim not to store verification tables. Most of these schemes can be divided into three categories: hash-based [12,13,25], symmetric cryptosystem based [24] and public key cryptosystem based [20–23]. Hash-based protocols are considered to be the most efficient. Among these schemes designed

---

* Corresponding author.
E-mail address: kpxue@ustc.edu.cn (K. Xue).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2013.07.004

Table 1
Notations used in Li et al.'s paper.

U_i	a user
S_j	a service providing server
CS	the control server
ID_i	the identity of U_i
SID_j	the identity of S_j
x	the master secret key
y	the secret number
b	a random number chosen by the user for registration
CID_i	the dynamic identity generated by U_i for authentication
SK	session key shared among the user, the server and CS
N_{i1}, N_{i2}, N_{i3}	random numbers chosen by U_i, S_j and CS
$h(\cdot)$	a one-way hash function
$\oplus$	the bitwise XOR operation
$\Vert$	the bitwise concatenation operation

for the multi-server environment, in [12], Hsiang and Shih proposed a dynamic identity and one-way hash-based remote user authentication protocol for multi-server architecture without a verification table. However, in 2011, Sood et al. [13] pointed that Hsiang and Shih's protocol cannot resist many types of security attacks, such as replay attack, impersonation attack and stolen smart card attack. Then Sood et al. proposed an improved scheme which is claimed to achieve user anonymity and resist different types of common security attacks. Recently, in [25], Li et al. found that Sood et al.'s protocol is still vulnerable to some types of known attacks, such as replay attack, stolen smart card attack and so on. Also the mutual authentication and key agreement phase of Sood et al.'s protocol cannot be successfully finished within some specific scenes. Furthermore, in [25], they proposed an improved dynamic identity based authentication and key agreement protocol for multi-server architecture, which is claimed to remove the aforementioned weaknesses of Sood et al.'s protocol. Unfortunately, our further research shows that Li et al.'s protocol contains several drawbacks and cannot resist some types of known attacks, such as leak-of-verifier attack, stolen smart card attack, eavesdropping attack, replay attack, Denial-of-Service attack and forgery attack and so on.

Meanwhile, identity protection is considered to be important for authentication and key agreement protocol design in single-server and multi-server architectures. Some existed researches adopt pseudonym [14] and dynamic identity [15,16] technologies to hide users' real identities. [12,13,25] also use dynamic identity technology to provide user anonymity, but the above discussion reveals that there are some security flaws of these schemes. Furthermore, they haven't provided traceability while providing user anonymity. Usually, there are conflicts between the anonymity and traceability objectives [17,18], which need to be well addressed. [12,13,25] don't provide the function of traceability while providing user anonymity. [19] proposes a scheme and claims that the scheme can provide the functions of traceability and anonymity simultaneously. But the pseudonym used in this scheme is fixed and can be considered as a user's another identity.

The rest of this paper is organized as follows: Section 2 gives the overview of Li et al.'s protocol; Section 3 points out the security weaknesses of the protocol in details. Section 4 gives our proposed protocol. Security and performance analysis of our proposed protocol is given in Section 5 and Section 6. At last, Section 7 presents the overall conclusion.

2. Overview of Li et al.'s protocol

In this section, we give the overview of Li et al.'s proposed protocol, which is an enhanced scheme based on Sood et al.'s protocol. We firstly summarize the notations used throughout Li et al.'s paper in Table 1. Li et al.'s protocol involves 3 kinds of participants: users (taking U_i for example), service providing servers (taking S_j for example), and the control server (CS). CS is a trusted third party (TTP) responsible for the registration and authentication of the users and the service providing servers. CS firstly chooses two security elements x and y. In the registration phase, S_j obtains $h(SID_j \Vert y)$ and $h(x \Vert y)$ from CS via a secure channel. U_i randomly selects a number b, and computes $A_i = h(b \Vert P_i)$, where P_i is U_i's password. After the initialization and the registration phases, U_i can get a smart card from CS via a secure channel. The following elements, $h(\cdot)$, $h(y)$ and b are stored in the smart card for the user U_i:

$$\begin{aligned} C_i &= h\big(ID_i \Vert h(y) \Vert A_i\big) \\ D_i &= B_i \oplus h(ID_i \Vert A_i) = h(ID_i \Vert x) \oplus h(ID_i \Vert A_i) \\ E_i &= B_i \oplus h(y \Vert x) = h(ID_i \Vert x) \oplus h(y \Vert x) \end{aligned} \tag{1}$$

In U_i's login phase, U_i inserts his smart card into a terminal and inputs his identity ID_i and password P_i, then computes $A_i^* = h(b \Vert P_i)$ and $C_i^* = h(ID_i \Vert h(y) \Vert A_i^*)$. If C_i^* is equal to the stored C_i, U_i is considered as a legitimate user. Else, the terminal rejects U_i's login request. After the verification, the authentication and key agreement phase takes place among U_i, S_j and CS, as depicted in Fig. 1. We introduce them as follows:

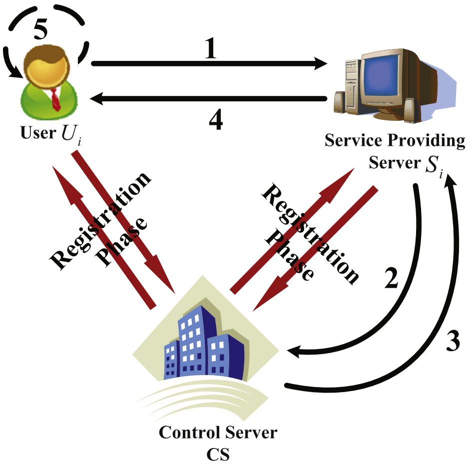

Fig. 1. Demonstration of registration, authentication and key agreement phases of Li et al.'s protocol.

Step 1: [1] $U_i \rightarrow S_j$: $\{F_i, G_i, P_{ij}, CID_i\}$.

U_i computes $B_i = D_i \oplus h(ID_i \| A_i)$ and generates a random number N_{i1}. Then U_i computes F_i, G_i, P_{ij}, CID_i as follows:

$$\begin{aligned} F_i &= h(y) \oplus N_{i1} \\ G_i &= h(B_i \| A_i \| N_{i1}) \\ P_{ij} &= E_i \oplus h\big(h(y) \| N_{i1} \| SID_j\big) \\ CID_i &= A_i \oplus h(B_i \| F_i \| N_{i1}) \end{aligned} \tag{2}$$

Then, U_i sends $\{F_i, G_i, P_{ij}, CID_i\}$ to S_j over a public channel.

Step 2: $S_j \rightarrow CS$: $\{F_i, G_i, P_{ij}, CID_i, SID_j, K_i, M_i\}$.

After receiving the message from U_i, the server S_j randomly selects a number N_{i2} and computes K_i, M_i as follows:

$$\begin{aligned} K_i &= h(SID_j \| y) \oplus N_{i2} \\ M_i &= h\big(h(x \| y) \| N_{i2}\big) \end{aligned} \tag{3}$$

Then S_j sends $\{F_i, G_i, P_{ij}, CID_i, SID_j, K_i, M_i\}$ to CS over the public channel.

Step 3: $CS \rightarrow S_j$: $\{Q_i, R_i, V_i, T_i\}$.

After receiving the message from S_j, CS gets $N_{i2} = K_i \oplus h(SID_j \| y)$ and $M^* = h(h(x \| y) \| N_{i2})$. Then CS verifies whether M^* is equal to the received M_i. If not, CS terminates the session; Else, the legitimacy of S_j is verified by CS. After that, CS computes the following elements:

$$\begin{aligned} N_{i1} &= F_i \oplus h(y) \\ B_i &= P_{ij} \oplus h\big(h(y) \| N_{i1} \| SID_j\big) \oplus h(y \| x) \\ A_i &= CID_i \oplus h(B_i \| F_i \| N_{i1}) \\ G_i^* &= h(B_i \| A_i \| N_{i1}) \end{aligned} \tag{4}$$

Then CS verifies whether G^* is equal to the received G_i. If not, CS terminates the session; Else, the legitimacy of U_i is verified by CS. CS randomly selects a number N_{i3}, and computes the following elements:

$$\begin{aligned} Q_i &= N_{i1} \oplus N_{i3} \oplus h(SID_j \| N_{i2}) \\ R_i &= h(A_i \| B_i) \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3}) \\ V_i &= h\big(h(A_i \| B_i) \| h(N_{i1} \oplus N_{i2} \oplus N_{i3})\big) \\ T_i &= N_{i2} \oplus N_{i3} \oplus h(A_i \| B_i \| N_{n1}) \end{aligned} \tag{5}$$

Then CS sends $\{Q_i, R_i, V_i, T_i\}$ to S_i over a public channel.

---

[1] In the description of [25], except for sending the message, this step is included in the login step.

Step 4: $S_j \rightarrow U_i$: $\{V_i, T_i\}$.
After receiving the message from CS, S_j computes

$$\begin{aligned} &N_{i1} \oplus N_{i3} = Q_i \oplus h(SID_j \| N_{i2}) \\ &h(A_i \| B_i) = R_i \oplus h(N_{i1} \oplus N_{i3} \oplus N_{i2}) \\ &V_i^* = h\big(h(A_i \| B_i) \| h(N_{i1} \oplus N_{i3} \oplus N_{i2})\big) \end{aligned} \tag{6}$$

Then S_j verifies whether V_i^* is equal to the received V_i. If not, S_j terminates the session; Else, the legitimacy of CS is verified by S_j. After that, S_j sends the message $\{V_i, T_i\}$ to U_i.

Step 5: After receiving the message from S_j, U_i computes to get V_i' as follows:

$$\begin{aligned} &N_{i2} \oplus N_{i3} = T_i \oplus h(A_i \| B_i \| N_{i1}) \\ &V_i' = h\big(h(A_i \| B_i) \| h\big(N_{i2} \oplus h(N_{i3}) \oplus h(N_{i1})\big)\big) \end{aligned} \tag{7}$$

Then U_j verifies whether V_i' is equal to the received V_i. If not, U_i terminates the session; Else, the legitimacy of CS and S_j is verified by U_i.

Finally, U_i, S_j and CS can separately compute the shared session key SK as follows:

$$SK = h\big(h(A_i \| B_i) \| (N_{i1} \oplus N_{i2} \oplus N_{i3})\big) \tag{8}$$

3. Security weakness analysis of the protocol

Although in [25], the authors claimed that their protocol can resist many types of security attacks. Unfortunately, our further research shows that Li et al.'s protocol contains several drawbacks and cannot resist some types of known attacks, such as replay attack, Denial-of-Service attack, smart card forgery attack and eavesdropping attack. The analysis in details is described as follows.

3.1. *Replay attack and Denial-of-Service attack*

Assume that a malicious attacker can eavesdrop the first sending message from a legitimate user to the server S_k in Step 1 of the authentication and key agreement phase. If the message $\{F_i, G_i, P_{ij}, CID_i\}$ is eavesdropped, replay attacks can easily be launched by retransmitting $\{F_i, G_i, P_{ij}, CID_i\}$ to S_j. This type of attacks can trick the server S_k and CS into implementing the following Steps 2–4. Moreover, S_K and CS cannot identify the message replayed by the malicious attackers. Even if the user cannot get the final correct session key SK, the server S_k and CS have made great consumption of computing resources, communication resources and storage resources. A large number of replay attacks launched at the same time will form a Denial-of-Service attack, which prevents normal visits from legitimate users.

3.2. *Internal attack*

In Li et al.'s protocol, attackers cannot get $h(y)$ and $h(y \| x)$ with stolen smart card attack. But it cannot resist insider attack, where the inside malicious user knows its password. Assume there is an inside malicious user who has a legitimate smart card. From the elements stored in the smart card, the malicious user can straightly get $h(y)$. The malicious attacker U_f can firstly compute his/her B_f $(= D_f \oplus h(ID_f \| A_f))$, and then computes $h(y \| x) = E_f \oplus B_f$. By knowing $h(y)$ and $h(y \| x)$, the attacker can further launch eavesdropping attacks to get the session key shared among any other users, the related service providing servers and CS.

3.3. *Smart card forgery attack*

Li et al.'s protocol lacks of verification of A_i and B_i by CS, thus a malicious attacker known $h(y)$ and $h(y \| x)$ in advance can arbitrarily forge a new smart card. If the attacker wants to forge U_s's smart card, he/she firstly sets $A_s = Num1$ and $B_i = Num2$, where $Num1$ and $Num2$ are two random numbers with the same length as A_i, B_i. Elements of a forgery smart card can be further set as

$$\begin{aligned} &C_s = h\big(ID_s \| h(y) \| A_s\big) = C_s = h\big(ID_s \| h(y) \| Num1\big) \\ &D_s = B_s \oplus h(ID_s \| A_s) = Num2 \oplus h(ID_s \| Num1) \\ &E_s = B_s \oplus h(y \| x) = Num2 \oplus h(y \| x) \end{aligned} \tag{9}$$

Then if the malicious attacker wants to access the service providing server S_j by using this forgery smart card, the first message (in Step 1) can be computed as

$$
\begin{aligned}
&F_s = h(y) \oplus N_{s1} \\
&G_s = h(B_s \| A_s \| N_{s1}) = h(Num2 \| Num1 \| N_{s1}) \\
&P_{sj} = E_s \oplus h\big(h(y) \| N_{s1} \| SID_j\big) = Num2 \oplus h(y \| x) \oplus h\big(h(y) \| N_{s1} \| SID_j\big) \\
&CID_s = A_s \oplus h(B_s \| F_s \| N_{s1}) = Num1 \oplus h(Num2 \| F_s \| N_{s1})
\end{aligned}
\tag{10}
$$

Following Li et al.'s protocol, this message can successfully pass the legitimacy verification by CS and S_j. If the random numbers separately chosen by S_j and CS are N_{s2} and N_{s3}, the malicious attacker, S_j and CS can successfully agree on a common session key $SK = h(h(Num1 \| Num2) \| (N_{s1} \oplus N_{s2} \oplus N_{s3}))$.

3.4. Eavesdropping attack

Assume the authentication and key agreement phase takes place among the legitimate user U_m, the service providing server S_n and the control server CS.

There is a malicious attacker who has the ability of eavesdropping all of the messages exchanged among these three participants. Furthermore, the malicious attacker is assumed to have known $h(y)$, $h(y \| x)$ in advance. The first message $\{F_m, G_m, P_{mn}, CID_m\}$ is sent from U_m. From F_m, N_{m1} can been easily obtained as follows:

$$
N_{m1} = h(y) \oplus F_m \tag{11}
$$

Next, E_m can be extracted from P_{mn}, then B_m can be extracted from E_m. The details are described as follows:

$$
\begin{aligned}
&E_m = P_{mn} \oplus h\big(h(y) \| N_{m1} \| SID_n\big) \\
&B_m = E_m \oplus h(y \| x)
\end{aligned}
\tag{12}
$$

After that from CID_m, A_m can also be easily extracted as

$$
A_m = CID_m \oplus h(B_m \| F_m \| N_{m1}) \tag{13}
$$

From the above process, only a sending message via a public channel can leak crucial security information (A_m, B_m, N_{m1}) of U_m. Also E_m stored in U_m's smart card can also be got. Although because of the user anonymity support, the malicious attacker cannot obtain U_m's identity ID_m to compute C_m and D_m, but nextly we will describe how to extract the final session key SK.

After eavesdropping the message sent in Step 3 or Step 4, the malicious attacker can extract $N_{m2} \oplus N_{m3}$ from T_m as follows

$$
N_{m2} \oplus N_{m3} = T_m \oplus h(A_m \| B_m \| N_{i1}) \tag{14}
$$

Now, the malicious attacker can compute the final session key negotiated among U_m, S_n and CS. Furthermore, he/she can decrypted all the encrypted data between U_m and S_n.

3.5. Masquerade attack to pose as a legitimate user

After successfully obtaining security information of a legitimate user (such as U_m) via the eavesdropping attack described in Section 3.4, the attacker can launch the masquerade attack to act as the legitimate user. By means of the internal attack, the malicious attackers can know $h(y)$ and $h(y \| x)$. By means of the eavesdropping attack, the malicious attacker can further compute A_m, B_m and E_m. By virtue of these information, malicious attackers can pose as U_m to launch authentication and key agreement phase to any other service providing server (take S_p for example) and CS.

Firstly, the malicious attacker randomly selects a number N_{MA} and can successfully forge the first step message to pretend to be U_m:

$$
\begin{aligned}
&F_m = h(y) \oplus N_{MA} \\
&G_m = h(B_m \| A_m \| N_{MA}) \\
&P_{mp} = E_m \oplus h\big(h(y) \| N_{MA} \| SID_p\big) \\
&CID_m = A_m \oplus h(B_m \| F_m \| N_{MA})
\end{aligned}
\tag{15}
$$

Then assume S_p and CS separately select random numbers N_{m2} and N_{m3}, and Steps 2–4 are performed normally. Then the malicious attacker, S_j and CS "successfully" agree on a session key $SK = h(h(A_m \| B_m) \| (N_{MA} \oplus N_{m2} \oplus N_{m3}))$. But unfortunately S_p and CS mistakenly believe that they are communicating with the legitimate user U_m.

Table 2
Notations used in our proposed protocol.

U_i	a user
S_j	a service providing server
CS	the control server
ID_i	the identity of U_i
SID_j	the identity of S_j
TS_i	timestamp value generated by U_i
x	the secret number only known to CS
y	the secret number only known to CS
b	a random number chosen by the user
d	a random number chosen by the service providing server
PID_i	the protected pseudonym identity of U_i
$PSID_j$	the protected pseudonym identity of S_j
SK	session key shared among the user, the server and CS
N_{i1}, N_{i2}, N_{i3}	random numbers chosen by U_i, S_j and CS
$h(\cdot)$	a one-way hash function
$\oplus$	the bitwise XOR operation
$\|$	the bitwise concatenation operation

3.6. Masquerade attack to pose as a legitimate service providing server

Firstly, we assume that the malicious attacker has eavesdropped a message sent from S_n to get K_i and M_i. Furthermore, we assume that a legitimate user U_m's security information has been leaked to the malicious attacker based on the internal attack and the eavesdropping attack. When U_m wants to login the server S_n, he/she selects a random number N_{m1} and sends the first message in Step 1 ($\{F_m, G_m, P_{mn}, CID_m\}$) to the service providing server S_n. The malicious attacker can attack the real server S_n to be down and masquerades to be S_n himself/herself. After eavesdropping this message, the malicious attacker can attach K_i and M_i in the first message: $\{F_m, G_m, P_{mn}, CID_m, SID_n, K_i, M_i\}$. This message can also successfully pass CS's verification. N_{m3} is the random number selected by CS. After implementing of Step 3 and Step 4, the user U_m and CS can compute the session key as

$$SK = h\big(h(A_m \| B_m) \| h(N_{m1} \oplus N_{i2} \oplus N_{m3})\big) \tag{16}$$

And unfortunately U_m mistakenly believes that he/she is communicating with the legitimate true S_n. Although the malicious attacker cannot extract the random number N_{i2} from K_i, he/she still can exact the session key SK by means of "masquerade attack as a legitimate user" described in Section 3.5. So the malicious attacker cannot only masquerade to be the real server, but also decrypt the encrypted data sent from the user in the dark.

4. Our proposed improved protocol

In this section, we will describe an improved protocol to make up the security weaknesses of Li et al.'s protocol. Our protocol contains three kinds of participants (the user, the service providing server and the controlling server) and contains three phases: 1) initialization and registration phase; 2) login phase; 3) authentication and key agreement phase. Because the notions are different in using from those of Li et al.'s protocol in protocol designing and some new notions are defined, here we firstly give the notations used in our proposed protocol (summarize in Table 2). We show the protocol in Fig. 2 and provide more details as follows.

4.1. Initialization and registration phase

Assume the control server CS is a trusted third party responsible for registration and authentication of users and service providing servers. CS chooses two random numbers x and y.

The registration phase of the user U_i is as follows:

Step 1: The user U_i freely chooses his/her password P_i, and a number b. Then U_i computes $A_i = h(b \| P_i)$, and submits the message $\{ID_i, b, A_i\}$ to CS via a secure channel.
Step 2: After receiving the message, CS first verifies user's legitimacy. Then, CS computes $PID_i = h(ID_i \| b)$, $B_i = h(PID_i \| x)$. CS sends B_i to U_i via a secure channel.
Step 3: After receiving the smart card, U_i computes $C_i = h(ID_i \| A_i)$ and $D_i = B_i \oplus h(PID_i \oplus A_i)$. Then U_i enters C_i, D_i, $h(\cdot)$ and b into the smart card. At last, the smart card contains (C_i, D_i, $h(\cdot)$, b).

For the service providing server S_j, he/she first chooses a random number d, and uses his/her identity S_j to register with CS. CS computes $PSID_j = h(SID_j \| d)$, $BS_j = h(PSD_j \| y)$. Then CS sends BS_j to S_j via a secure channel. S_j stores BS_j and d in his/her memory.

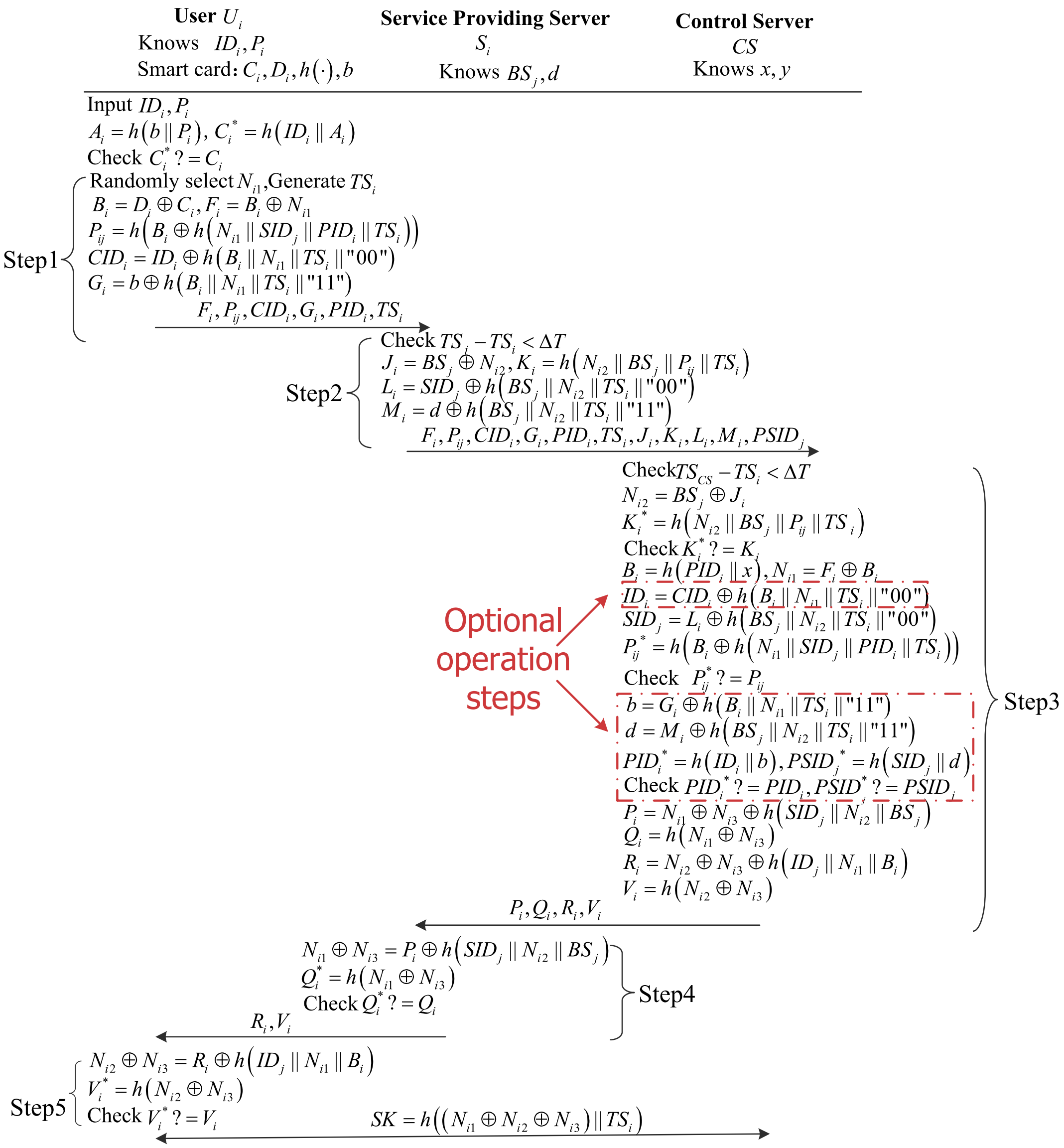

Fig. 2. The implementation phases of our proposed protocol.

4.2. *The login phase*

When the user U_i wants to login to access the server S_j, U_i inserts his smart card into a terminal and inputs his/her identity ID_i and password P_i, then computes $A_i^* = h(b\|P_i)$ and $C_i^* = h(ID_i\|A_i^*)$. If C_i^* is equal to the stored C_i, U_i is considered as a legitimate user. Otherwise, the terminal rejects U_i's login request.

4.3. *The authentication and key agreement phase*

Step 1: $U_i \rightarrow S_j$: $\{F_i, P_{ij}, CID_i, G_i, PID_i, TS_i\}$.
U_i chooses a random number N_{i1} and generates a current timestamp value TS_i. Then U_i computes B_i, F_i, CID_i, P_{ij}, G_i as follows:

$$B_i = D_i \oplus C_i$$

$$F_i = B_i \oplus N_{i1}$$

$$P_{ij} = h\big(B_i \oplus h(N_{i1}\|SID_j\|PID_i\|TS_i)\big)$$

$$\begin{aligned}
&CID_i = ID_i \oplus h(B_i \| N_{i1} \| TS_i \| \text{“00”}) \\
&G_i = b \oplus h(B_i \| N_{i1} \| TS_i \| \text{“11”})
\end{aligned} \tag{17}$$

where “00” is a 2-bit binary-“0”, and “11” is a 2-bit binary-“1”.

Then, U_i sends $\{F_i, P_{ij}, CID_i, G_i, PID_i, TS_i\}$ to S_j over a public channel.

Step 2: $S_j \to CS$: $\{F_i, P_{ij}, CID_i, G_i, PID_i, TS_i, J_i, K_i, L_i, M_i, PSID_j\}$.

After receiving the message from U_i, the server S_j first checks whether the session delay is within the tolerable time interval ΔT. Assume the current time is TS_j. If $TS_j - TS_i > \Delta T$, the session is timeout and S_j terminates the session; Otherwise, S_j continues to perform the following operations.

S_j randomly selects a number N_{i2} and computes J_i, K_i, L_i, M_i as follows:

$$\begin{aligned}
&J_i = BS_j \oplus N_{i2} \\
&K_i = h(N_{i2} \| BS_j \| P_{ij} \| TS_i) \\
&L_i = SID_j \oplus h(BS_j \| N_{i2} \| TS_i \| \text{“00”}) \\
&M_i = d \oplus h(BS_j \| N_{i2} \| TS_i \| \text{“11”})
\end{aligned} \tag{18}$$

where “00” is a 2-bit binary-“0”, and “11” is a 2-bit binary-“1”.

Then S_j sends $\{F_i, P_{ij}, CID_i, G_i, PID_i, TS_i, J_i, K_i, L_i, M_i, PSID_j\}$ to CS over the public channel.

Step 3: $CS \to S_j$: $\{P_i, Q_i, R_i, V_i\}$.

After receiving the message from S_j, CS first checks whether the session delay is within the allow time interval ΔT. Assume the current time is TS_{CS}. If $TS_{CS} - TS_i > \Delta T$, the session is timeout and CS terminates the session; CS continues to perform the following operations.

CS computes $BS_j = h(PSID_j \| y)$, $N_{i2} = J_i \oplus BS_j$ and $K^* = h(N_{i2} \| BS_j \| P_{ij} \| TS_i)$. Then CS verifies whether K_i^* is equal to the received K_i. If not, CS terminates the session; Otherwise, CS continues to perform the following operations. CS computes the following elements:

$$\begin{aligned}
&B_i = h(PID_i \| x) \\
&N_{i1} = F_i \oplus B_i \\
&ID_i = CID_i \oplus h(B_i \| N_{i1} \| TS_i \| \text{“00”}) \\
&SID_i = L_i \oplus h(BS_j \| N_{i2} \| TS_i \| \text{“00”}) \\
&P_{ij}^* = h\big(B_i \oplus h(N_{i1} \| SID_j \| PID_i \| TS_i)\big)
\end{aligned} \tag{19}$$

Then CS verifies whether P_{ij}^* is equal to the received P_{ij}. If not, CS terminates the session; Otherwise, CS continues to compute the following elements:

$$\begin{aligned}
&b = G_i \oplus h(B_i \| N_{i1} \| TS_i \| \text{“11”}) \\
&d = M_i \oplus h(BS_j \| N_{i2} \| TS_i \| \text{“11”}) \\
&PID_i^* = h(ID_i \| b) \\
&PSID_j^* = h(SID_j \| d)
\end{aligned} \tag{20}$$

Then CS verifies whether $PID_i^* = PID_i$ and $PSID_j^* = PSID_j$. If not, CS terminates the session; Otherwise, CS makes sure the messages are from real U_i and S_j. After the verification, CS randomly selects a number N_{i3}, and computes P_i, Q_i, R_i V_i as follows:

$$\begin{aligned}
&P_i = N_{i1} \oplus N_{i3} \oplus h(SID_j \| N_{i2} \| BS_j) \\
&Q_i = h(N_{i1} \oplus N_{i3}) \\
&R_i = N_{i2} \oplus N_{i3} \oplus h(ID_i \| N_{i1} \| B_i) \\
&V_i = h(N_{i2} \oplus N_{i3})
\end{aligned} \tag{21}$$

Then CS sends $\{P_i, Q_i, R_i, V_i\}$ to S_i over a public channel.

Step 4: $S_j \to U_i$: $\{R_i, V_i\}$.

After receiving the message from CS, S_j firstly computes to get the following elements:

$$\begin{aligned}
&N_{i1} \oplus N_{i3} = P_i \oplus h(SID_j \| N_{i2} \| BS_j) \\
&Q_i^* = h(N_{i1} \oplus N_{i3})
\end{aligned} \tag{22}$$

Then S_j verifies whether Q_i^* is equal to the received Q_i. If not, S_j terminates the session; Otherwise, the legitimacy of CS and U_i is verified by S_j. After that, S_j sends the message $\{R_i, V_i\}$ to U_i.

Table 3
Security functionality comparison of our protocol and two other related protocols.

Security functionality	Our proposed protocol	Li et al.'s protocol (2012)	Sood et al.'s protocol (2011)
Identity protection and user anonymity	Yes	Yes	Yes
Dynamic identity updating	Yes	Yes	No
Traceability	Yes	No	No
Mutual authentication	Yes	Yes	Yes
Session key agreement	Yes	Yes	Yes
Password updating	Yes	Yes	Yes
Resistance of insider attack and smart card forgery attack	Yes	No	No
Resistance of stolen smart card attack	Yes	Yes	No
Resistance of replay attack	Yes	No	No
Resistance of Denial-of-Service attack	Yes	No	No
Resistance of eavesdropping attack	Yes	No	No
Resistance of masquerade attack	Yes	No	No

Step 5: After receiving the message from S_j, U_i computes to get V_i^* as follows:

$$\begin{aligned} &N_{i2} \oplus N_{i3} = R_i \oplus h(ID_i \| N_{i1} \| B_i) \\ &V_i^* = h(N_{i2} \oplus N_{i3}) \end{aligned} \tag{23}$$

Then U_j verifies whether V_i^* is equal to the received V_i. If not, U_i terminates the session; Otherwise, the legitimacy of CS and S_j is verified by U_i.

Finally, U_i, S_j and CS can separately compute the common session key SK as follows:

$$SK = h\big((N_{i1} \oplus N_{i2} \oplus N_{i3}) \| TS_i\big) \tag{24}$$

4.4. The password updating phase

After password based verification in the registration phase, the user U_i's password P_i does not appear in B_i. Thus, password updating/changing can happen in anytime without CS's help. U_i can update the parameters in his/her smart card:

$$\begin{aligned} &C_i' = h\big(ID_i \| A_i'\big) \\ &D_i' = B_i \oplus h\big(PID_i \oplus A_i'\big) \end{aligned} \tag{25}$$

Meanwhile, in order to keep password consistency between U_i and CS, U_i needs to submit his/her ID_i and A_i' with new password P_i' to CS via a secure channel. CS updates U_i's password in its verification table. However, the submission process does not have to happen after the password updating immediately.

4.5. The dynamic identity updating phase

In order to prevent malicious attackers linking eavesdropped messages of different sessions, we can update the user's PID periodically to provide security. U_i re-selects a random number $b^\#$, and computes $A_i^\# = h(b^\# \| P_i)$. Then U_i submits $\{ID_i, b^\#, A_i^\#\}$ to CS. After verifying U_i's legitimacy, CS recomputes $PID_i^\# = h(ID_i \| b^\#)$, $B_i^\# = h(PID_i^\# \| x)$ and submits $B_i^\#$ to U_i via a secure channel. After receiving $B_i^\#$, U_i computes $C_i^\# = h(ID_i \| A_i^\#)$, $D_i^\# = B_i^\# \oplus h(PID_i^\# \oplus A_i^\#)$. At last the smart card is updated to $\{C_i^\#, D_i^\#, h(\cdot), b^\#\}$. Now U_i's protected pseudonym identity PID_i is dynamically changed to $PID_i^\#$.

Service providing servers can also periodically update their protected pseudonym identities. Take S_j for example, S_j re-selects a random number $d^\#$, and use his/her identity S_j to register with CS. CS computes $PSID_j^\# = h(SID_j \| d^\#)$, $BS_j^\# = h(PSD_j^\# \| y)$. Then CS sends $BS_j^\#$ to S_j via a secure channel. S_j updates $BS_j^\#$ and $d^\#$ in his/her memory.

5. Security analysis of our protocol

In this section, we summarize security analysis of our proposed protocol and compare it with other two related protocols. First we list security functionality comparison among our protocol and other two related protocols in Table 3. It demonstrates that our protocol is more secure than other two related protocols [13,25].

Here, we discuss the security features of our proposed protocol as follows in details:

A. **Identity protection and user anonymity:** For the user U_i, we use PID_i instead of ID_i. By using protected pseudonym identities of users instead of real ones, the malicious attacker cannot get users' real identities. Further, in our scheme, while providing authentication of users, service providing servers cannot know users' real identities either. In this way, our protocol provides user anonymity, which can prevent the leakage of private user identities and server identities to malicious attackers. Updating users' pseudonym identities periodically and dynamically can prevent the malicious attacker linking eavesdropped messages of different sessions from the same user.

B. **Traceability:** [25] doesn't provide traceability, but in our scheme, *CS* can still extract users' real identities and link them with protected pseudonym identities, while provide the function of anonymity between the user U_i and the service providing server S_j. ID_i can be retrieved from received CID_i in formula (19). This makes our protocol have the feature of traceability. This is a newly-added function in our proposed protocol different from Li et al.'s protocol.

C. **Mutual authentication:** Based on the authentication and key agreement phase described in Section 4.3, our proposed scheme can provide mutual authentication among the user (U_i), the service providing server (S_j) and the control server. In Step 3, by checking whether $PID_i^* = PID_i$ and $PSID_j^* = PSID_j$, *CS* can verify the legitimacy of U_i and S_j. In Step 4, by checking whether $Q_i^* = Q_i$, S_j can verify the legitimacy of U_i and *CS*. In Step 5, by checking whether $V_i^* = V_i$, U_i can verify the legitimacy of S_j and *CS*.

D. **Session key agreement:** In order to protect the data communication between the user U_i and the service providing server S_j, a session key need to be negotiated between them in advance, which can further derive encryption keys and MAC keys. In this paper, we only use the hash function and the XOR operation to design a simple but efficient key agreement scheme. By securely exchanging N_{i1}, N_{i2} and N_{i3}, U_i and S_j can separately compute the common session key as in formula (24).

E. **Password updating/changing:** By the method described in Section 4.4, user's password updating/changing can happen in anytime, which will affect the parameters C_i, D_i in user's smart card and the verification table which is maintained by *CS*.

F. **Resistance of insider attack and smart card forgery attack:** As in Section 3.2, within Li et al.'s protocol, an internal attack can cause information leakage. $h(y)$ and $h(y\|x)$ are the common parameters for all users, which can further launch eavesdropping attacks, smart card forgery attacks, masquerade attacks and so on. In our proposed protocol, we do not straightly use $h(y)$, $h(x)$, $h(y\|x)$ directly. Take the user U_f as insider attacker for example. We use $B_f = h(PID_f\|x)$ and compute to get C_f, D_f in his/her smart card. U_f cannot guess to generate parameters of any other users' smart cards and cannot masquerade as any other legitimate user by using security information of himself/herself.

G. **Resistance of stolen smart card attack:** In our proposed protocol, we firstly assume that if a smart card is stolen, physical protection methods cannot prevent malicious attackers to get the stored security elements. Still take U_i for example, if his/her smart card is stolen, the malicious attacker can get $(C_i, D_i, h(\cdot), b)$. But without inputting the right password P_i, the malicious attacker cannot compute A_i, and further extract B_i from D_i.

H. **Resistance of replay attack and Denial-of-Service attack:** Firstly the timestamp value is used in our proposed protocol which makes the malicious attacker cannot use an early message to launch replay attacks. This makes replay attacks and Denial-of-Service attacks hard to be launched. Using P_{ij} and TS_i in computing K_i avoids the case in Li et al.'s protocol: If K_i and M_i attached by the service providing server S_j are eavesdropped, they can be used to launch replay attacks, which are described in Section 3.6. Moreover using and verifying timestamp can reduce the success rate of replay attacks.

I. **Resistance of eavesdropping attack:** The malicious attacker cannot extract private security information from eavesdropped messages over public channels. Different from Li et al.'s protocol, because of using *PID* in computing B_i and not sharing $h(x)$ and $h(y\|x)$ between *CS* and every user, the malicious attacker cannot use one user's elements to extract any other user's security elements in our proposed protocol. Moreover, the malicious attacker cannot compute $N_{i1} \oplus N_{i2} \oplus N_{i3}$, so *SK* cannot be computed by the malicious attacker.

J. **Resistance of masquerade attack:** The malicious attacker cannot derive U_i's security information from eavesdropped sending messages among U_i, S_j and *CS*; Meanwhile, the malicious attacker cannot forge other user's smart card from known security information of a malicious inside user. Furthermore, using the timestamp value prevents replay of the first message. Because of the above 3 reasons, users cannot be masqueraded by malicious attackers. Because of using P_{ij} and TS_i in computing K_i, the malicious attacker cannot replay S_j's message to attach to the end of the message in Step 1, thus servers cannot be masqueraded by malicious attackers.

6. Performance analysis

In Sood et al.'s protocol [13], Li et al.' protocol [25] and our proposed protocol, the protocol implementation delay is all mainly caused by the login phase, the authentication and key agreement phase, so in this section, we only take the login phase, authentication and session key agreement phase into consideration. Take our proposed protocol for example, the protocol implementation delay mainly includes the delay caused by communication overhead ($T_{Communication}$) and the delay caused by computation overhead ($T_{Computing}$), which can be further described as follows:

$$\begin{aligned}
&T_{total\text{-}delay} = T_{Communication} + T_{Computation}\\
&T_{Communication} = T_{U_i \to S_j} + T_{S_j \to CS} + T_{CS \to S_j} + T_{S_j \to U_i}\\
&T_{Computation} = T_{Step\,1} + T_{Step\,2} + + T_{Step\,3} + T_{Step\,4} + T_{Step\,5}
\end{aligned} \tag{26}$$

Table 4
Computation overhead comparison of our protocol and two other related protocols.

Protocols	The login phase (U_i)	The authentication and key agreement phase (U_i, S_j, CS)
Our proposed protocol	$2T_{hash}$	$19T_{hash} + (optional)5T_{hash}$: $6T_{hash}$ (U_i), $5T_{hash}$ (S_j), $8T_{hash} + (optional)5T_{hash}$ (CS)
Li et al.'s protocol (2012)	$2T_{hash}$	$25T_{hash}$: $8T_{hash}$ (U_i), $4T_{hash}$ (S_j), $13T_{hash}$ (CS)
Sood et al.'s protocol (2011)	$1T_{hash}$	$24T_{hash}$: $9T_{hash}$ (U_i), $4T_{hash}$ (S_j), $11T_{hash}$ (CS)

Table 5
Message length comparison of our protocol and two other related protocols.

Protocols	Message length (byte)			
	$U_i \to S_j$	$S_j \to CS$	$CS \to S_j$	$S_j \to U_i$
Our proposed protocol	83	163	64	32
Li et al.'s protocol (2012)	64	112	64	32
Sood et al.'s protocol (2011)	64	80	64	32

where $T_{A \to B}$ represents the implementation delay of signaling communication from A to B. $T_{Step\, i}$ represents the implementation delay of computation happened in *Step i*.

In this section, we evaluate the computation overhead, communication overhead of our proposed protocol and give the comparisons with the other two related protocols: Li et al.' protocol [25] and Sood et al.'s protocol [13]. Storage overhead analysis is given in Section 6.3. Before analyzing in details, we first set the notation T_{hash} as the time of computing the hash operation. Because XOR and "$\|$" operations require very little computation overhead, they usually can be omitted.

Briefly, from the analysis in this section, compared with the related works, while providing relatively more security feature and the higher security level, our proposed scheme doesn't increase too much overhead.

6.1. Computation overhead analysis

Computation overhead comparison of our protocol and the other two related protocols are given in Table 4. As in [13,25], because the protocol implementation delay is all mainly caused by the login phase, the authentication and key agreement phase, we only take these two phases into consideration. Different from the protocol description in [25], in this paper, the login phase description of Li et al.'s protocol relates only to user legitimacy verification by terminal. Similarly, we merge Step 2 of the login phase of [25] into the first step of the authentication and key agreement phase in this paper. The similar decryption modification is adopted to Sood et al.'s protocol [13]. Furthermore, there is separately one time of hash computation for computing SK for the user, the service providing server and CS, which is not mentioned in Table 4. From Table 4, it is obvious that our protocol almost has the same computation overhead as the other two related protocols. In the authentication and key agreement phase of our proposed protocol, CS have five optional hash operations, which provide the function of traceability.

6.2. Communication overhead analysis

Our proposed protocol and other two related protocols all require 4 times of message transmission in the authentication and key agreement phase. Take U_i, S_j and CS for example, four times of message transmission are $U_i \to S_j$, $S_j \to CS$, $CS \to S_j$ and $S_j \to U_i$, which is demonstrated in Fig. 1. Assuming the length of the each hash value is 128-bit, the length of the timestamp value is 24-bit, and each of the other transmitted elements is also 128-bit. The message length comparison of these three protocols is given in Table 5.

Because of providing more security features, the message transmission overhead is increasing accordingly in our proposed protocol. Nevertheless, this increasing byte is acceptable and can be omitted in the protocol design.

6.3. Storage overhead analysis

Just as Li et al.'s protocol and Sood et al.'s protocol, for each service providing server, it doesn't need to maintain a verification table in our proposed protocol. Meanwhile, CS maintains a verification table which is only required to be searched in the registration phase. CS doesn't need to use the verification table in the authentication and key agreement phase. Each user only needs to have a smart card. Each service providing server (take S_j for example) only needs to store BS_j and a randomly chosen number d obtained in the registration phase. Besides the verification table, CS only knows x and y.

7. Conclusions and future works

In this paper, based on discussing the security weaknesses of Li et al.'s protocol, we propose an improved dynamic pseudonym identity based authentication and key agreement protocol, which is suitable for the multi-server environment. Compared with related protocols, our proposed protocol is demonstrated to satisfy all the essential security requirements for authentication and key agreement in the multi-server environment. Meanwhile, in comparison with Li et al.'s protocol and Sood et al.'s protocol, our proposed protocol keeps efficiency. In the future, we will introduce suitable solutions to further reduce the computation overhead and improve protocol performance without compromising security. Moreover, in our protocol, we use timestamp value to resist replay attack, which requires loose clock synchronization. We will further study how to use random number or serial number to replace the use of timestamp value.

Acknowledgments

This work is supported by the National S&T Major Project of China under Grant Nos. 2010ZX03003-002, 2011ZX03005-006, the National Natural Science Foundation of China under Grant No. 61170231.

References

[1] M.S. Hwang, L.H. Li, A new remote user authentication scheme using smart cards, IEEE Transactions on Industrial Electronics 46 (1) (2000) 28–30.
[2] T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms, Advances in Cryptology 196 (1985) 10–18.
[3] T. Hwang, Y. Chen, C.S. Laih, Non-interactive password authentication without password tables, in: Proc. of IEEE Region 10 Conference on Computer and Communication System, September 1990.
[4] C.C. Chang, T.C. Wu, Remote password authentication with smart cards, IEE Proc. Computers and Digital Techniques 138 (3) (1999) 165–168.
[5] 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 57 (2) (2010) 793–800.
[6] H. Chien, J. Jan, Y. Tseng, An efficient and practical solution to remote authentication: smart card, Computers & Security 21 (4) (2002) 372–375.
[7] W. Yang, S. Shieh, Password authentication schemes with smart card, Computers & Security 18 (8) (1999) 727–733.
[8] A.K. Awashti, S. Lal, An enhanced remote user authentication scheme using smart cards, IEEE Transactions on Industrial Electronics 50 (2) (2004) 583–586.
[9] J. Xu, W.T. Zhu, D.G. Feng, An improved smart card based password authentication scheme with provable security, Computer Standards & Interfaces 31 (4) (2009) 723–728.
[10] R.G. Song, Advanced smart card based password authentication protocol, Computer Standards & Interfaces 32 (5–6) (2010) 321–326.
[11] M. Badra, P. Urien, Introducing smartcards to remote authenticate passwords using public key encryption, in: Proc. of 2004 IEEE Symposium on Advances in Wired and Wireless Communications, NJ, USA, 2004, pp. 123–126.
[12] H.C. Hsiang, W.K. Shih, Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment, Computer Standards & Interfaces 31 (6) (2009) 1118–1123.
[13] S.K. Sood, A.K. Sarje, K. Singh, A secure dynamic identity based authentication protocol for multi-server architecture, Journal of Network and Computer Applications 34 (2) (2011) 609–618.
[14] M. Badra, P. Urien, IETF Draft: EAP-Double-TLS Authentication Protocol, draft-badraeap-double-tls-05.txt, June 2006.
[15] M. Badra, P. Urien, Adding client identity protection to EAP-TLS SmartCards, in: IEEE Wireless Communications and Networking Conference, IEEE WCNC 2007, Hong Kong, 2007.
[16] P. Urien, M. Badra, Secure access modules for identity protection over the EAPTLS: Smartcard benefits for user anonymity in wireless infrastructures, in: Proc. of the 2006 International Conference on Security and Cryptography, SECRYPT2006, Barcelona, Spain, July 2006, pp. 157–163.
[17] J.Y. Sun, C. Zhang, Y.C. Zhang, Y.G. Fang, SAT: A security architecture achieving anonymity and traceability in wireless mesh networks, IEEE Transactions on Dependable and Secure Computing 8 (2) (March–April 2011) 295–307.
[18] W. Hu, K.P. Xue, P.L. Hong, C.C. Wu, ATCS: A novel anonymous and traceable communication scheme for vehicular ad hoc networks, International Journal of Network Security 13 (2) (September 2011) 71–78.
[19] S. Kim, H.S. Rhee, J.Y. Chun, D.H. Lee, Anonymous and traceable authentication scheme using smart cards, in: Proceedings of the 2008 International Conference on Information Security and Assurance, ISA2008, April 2008, pp. 162–165.
[20] W.J. Tsuar, C.C. Wu, W.B. Lee, An enhanced user authentication scheme for multi-server Internet services, Applied Mathematics and Computation 170 (1) (2005) 258–266.
[21] I.C. Lin, M.S. Hwang, L.H. Li, A new remote user authentication scheme for multi-server architecture, Journal of Future Generation Computer System 19 (1) (2003) 13–22.
[22] Y. Yang, S. Wang, F. Bao, J. Wang, R. Deng, New efficient user identification and key distribution scheme providing enhanced security, Computers & Security 23 (8) (2004) 697–704.
[23] W.J. Tsuar, C.C. Wu, W.B. Lee, A smart card based remote scheme for password authentication in multi-server Internet services, Computer Standards & Interfaces 27 (1) (2004) 39–51.
[24] W.S. Juang, Efficient multi-server password authenticated key agreement using smart cards, IEEE Transactions on Consumer Electronics 50 (1) (2004) 251–255.
[25] X. Li, Y.P. Xiong, J. Ma, W.D. Wang, An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards, Journal of Network and Computer Applications 35 (2) (2012) 763–769.