Wireless Pers Commun (2014) 79:1621–1634
DOI 10.1007/s11277-014-1948-z

A Provably Secure Multi-server Based Authentication Scheme

Kuo-Hui Yeh

Abstract With the rapid growth of electronic commerce and demand on variants of Internet based applications, the system providing resources and business services often consists of many servers around the world. So far, a variety of authentication schemes have been published to achieve remote user authentication on multi-server communication environment. Recently, Pippal et al. proposed a multi-server based authentication protocol to pursue the system security and computation efficiency. Nevertheless, based on our analysis, the proposed scheme is insecure against user impersonation attack, server counterfeit attack, and man-in-the-middle attack. In this study, we first demonstrate how these malicious attacks can be invoked by an adversary. Then, a security enhanced authentication protocol is developed to eliminate all identified weaknesses. Meanwhile, the proposed protocol can achieve the same order of computation complexity as Pippal et al.'s protocol does.

Keywords Authentication · Multi-server · Privacy · Security · Smart card

1 Introduction

Following the advances in network technologies and the widespread distribution of remote system backup, lots of multi-server based applications have been deployed to make legitimate user access network service more conveniently and efficiently. As password based authentication scheme provides an efficient and accurate way to identify valid remote user and at the same time preserves the secrecy of communication, a lot of password based authentication mechanisms have been investigated in these years. However, once the scale of the networks

---

The author gratefully acknowledges the support from Taiwan Information Security Center (TWISC) and Ministry of Science and Technology, Taiwan, under the Grants Numbers MOST 103-2221-E-259-016-MY2 and MOST 103-2221-E-011-090-MY2.

K.-H. Yeh (✉)
Department of Information Management, National Dong Hwa University, Hualien 974, Taiwan
e-mail: khyeh@mail.ndhu.edu.tw

becomes larger, the authentication scheme which supports the circumstance of single-server architecture does not suffice for users' need anymore. This may limit the future development and pervasive usage of existing Internet based applications. For example, to pursue the reliability and efficiency in a resource acquiring process, the remote service system often consists of many servers located at different places. The single-server based authentication protocols will be in-efficient on multi-server communication architecture. In addition, a legal user, who intends to access distinct network services, must register with the services providers (or servers) in advance and memorize all corresponding identities and passwords. This inconvenience will impede the pervasive usage of such multi-server based application systems. Hence, providing a secure and efficient authentication mechanism compatible to multi-server architecture will be crucial for future service systems.

Due to the difficult tradeoff between security robustness and computation complexity, it is a particular challenge to design an authentication scheme which simultaneously possesses system reliability and performance efficiency. The research community has promptly focused on this research area in these years. In 2004, Juang [9] developed a key agreement based authentication protocol which allows legal remote user to register only once and then access network services from distinct servers efficiently. Later, Chang and Lee [4] presented an improved version of Juang's protocol to pursue better system efficiency without losing any security robustness. Next year, Ku et al. [10] showed that Juang's protocol cannot withstand insider attack and provide forward secrecy. Later, Liao and Wang [13] proposed a dynamic ID based remote user authentication scheme. However, Hsiang and Shih [8] demonstrated that Liao–Wang's scheme is insecure against insider attack, impersonation attack, server spoofing attack and cannot provide mutual authentication. Later, Sood et al. [15] pointed out that Hsiang–Shih's scheme cannot resist to replay attack, impersonation attack and stolen smart card attack. In addition, the authors proposed a security enhanced scheme. Nevertheless, this scheme is vulnerable to stolen smart card attack and leak of verifier attack [6,12]. In 2012, Wang and Ma [17] presented a smart card based authentication scheme for multi-server architecture. The authors claimed that their scheme is able to resist replay attack, offline dictionary attack, server spoofing attack and impersonation attack. Unfortunately, the proposed scheme cannot withstand server spoofing attack, impersonation attack, privileged insider attack and off-line password guessing attack [7]. At the same year, Tsai et al. [16] introduced a multi-server based authentication scheme to withstand password guessing attack. The authors claimed that the proposed scheme can resist to undetectable on-line password guessing attack. However, the undetectable on-line password guessing attack is a natural weakness in password based authentication scheme [18]. Recently, Pippal et al. [14] proposed a smart card based authentication scheme for multi-server architecture. The authors claimed that their scheme can withstand various attacks such as user impersonation attack, server spoofing attack, replay attack, reflection and parallel session attacks, password guessing attack, insider attack, smart card loss attack, stolen verifier attack and known session key attack. Nevertheless, we find that Pippal et al.'s scheme is vulnerable to server counterfeit attack, user impersonation attack and man-in-the-middle attack. All of these weaknesses will be presented in the following sections.

2 Review of PTJ Scheme

In this section, we review the authentication process of Pippal et al.'s scheme [14].

2.1 Initialization Phase

The registration center RC selects two 1,024-bits prime numbers p and q and a generator $g \in Z_N^*$ and computes $N = p \times q$, where $Z_N^* = (g|1 \leq g \leq N-1, gcd(g, N) = 1)$. Next, RC generates k random numbers $(r_1, r_2, \ldots, r_k)$ for k servers, respectively. Note that $gcd(r_i, r_j) = 1, gcd(r_i, \emptyset(N)) = 1$, where $1 \leq i, j \leq k, i \neq j$. After that, RC computes secret key $S_j = g^{\prod_{i=1, i\neq j}^{K} r_j} mod\ N$ and $t = \frac{1}{g^{\prod_{i=1}^{k} r_i} mod\ N}$ for every server S_j.

2.2 Server Registration Phase

In this phase, the server S_j submits SID_j to RC over a secure channel. Once receiving the registration request from S_j, RC assigns r_j to S_j and sends $\{r_j, t, g, N, h(.)\}$ to S_j via a secure channel.

2.3 User Registration Phase

In this phase, the user U_i submits $\{UID_i, PW_i\}$ to RC which then computes $P = h(UID_i \| PW_i \| t)$ and issues a smart card to U_i. Note that $\{(s_1, s_2, \ldots, s_k), t, g, N, P, h(.)\}$ is stored in this smart card's memory.

2.4 Login and Authentication Phase (Fig. 1)

When U_i wants to access S_j, U_i inserts his/her smart card into the card reader and inputs his/her identity UID_i' and password PW_i'. The smart card then calculates $P' = h(UID_i' \| PW_i' \| t)$ and checks whether P' equals to stored P or not. If it holds, U_i is authenticated. Next, the smart card generates a random nonce a, computes, $A = g^a mod\ N, M_1 = (s_j^{UID_i \times SID_j} \times A)\, mod\ N$, and sends $\{UID_i, M_1\}$ to S_j. Upon receiving $\{UID_i, M_1\}$, S_j

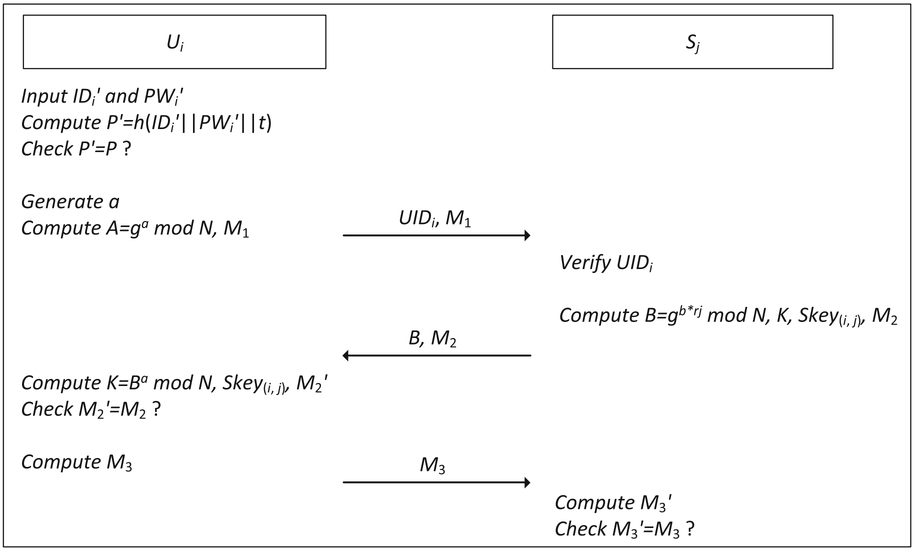

Fig. 1 PTJ scheme

verifies UID_i. If it is valid, S_j generates a random nonce b, performs the computations of B, K, $SKey_{(i,j)}$, and M_2. After that, S_j sends the response $\{B, M_2\}$ to U_i.

$$\begin{aligned} B &= g^{b\times r_j} mod\ N; \\ K &= \left(\left(M_1^{(r_j)} \times t^{(UID_i \times SID_j)}\right) mod\ N\right)^b = g^{(a\times b\times r_j)} mod\ N; \\ SKey_{(i,j)} &= h(K \| UID_i \| SID_j); \\ M_2 &= h\left(K \| UID_i \| SID_j \| B \| SKey_{(i,j)}\right). \end{aligned}$$

Once U_i receives $\{B, M_2\}$, U_i first performs the following computations and then checks whether computed $M_2{}'$ equals to received M_2 or not. If it holds, S_j is authenticated.

$$\begin{aligned} K &= (B)^a\ mod\ N = \left(g^{b\times r_j}\right)^a mod\ N = g^{(a\times b\times r_j)} mod\ N; \\ SKey_{(i,j)} &= h(K \| UID_i \| SID_j); \\ M_2' &= h\left(K \| UID_i \| SID_j \| B \| SKey_{(i,j)}\right) \end{aligned}$$

Subsequently, U_i computes M_3 and sends it to S_j. Finally, S_j calculates M_3' and checks whether computed M_3' is equal to received M_3 or not. If it holds, mutal authentication is achieved. Both U_i and S_j agree upon a common session key $SKey_{(i,j)}$.

$$\begin{aligned} M_3 &= h\left(K \| UID_i \| SID_j \| A \| B \| SKey_{(i,j)}\right); \\ M_3' &= h\left(K \| UID_i \| SID_j \| A \| B \| SKey_{(i,j)}\right) \end{aligned}$$

2.5 Password Change Phase

When U_i wants to change the password. U_i inserts the smart card into the card reader and keys in UID_i' and PW_i'. The smart card computes $P' = h(UID_i' \| PW_i' \| t)$ and checks whether P' equals to stored P or not. If it holds, U_i is legitimate. After that, U_i is allowed to enter a new password PW_{inew}, and the card reader computes $P_{new} = h(UID_i \| PW_{\text{inew}} \| t)$ and stores P_{new} in the smart card's memory.

3 Vulnerabilities of PTJ Scheme

3.1 Server Counterfeit Attack

Suppose there exists a legal but malicious user U_k possessing a smart card with $\{(s_1, s_2, \ldots, s_k), t, g, N, P_k, h(.)\}$, where $P_k = h(UID_k \| PW_k \| t)$. Once U_k intends to launch a server counterfeit attack, U_k can perform the following steps to cheat U_i that he/she is S_j.

Step 1: During a normal authentication session between U_i and S_j, U_k interrupts $\{UID_i, M_1\}$, where $M_1 = (s_j^{UID_i \times SID_j} \times A) mod\ N$ and $A = g^a mod\ N$.

Step 2: U_k computes the following values.

1. $B = g^b mod\ N$ with a random nonce b.
2. $K = \left(\left(M_1 \times \frac{1}{s_j}^{(UID_i \times SID_j)}\right) mod\ N\right)^b = g^{(a\times b)} mod\ N$.
3. $SKey_{(i,j)} = h(K \| UID_i \| SID_j)$. Note that SID_j is a public value.
4. $M_2 = h\left(K \| UID_i \| SID_j \| B \| SKey_{(i,j)}\right)$

After that, U_k pretends that he/she is S_j and sends the response $\{B, M_2\}$ to U_i.

Step 3: With $\{B, M_2\}$, U_i performs the following verification.

1. $K = (B)^a \, mod \; N = \left(g^b\right)^a mod \; N = g^{(a \times b)} mod \; N$
2. $SKey_{(i,j)} = h\left(K \| UID_i \| SID_j\right)$
3. $M_2' = h\left(K \| UID_i \| SID_j \| B \| SKey_{(i,j)}\right)$
4. Check $M_2' = M_2$?

It is obvious that this verification will be passed. Next, U_i sends M_3 to S_j.

Step 4: U_k interrupts M_3. So far, U_i misunderstands that he/she is communicating with S_j (actually it is U_k). In addition, U_i believes that he/she and S_j share a session key $SKey_{(i,j)} = h\left(K \| UID_i \| SID_j\right)$, where $K = g^{(a \times b)} mod \; N$. However, this session key is shared between U_i and U_k. Hence, we conclude that the server counterfeit attack can successfully be launched on PTJ scheme.

3.2 User Impersonation Attack

Suppose there exists a legal but malicious user U_k possessing a smart card with $\{(s_1, s_2, \ldots, s_k), t, g, N, P_k, h(.)\}$, where $P_k = h(UID_k \| PW_k \| t)$. It is obvious that U_k can easily cheat S_j that he/she is U_i with eavesdropped UID_i. This is because U_k possesses all the parameters $\{(s_1, s_2, \ldots, s_k), t, g, N, P_k, h(.)\}$. With the eavesdropped UID_i, U_k has the ability to create any legal message involved with U_i. Note that as UID_i is transmitted in public, UID_i is easily to obtain. In more details, U_k can choose a random nonce a, and compute $A = g^a mod \; N$, $M_1 = (s_j^{UID_i \times SID_j} \times A) mod \; N$, and impersonates U_i to send $\{UID_i, M_1\}$ to S_j. This cheating can easily be achieved as $\{(s_1, s_2, \ldots, s_k), t, g, N, P, h(.)\}$ is also stored in the memory of U_k's smart card. Hence, we can conclude that the user impersonation attack cannot be avoided in PTJ scheme.

3.3 Man-in-the-Middle Attack

Suppose there exists a legal but malicious user U_k possessing a smart card with $\{(s_1, s_2, \ldots, s_k), t, g, N, P_k, h(.)\}$, where $P_k = h(UID_k \| PW_k \| t)$. Now we utilize the following steps to demonstrate a man-in-the-middle attack. That is, U_k can exploit its man-in-the-middle status to cheat U_i and S_j at the same time (Fig. 2).

Step 1: The smart card at U_i side generates a random nonce a, computes $A = g^a mod \; N$, $M_1 = (s_j^{UID_i \times SID_j} \times A) mod \; N$, and sends $\{UID_i, M_1\}$ to S_j.

Step 2: U_k interrupts $\{UID_i, M_1\}$, and generates a random nonce a', computes $A' = g^{a'} mod \; N$, $M_1' = (s_j^{UID_i \times SID_j} \times A') mod \; N$, and sends $\{UID_i, M_1'\}$ to S_j.

Step 3: S_j verifies UID_i, and generates a random nonce b, computates values B, K, $SKey_{(i,j)}$, and M_2. Next, S_j sends the response $\{B, M_2\}$ back to U_i.

$$
\begin{aligned}
B &= g^{b \times r_j} mod \; N; \\
K &= \left(\left(M_1'^{(r_j)} \times t^{(UID_i \times SID_j)}\right) mod \; N\right)^b = g^{(a' \times b \times r_j)} mod \; N; \\
SKey_{(i,j)} &= h(K \| UID_i \| SID_j); \\
M_2 &= h\left(K \| UID_i \| SID_j \| B \| SKey_{(i,j)}\right).
\end{aligned}
$$

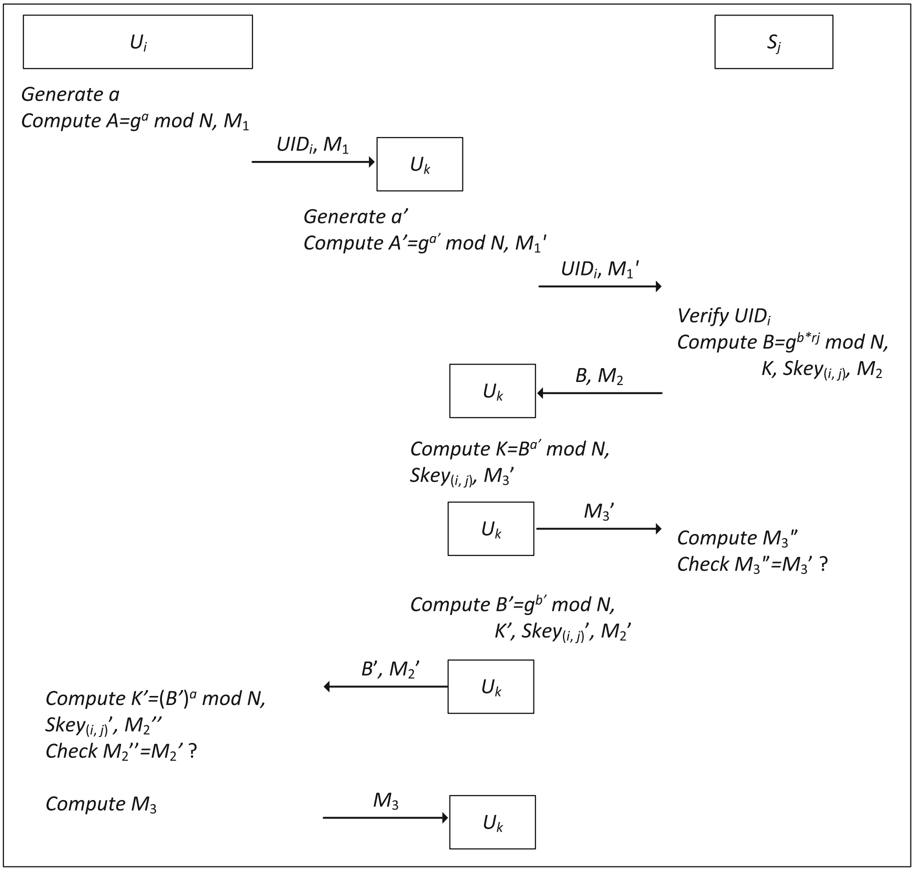

Fig. 2 Man-in-the-middle attack on PTJ scheme

Step 4: U_k interrupts $\{B, M_2\}$, and computes values K, $SKey_{(i,j)}$, and M_3'. Then, U_k sends M_3' to S_j. Obviously, the verification of M_3' will be passed at the S_j side.

$$K = (B)^{a'} \, mod \; N = \left(g^{b\times r_j}\right)^{a'} mod \; N = g^{(a'\times b\times r_j)} mod \; N;$$
$$SKey_{(i,j)} = h(K \| UID_i \| SID_j);$$
$$M_3' = h\left(K \| UID_i \| SID_j \| A \| B \| SKey_{(i,j)}\right)$$

Now, U_k and S_j share a session key $SKey_{(i,j)} = h(K \| UID_i \| SID_j)$, where $K = g^{(a'\times b\times r_j)} mod \; N$.

Step 5: U_k computes values B', K', $SKey_{(i,j)}{}'$, and M_2'. After that, U_k sends $\{B', M_2'\}$ to U_i.

$$B' = g^{b'} mod \; N \text{ with a random nonce } b';$$
$$K' = \left(\left(M_1 \times \frac{1}{s_j}^{(UID_i\times SID_j)}\right) mod \; N\right)^{b'} = g^{(a\times b')} mod \; N;$$
$$SKey_{(i,j)}{}' = h(K' \| UID_i \| SID_j);$$

$$M_2' = h\left(K' \| UID_i \| SID_j \| B' \| SKey_{(i,j)}{}'\right).$$

Step 6: With $\{B', M_2'\}$, U_i performs the following verification.

- $K' = \left(B'\right)^a \, mod \; N = \left(g^{b'}\right)^a \, mod \; N = g^{(a \times b')} mod \; N$
- $SKey_{(i,j)}{}' = h(K' \| UID_i \| SID_j)$
- $M_2'' = h\left(K' \| UID_i \| SID_j \| B' \| SKey_{(i,j)}{}'\right)$
- Check $M_2'' = M_2'$?

It is obvious that all the verifications will be passed. U_i then computes M_3 and sends it back to S_j.

$$M_3 = h\left(K' \| UID_i \| SID_j \| A \| B' \| SKey_{(i,j)}{}'\right)$$

Step 7: U_k interrupts M_3. Now U_k and U_i share a session key $SKey'_{(i,j)} = h(K' \| UID_i \| SID_j)$, where $K' = g^{(a \times b')} mod \; N$. Since U_k shares two different session keys with U_i and S_j, respectively, the man-in-the-middle attack is successfully performed.

4 The Proposed Scheme

In this section, we propose a novel protocol for multi-server architecture, where a trusted registration center RC exists. First, RC chooses two 1024-bits prime numbers p and q and a generator $g \in Z_N^*$ and computes $N = p \times q$, where $Z_N^* = (g | 1 \le g \le N-1, gcd\,(g, N) = 1)$. Next, RC generates k random numbers $(r_1, r_2, \ldots, r_k)$ for k servers, respectively. Note that $gcd\left(r_i, r_j\right) = 1$, $gcd\,(r_i, \phi(N)) = 1$, where $1 \le i, j \le k, i \ne j$.

4.1 Server Registration Phase

In this phase, the server S_j submits SID_j to RC over a secure channel. Once RC receives the request from S_j, RC assigns r_j to S_j and computes $h(r_j \| y_{RC})$, where y_{RC} is a secret value chosen by RC. Next, RC sends $\{r_j, t, h(r_j \| y_{RC}), g, N, h(.)\}$ to S_j via a secure channel. Note that $t = \frac{1}{g^{\prod_{i=1}^{k} r_i} mod \; N}$.

4.2 User Registration Phase

In this phase, the user U_i submits $\{UID_i, PW_i\}$ to RC which then generates a random number r_i and computes $P = h(UID_i \| PW_i \| r_i)$. Next, RC issues a smart card storing parameters $\{(s_{1\_i}, s_{2\_i}, \ldots, s_{k\_i}), r_i, g, N, P, h(.)\}$ to U_i. Note that $s_{j\_i} = g^{h(SID_j \| UID_i \| h(r_j \| y_{RC})) \times \prod_{i=1, i \ne j}^{k} r_j} mod \; N$, where y_{RC} is a secret value chosen by RC.

4.3 Login and Authentication Phase (Fig. 3)

When U_i wants to login S_j, U_i inserts his/her smart card into the card reader and inputs his/her identity UID_i' and password PW_i'. The smart card then calculates $P' = h(UID_i' \| PW_i' \| r_i)$ and checks whether P' equals to Por not. If it holds, U_i is legal. Next, the smart card generates a random nonce a, computes $M_1 = (s_{j\_i} \times g^a) mod \; N$, and sends $\{UID_i, M_1\}$ to S_j. Once S_j receives $\{UID_i, M_1\}$, S_j verifies UID_i. If UID_i is valid, S_j generates a random nonce b, and computes B, K, $SKey_{(i,j)}$ and M_2. After that, S_j sends $\{B, M_2\}$ to U_i.

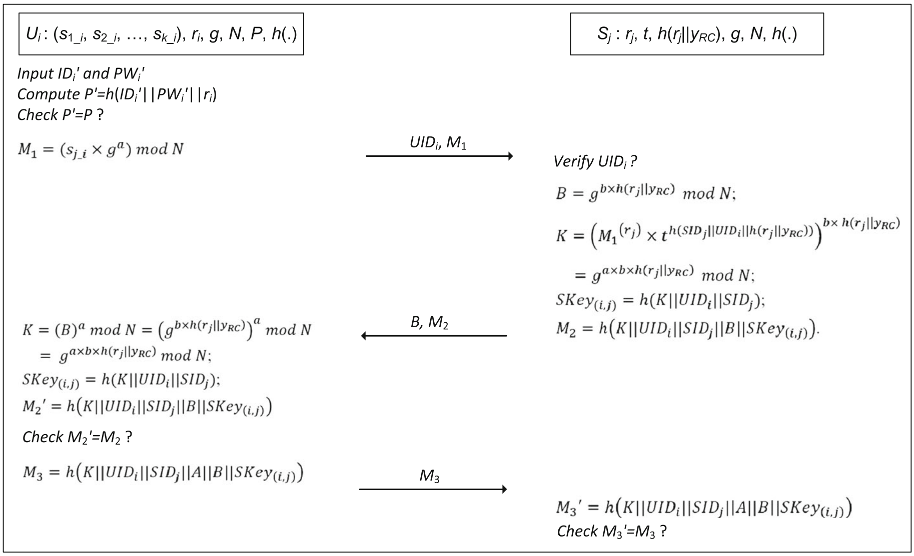

Fig. 3 Login and authentication phase of the proposed scheme

$$B = g^{b\times h(r_j\|y_{RC})} mod\ N;$$
$$K = \left(M_1^{(r_j)} \times t^{h(SID_j\|UID_i\|h(r_j\|y_{RC}))}\right)^{b\times h(r_j\|y_{RC})} = g^{a\times b\times h(r_j\|y_{RC})} mod\ N;$$
$$SKey_{(i,j)} = h(K\|UID_i\|SID_j);$$
$$M_2 = h\left(K\|UID_i\|SID_j\|B\|SKey_{(i,j)}\right).$$

Once receiving $\{B, M_2\}$, U_i calculates K, $SKey_{(i,j)}$ and M_2', and checks whether M_2' equals to M_2 or not. If it holds, S_j is authenticated.

$$K = (B)^a\ mod\ N = \left(g^{b\times h(r_j\|y_{RC})}\right)^a mod\ N = g^{a\times b\times h(r_j\|y_{RC})} mod\ N;$$
$$SKey_{(i,j)} = h(K\|UID_i\|SID_j);$$
$$M_2' = h\left(K\|UID_i\|SID_j\|B\|SKey_{(i,j)}\right)$$

After that, U_i computes M_3 and sends M_3 to S_j. Upon getting M_3, S_j calculates M_3' and checks whether M_3' is equal to M_3 or not. If it holds, mutal authentication is achieved. Both U_i andS_j agree upon a common session key $SKey_{(i,j)}$.

$$M_3 = h\left(K\|UID_i\|SID_j\|A\|B\|SKey_{(i,j)}\right);$$
$$M_3' = h\left(K\|UID_i\|SID_j\|A\|B\|SKey_{(i,j)}\right)$$

4.4 Password Change Phase

When U_i wants to change the password. U_i inserts the smart card into the card reader and keys in UID_i' and PW_i'. The smart card computes $P' = h(UID_i'\|PW_i'\|r_i)$ and checks whether P' equals to stored P or not. It holds, U_i is legitimate. After that, U_i is allowed to enter a new password PW_{inew}, and the card reader generates a new random number r_i' and computes $P_{new} = h(UID_i\|PW_{\text{inew}}\|r_i')$ and stores P_{new} and r_i' in the smart card's memory.

5 Security Analysis

In this section, we present the formal analysis of our proposed authentication scheme based on [1–3,5].

5.1 Communication Model

In the communication model, we assume that a user U_i, intends to establish a session key $SKey_{(i,j)}$ with a service provider S_j. Therefore, some definitions must be presented.

- Protocol Participants: there exists a non-empty set of users, called *Client*, and a non-empty set of service providers, i.e.*Server*, in the protocol P in which the participant is either a user or a service provider. Each participant may possess several instances, called *oracles*, which are involved in distinctly concurrent executions of P. Here, Π^i_U is denoted as the instance i of a participant U.
- Long-term Secret Keys: For each $S_j \in Server$, it owns a secret value r_j as its long-term secret key, and y_{RC} is *RC*'s long-term secret key trusted by all $U_i \in Client$ and all $S_j \in Server$.
- Accepting and Terminating: There exists two states, ACC_Π^i_U and TERM_Π^i_U, for oracle Π^i_U. ACC_Π^i_U is set to *true* when Π^i_U is able to compute a valid session key, while TERM_Π^i_U will be set to *true* when Π^i_U sends (or receives) the last message of the protocol, receives an unexpected message, or misses an expected message.
- Session and Partner Identities: the session identity (*sid*) is utilized for representating each unique session. We define *sid* for oracles $\Pi^i_{U_A}$ and $\Pi^i_{U_B}$ in the execution of a protocol as $sid\_\Pi^i_{U_A} = sid\_\Pi^i_{U_B}$=$\{Flows_{U_A U_B}$|all flows that $\Pi^i_{U_A}$ exchanges with $\Pi^i_{U_B}$ in the execution of a protocol$\}$. In addition, $pid\_\Pi^i_{U_A} = U_B$ is defined as that the oracle $\Pi^i_{U_A}$believes that it has just exchanged a session key with an oracle of participant U_B.

5.2 Adversary Model

In this paper, we assume that the adversary is able to interact with the participants via oracles queries. The following major queries model the capabilities of the adversary.

- Send(Π^i_U, m): This query sends a message m to an oracle Π^i_U, and gets the corresponding results.
- Reveal(Π^i_U): This query returns the session key of the oracle Π^i_U.
- Corrupt(U): This query returns the long-term secret key of U.
- Execute($\Pi^i_{U_A}, \Pi^i_{U_B}$): This query models passive attacks in which the adversary can obtain the messages exchanged during the honest execution of the protocol between two oracles $\Pi^i_{U_A}$ and $\Pi^i_{U_B}$.
- Hash(m): The one-way hash function can be viewed as random functions with the appropriate range in the ideal hash model. Note that, if m has never been queried before, it returns a truly random number r to the adversary and stores (r, m) in the hash table. Otherwise, it returns the previously generated result to the adversary.
- Test(Π^i_U): This query models the security of the session key, i.e., whether the real session key can be distinguished from a random string or not. For answering this question, a unbiased coin b is flipped by the oracle Π^i_U. When the adversary issues a single Test query to Π^i_U, the adversary either obtains the real session key $SKey_{(i,j)}$ if b= 1 or a random string if b= 0.

5.3 Security Properties

This subsection describes the security required in the proposed authentication.

- Freshness: An oracle Π_U^i is fresh if the following conditions hold.
  1. ACC_Π_U^i is set to *true*.
  2. No Corrupt query has been issued by the adversary before ACC_Π_U^i is set to *true*.
  3. Neither Π_U^i nor its partner has been issued a Reveal query.

  In general, a session key is fresh if, and only if, all oracles participated in current session were fresh.
- Partnering: In the protocol P, two oracles $\Pi_{U_i}^i$ and $\Pi_{S_j}^j$ are partnered if the following conditions hold.
  1. Both ACC_$\Pi_{U_i}^i$ and ACC_$\Pi_{S_j}^j$ have been set to *true*.
  2. A session key $SKey_{(i,j)}$ has been agreed by $\Pi_{U_i}^i$ and $\Pi_{S_j}^j$.
  3. $sid\_\Pi_{U_i}^i = sid\_\Pi_{U_j}^i$.
  4. $pid\_\Pi_{U_i}^i = S_j$.
  5. $pid\_\Pi_{S_j}^i = U_i$.
- AKE Security (Session Key Security): The adversary tries to guess the hidden bit b involved in a Test query via a guess b'. We say that the adversary wins the game of breaking session key security of an AKE (Authenticated Key Exchange) protocol P if the adversary issues Test queries to a fresh oracle Π_U^i and guesses the hidden bit b successfully. The probability that the adversary wins the game is $\Pr[b' = b]$. In brief, the advantage of an adversary A in attacking protocol P can be defined as $Adv_P^{AKE}(A) = |2 \times \Pr\left[b' = b\right] - 1|$. In brief, P is AKE-secure if $Adv_P^{AKE}(A)$ is negligible.

5.4 Formal Security Analysis

In this subsection, we formally analyze the security of our proposed authentication protocol. Notations and definition will be presented first, and the formal security analysis is then demonstrated. We define T_A be the adversary's total running time, and q_s, q_r, q_c, q_e and q_h are the number of Send, Reveal, Corrupt, Execute, and Hash queries, respectively.

Definition 1 (*Computational Diffie-Hellman (CDH) Assumption*) Let $G = \langle g \rangle$ be a multiplicative cyclic group of order N, and two random numbers t and k, are chosen in Z_N^*. Given g, g^t, and g^k, the adversary A has a negligible success probability $Succ_G^{CDH}(A)$ for obtaining an element $z \in G$, such that $z = g^{tk}$ within polynomial time.

Theorem 1 *Let A be an adversary against the AKE security of our proposed authentication protocol within a time bound T_A, with less than q_s Send queries with the communication entities, and asking q_h times Hash queries. Then, $Adv_P^{AKE}(A, q_s, q_h) \le q_h q_s \times Succ_G^{CDH}\left(T'_A\right)$, where T'_A denotes the computational time for $Succ_G^{CDH}$ and $q_s = \sum_{i=1}^{4} q_{s\_i}$ is the sum of number of $Send_1$, $Send_2$, $Send_3$, and $Send_4$.*

Proof Let A be an adversary which is able to get an advantage ε to break an AKE-secure protocol within time T_A. We can construct a CDH attacker B from A to respond all of A's queries and deal with the CDH problem, where B is given a challenge $\Omega = (g^t,\ g^k)$ and outputs an element z such that $z = g^{tk}$.

First, when A issues Send$_1$ query as a *start* command, B responds $\{UID_i,\ M_1\}$ to A. Second, when A issues Send$_2$ query, B randomly chooses two integers c_1 and c_2 from $[1,\ q_{s\_2}]$. If $c_1 \neq c_2$, B responds $\{B,\ M_2\}$ to A. Otherwise, B replaces the corresponding parameters of $\{B, M_2\}$ with the element g^k from Ω to generate a new and random message $\{B, M_2\}'$, and then responds the message $\{B, M_2\}'$ to A. Third, once B receives the Send$_3$ query from A, B answers the message $\{M_3\}$ as the protocol. If the input of the query is from Ω, B generates a new message $\{M_3\}'$, and then responds $\{M_3\}'$ to A. Fourth, when A issues the Send$_4$ query, B answers a null string and then sets ACC_$\Pi^i_{U_i}$, ACC_$\Pi^j_{S_j}$, TERM_$\Pi^i_{U_i}$ and TERM_$\Pi^j_{S_j}$ to *true*.

On the other hand, when A issues a Reveal($\Pi^i_{U_i}$) or a Reveal($\Pi^j_{S_j}$) query, B checks whether the oracle has been accepted and is fresh or not. If the result is true, B answers the session key $SKey_{(i,j)}$ to A. Otherwise, if the session key has been constructed from the challenge Ω, B terminates. When A issues Corrupt(U_i), Corrupt(S_j), Execute($\Pi^i_{U_i}$, $\Pi^j_{S_j}$), Hash(m) queries, B answers in a straightforward way. When A issues a Test query, B answers in a straightforward way. Otherwise, if the session key has been constructed from the challenge Ω, B answers A with a random string with the length of the session key $SKey_{(i,j)}$.

The above simulation is indistinguishable from any execution of the proposed protocol P except for one execution which the challenge Ω is involved with. The probability γ that $\boldsymbol{B}$ correctly guesses the session key that $\boldsymbol{A}$ will make a Test query on is equal to the probability of $c_1 = c_2$. Hence, we have

$$\gamma = \frac{1}{q_{s\_2}} \geq \frac{1}{q_s}$$

Assume that $\boldsymbol{A}$ issues a Test query to output b', where $b' = b$. This means that $\boldsymbol{A}$ knows the session key, so there must be at least one Hash query that returns the session key. The probability λ that $\boldsymbol{B}$ will choose the hash query correctly is

$$\lambda \geq \frac{1}{q_h}$$

The successful probability $Succ^{CDH}_G(B)$ that $\boldsymbol{B}$ will expose g^{kt} from the challenge Ω is thus

$$Succ^{CDH}_G(B) = \varepsilon \times \gamma \times \lambda \geq \varepsilon \times \frac{1}{q_s} \times \frac{1}{q_h}$$

Finally, the advantage of A to break the AKE-security of the protocol P is derived as follows.

$$\varepsilon = \mathrm{Adv}^{AKE}_P(A, q_s, q_h) \leq q_h q_s \times Succ^{CDH}_G\left(T'_A\right)$$

□

6 Security and Performance Comparison

To further investigate the advantage of our proposed authentication protocol, we compare the proposed scheme with five relevant multi-server authentication schemes [11,12,14,15,17] in terms of major security features. From the viewpoint of robustness (Table 1), our proposed authentication scheme is superior to all of these five protocols by supporting all the major

Table 1 Security comparison among our proposed protocol and other schemes

	The proposed scheme	PTJ Scheme [14]	WM scheme [17]	LXMW scheme [12]	SSS scheme [15]	LLC Scheme [11]
No verification table	Yes	Yes	No	Yes	No	Yes
Freely to choose and change password	Yes	Yes	Yes	Yes	Yes	Yes
No involvement of *RC* during password change phase	Yes	Yes	No	Yes	Yes	No
Provide mutual authentication without the support of *RC*	Yes	Yes	Yes	No	No	Yes
Resistance to known key attack	Yes	Yes	Yes	Yes	Yes	Yes
Resistance to user impersonation attack	Yes	No	No	Yes	Yes	No
Resistance to server counterfeit attack	Yes	No	No	Yes	Yes	No
Resistance to man-in-the middle attack	Yes	No	No	Yes	Yes	No
Resistance to replay attack	Yes	Yes	Yes	Yes	Yes	Yes
Resistance to parallel session attack	Yes	Yes	Yes	Yes	Yes	Yes
Provide validity proof	Yes	Yes	Yes	No	No	No

security features. In particular, our protocol inherits the merit from PTJ scheme, i.e. providing mutual authentication without the support of *RC*. This design significantly improves the efficiency of the protocol round. In brief, it is clearly seen that our proposed authentication scheme keeps all the advantages and achieves the security requirements.

Performance evaluation is an important issue while designing a robust and efficient authentication schemes. This evaluation reflects the practicability of implementing the proposed authentication protocol in the real world. As study [14] had demonstrated the computation efficiency of their proposed authentication scheme is better than other related studies [11,12,15,17]. Hence, here we only compare our proposed protocol with two most-relevant proposals, i.e. PTJ scheme [14] and WM scheme [17] in terms of protocol efficiency. The performance comparison among our proposed scheme and other two schemes has been listed in Table 2. The metrics are Hash Function (HF), Modular Multiplication (MM), Modular Exponentiation (ME), ECC Point Multiplication (PM) and Encryption/Decryption (E/D). Although our proposed scheme requires extra 2 one-way hash functions than PTJ scheme, our scheme is efficient. This is as the cost of performing one-way hash functions can be almost ignored in comparison with other heavy computation modules such as Modular Exponentiation, ECC Point Multiplication or E/D. Moreover, compared to PTJ scheme, as one ME is reduced, the efficiency is improved. It is obvious that our scheme can achieve the same order of computation complexity as PTJ scheme does.

7 Conclusion

To efficiently protect a multi-server based service system is a particular challenge owing to the difficult tradeoff between system security and computation efficiency. One of the most

Table 2 Performance comparison among our proposed protocol and other schemes

	Type of operations	The proposed scheme	PTJ scheme [14]	WM scheme [17]
Registration Phase	HF	2	1	2
	MM	0	0	0
	ME	0	0	0
	PM	0	0	2
	E/D	0	0	0
Login and Authentication Phase	HF	8	7	11
	MM	2	2	0
	ME	6	7	0
	PM	0	0	4
	E/D	0	0	0
Password Phase	HF	2	2	12
	MM	0	0	0
	ME	0	0	0
	PM	0	0	12
	E/D	0	0	4
Total number of operations		12 HF+	10 HF+	25 HF+
		2 MM+	2 MM+	0MM+
		6 ME+	7 ME+	0 ME+
		0 PM+	0 PM+	18 PM+
		0 E/D	0 E/D	4 E/D

promising directions is to implement an efficient authentication mechanism for multi-server architecture. A recent study proposed by Pippal et al. is one of the pioneers on this interesting research area. However, it has space for improvement. In this paper, we have demonstrated that Pippal et al.'s multi-server based authentication scheme fails to provide adequate security and is subject to user impersonation attack, server counterfeit attack, and man-in-the-middle attack. A novel authentication protocol is thus introduced for security enhancement. With the formal analysis and performance comparison, the security robustness and computation efficiency of our proposed protocol can be guaranteed. Therefore, we believe that our proposed authentication protocol is practical and secure for multi-server communication environment.

References

1. Bellare, M., Pointcheval, D., & Rogaway, P. (2000). Authenticated key exchange secure against dictionary attacks. In *Proceedings of EUROCRYPT* (Vol. 1807, pp. 140–156). LNCS 2000.
2. Bellare, M., & Rogaway, P. (1993). Entity authentication and key distribution. In *Proceedings of CRYPTO* (Vol. 773, pp. 232–249) LNCS.
3. Blake-Wilson, S., Johnson, D., & Menezes, A. (1997). Key agreement protocols and their security analysis. In *Proceedings of th 6th IMA international conference on cryptography and coding* (Vol. 1355, pp. 30–45). LNCS.
4. Chang, C. C., & Lee, J. S. (2004). An efficient and secure multi-server password authentication scheme using smart card. In *Proceedings of international conference on cyberworlds* (pp. 417–422).
5. Chang, C. C., & Lee, C. Y. (2012). A secure single sign-on mechanism for distributed computer networks. *IEEE Transactions on Industrial Electronics*, *59*(1), 629–637.

6. Chen, B. L., Kuo, W. C., & Wu, L. C. (2012). Cryptanalysis of Sood et al.'s dynamic identity based authentication protocol for multi-server architecture. *International Journal of Digital Content Technology and its Applications (JDCTA)*, *6*(4), 180–187.
7. He, D., & Wu, S. (2012). Security flaws in a smart card based authentication scheme for multi-server environment. *Wireless Personal Communications*,. doi:10.1007/s11277-012-0696-1.
8. Hsiang, C., & Shih, W. K. (2009). Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment. *Computer Standards & Interfaces*, *31*(6), 1118–1123.
9. Juang, W. S. (2004). Efficient multi-server password authenticated key agreement using smart cards. *IEEE Transaction on Consumer Electronics*, *50*(1), 251–255.
10. Ku, W. C., Chuang, H. M., Chiang, M. H., & Chang, K. T. (2005). Weaknesses of a multi-server password authenticated key agreement scheme. In *Proceedings of 2005 national computer symposium* (pp. 1–5).
11. Lee, C. C., Lin, T. H., & Chang, R. X. (2011). A secure dynamic ID based remote user authentication scheme for multi-server environment using smart cards. *Expert Systems with Applications*, *38*(11), 13863–13870.
12. Li, X., Xiong, Y., Ma, J., & Wang, W. (2012). An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards. *Journal of Network and Computer Applications*, *35*(2), 763–769.
13. Liao, Y. P., & Wang, S. S. (2009). A secure dynamic ID based remote user authentication scheme for multi-server environment. *Computer Standards & Interfaces*, *31*(1), 24–29.
14. Pippal, R. S., Jaidhar, C. D., & Tapaswi, S. (2013). Robust smart card authentication scheme for multi-server architecture. *Wireless Personal Communications*. doi:10.1007/s11277-013-1039-6.
15. Sood, S. K., Sarje, A. K., & Singh, K. (2011). A secure dynamic identity based authentication protocol for multi-server architecture. *Journal of Network and Computer Applications*, *34*(2), 609–618.
16. Tsai, J.-L., Lo, N.-W., & Wu, T.-C. (2012). A new password-based multi-server authentication scheme robust to password guessing attacks. *Wireless Personal Communications*. doi:10.1007/s11277-012-0918-6.
17. Wang, B., & Ma, M. (2012). A smart card based efficient and secured multi-server authentication scheme. *Wireless Personal Communications*. doi:10.1007/s11277-011-0456-7.
18. Yeh, K.-H., Lo, N. W., Hsiang, T.-R., Wei, Y.-C., & Hsieh, H.-Y. (2013). Chaos between password-based authentication protocol and dictionary attacks. *Advanced Science Letters*, *19*(3), 1048–1051(4).

Kuo-Hui Yeh received his B.S. degree in Mathematics from the Fu Jen Catholic University, Taipei County, Taiwan, in 2000, and the M.S. and Ph.D. degrees in Information Management from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2005 and 2010, respectively. He is currently an assistant professor of Department of Information Management at the National Dong Hwa University, Hualien, Taiwan. His research interests include cloud computing, RFID applications and security, wireless network protocol, and anonymous authentication.