Novel Attacks and Novel Efficient Three-Party Authenticated Key Agreement Schemes for Resource-limited Devices

Hung-Yu Chien

Department of Information Management, National Chi-Nan University, Taiwan
hychien@ncnu.edu.tw

Abstract

A three-party authenticated key agreement scheme (3PAKA) is a protocol that enables a pair of two registered clients to establish session keys via the help of a trusted server such that each client only pre-shares some secret with the server. As the resource-constrained devices are becoming more and more popular and deployed, it is important to design secure 3PAKA schemes that are efficient in terms of both the communication and the computation.

Among existent 3PAKA schemes, Yang et al.'s scheme significantly reduces the devices' computational load by blinding the Diffie-Hellman values. However, we find a very powerful kind of attacks, which has never been reported the attackers only eavesdrop on the transmissions and can derive the secret keys and the session keys. We pinpoint the design pitfalls and propose our countermeasure.

Based on the Modified Computational Diffie-Hellman Problem (MCDHP), we propose a novel 3PAKA scheme that simultaneously improves the security, the communication, and the computation. The proposed scheme shows the best performance in terms of security, communications and computations, when we evaluate the related works under the same criteria. The protocol security checker Automated Validation of Internet Security Protocols and Applications (AVISPA) has verified the security properties of our scheme.

Keywords: Authentication, Key agreement, Password, Security, Random oracle

1 Introduction

To make secure pair-wise communications possible among n entities in a distributed network, a simple solution is to let every entity share a distinct key with each potential partner. However, this approach would require each entity equipped with n-1 secret keys. It turns out that this solution has poor scalability and high secret key maintenance cost, because there are potentially a large number of entities in a distributed network, and there would be new entities joining in and some entities dropping out at any time. To improve the scalability and lower the maintenance cost, the so-called three-Party Authenticated Key Agreement (3PAKA) approach is commonly adopted, where each registered client keeps only one secret with a trusted server, and any pair of the registered clients can establish authenticated session keys with the help of the server.

Owing to the hardness of the Computational Diffie-Hellman Problem (CDHP), the main stream of 3PAKA schemes such as [1-19] adopt the CDHP to design their authenticated key agreement schemes. Among the various computations involved in a 3PAKA scheme, the exponentiation is one of the most computationally expensive operations. Even though the exponentiations are affordable for many computers, the exponentiations are still very resource-depleting for many resource-constrained devices. It is, therefore, desirable to design secure 3PAKA schemes that can reduce these intensive computations.

Regarding the communication performance of 3PAKA schemes, two metrics are popular in evaluating the performance: one is the number of the communication steps and the other is the number of the message rounds. A message step denotes one transmission step by which one entity sends data to another entity, and a message round represents the integration of one or more message steps of which there is no data dependency between these steps and these steps can be executed in parallel to save communication time.

It is very challenging to design 3PAKA schemes that simultaneously achieve good performance in terms of security properties, computations, communications and user convenience. Yang et al. [1] proposed a promising 3PAKA scheme which owns several advantages: (1) only three message rounds; (2) no requirement of time synchronization; and (3) the adoption of smart cards for users' convenience. The scheme depends on nonce instead of timestamp to ensure message freshness, which makes it more practical than those timestamp-based schemes (such as the scheme [18]) because timestamp synchronization is very difficult in a large distributed environment. Yang et al.'s scheme demands

*Corresponding Author: Hung-Yu Chien; E-mail: hychien@ncnu.edu.tw
DOI: 10.3966/160792642019122007015

only three rounds. To reduce the number of exponentiations, Yang et al.'s scheme, instead of sending ephemeral Diffie-Hellman (DH) public keys in the conventional 3PAKA schemes, uses random numbers to blind the secret exponents, which reduces the number of exponentiations. These merits make it more attractive than its counterparts. Unfortunately, Amin and Biswas [20] have shown several weaknesses of Yang et al.'s scheme. However, their attacks (the impersonation attacks and the password-guessing attacks) are based on a strong assumption that, once the user's card is captured, the content of the card is total compromised [20]. This assumption might hold or not, depending on whether the implementation adopts tamper-resistant cards. That is, if the card is tamper-resistant, then Amin-Biuswas's impersonation attacks and password-guessing attacks do not work.

Here, we would like to demonstrate one novel attack on such kind of scheme: we show that Yang et al.'s scheme is vulnerable to the secret key disclosure attack, the session key disclosure and the impersonation attacks, even if an attacker just eavesdrops the transmissions. The attacks leverage the design pitfalls of not-well-formed ephemeral DH public keys in Yang et al.'s scheme. The attack is much more powerful and has not been reported before. Our approach works by deriving the secret key from a set of secret-key-related values, and we call it the secret exponent derivation attack. The attackers can effectively derive the secret keys, the session keys, and impersonate the communicating parties. To conquer the security weaknesses, we discuss the design pitfall and propose our improved scheme. The proposed scheme not only conquers the security attacks but also enhance the DH key computation efficiency and the communication performance. The AVISPA security checker [25-30] has verified the security properties of our scheme.

The rest of this paper is organized as follows. The related works are discussed in Section 2. We review Yang et al.'s scheme in Section 3. The attacks are introduced and discussed in Section 4. The improved scheme is described in Section 5. The performance evaluation and the security analysis are given in Section 6. Finally, Section 7 states our conclusions and the future work.

2 Related works

Since Steiner et al. [15] proposed the first 3PAKA, many researchers such as [1-4, 6-9, 11-14, 16-20] have devoted themselves to improve the security, the computational performance or the communicational performance of 3PAKA schemes. Gong [5] studied that the lower bound of the message steps of 3PAKA schemes. Considering those 3PAKA schemes that rely on nonce instead of time-stamp and let the clients choose the keying materials, Gong claimed that the lower bound of the message steps is five for key distribution schemes and it is six steps for key confirmation schemes. Later Chien and Wu [3] corrected the lower bounds for them to be three steps and four steps respectively. Yang et al.'s scheme [1] required three message rounds in four message steps. The scheme is quite efficient.

Chien [21] proposed the Modified Computational Diffie-Hellman Problem (MCDHP) and proved the hardness of the problem; he also proposed a generic approach to apply the MCDHP problem to enhance the computational efficiency of the clients in two-party AKA schemes. Here, we will apply the MCDHP problem to conquer the weakness of Yang et al.'s not-well-formed DH public keys and enhance the computational efficiency.

Conventionally, each scan of a user's biometric information will have some variations, and it is not suitable to use it as a key in authenticating the user. Luckily, Jin et al. [22] proposed their bio-hashing that produces a user-specific code from the user's biometric features. The bio-hashing has been improved by Lumini and Nanni [23], and it has been applied in many authentication schemes such as [1, 8, 22-23].

The AVISPA is designed to formally verify cryptographic protocols, and various verification tools embedded in the AVISPA can be used to verify the goals and the specified security properties [25-30]. It has been used to formally verify several important cryptographic protocol standards and new schemes like [20, 31-33].

3 Review of Yang et al.'s Scheme

Before reviewing the scheme, we first introduce the following notations that will be used through the rest of this paper.

(G, g, p, q): p is a large prime, q is a large prime divisor of p, G is a multiplicative sub-group of Z_p^* with a prime order q, and g is a generator for G. In the following, we omit the mod p operation when the semantic is clear.

sid: the session identifier which is used to uniquely identify a session from others.

$x \leftarrow_R Z_q^*$: an integer x is randomly chosen from the set Z_q^*.

A, B: A and B denote the identities of two clients.

S, s: S denotes the server and s denotes its secret key.

$H_0 : \{0,1\}^* \times \{0,1\}^* \rightarrow Z_q^*$,

$H_1, H_2 : \{0,1\}^* \times Z_q^* \rightarrow Z_q^*$, $H_3 : Z_q^* \times Z_q^* \rightarrow Z_q^*$,

$H_4 : \{0,1\}^* \times \{0,1\}^* \times G \times G \rightarrow \{0,1\}^{l_4}$,

$H_5 : \{0,1\}^* \times \{0,1\}^* \times Z_q^* \times G \times G \rightarrow \{0,1\}^{l_5}$,

$H_6 : \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \rightarrow \{0,1\}^{l_6}$,

$h()$: $\{0,1\}^* \rightarrow \{0,1\}^{|Key|}$;

$H_{0\sim6}$ and $h()$ are eight independent cryptographic hash functions.

$h_{bio}()$, $bio_A : h_{bio}()$ is a bio-hashing function;

bio_A is the biometric information of A.

pw_A, pw_B: the passwords with which the clients A and B respectively use to protect their smart cards.

x_A, x'_A, x'_B, x'_B : the clients A and B respectively own two secret keys ($x_A = H_1(A, s), x'_A = H_2(A,s)$) and ($x_B = H_1(B,s), x'_B = H_2(B,s)$).

a, b, R_A, R_B: a and b denote ephemeral private keys chosen by A and B respectively. $R_A = g^a \bmod p$ and $R_B = g^b \bmod p$ respectively denote the corresponding ephemeral public keys.

sk_A, sk_B : the session keys computed by A and B respectively.

$pr_A, pr_B \in_R Z_q^*$: long-term private keys of the clients A and B respectively.

$Y_A = g^{pr_A}, Y_B = g^{pr_B}$: the long-term public key of A and B respectively.

Yang et al.'s scheme consists of four phases- the registration phase, the login phase, the password updating phase, and the key agreement phase. Because the password updating phase is irrelevant to our discussions, we do not review the phase here to save space.

Registration phase. This phase is executed in a secure environment. When a user (say A) wants to register in the system, he submits his identity A and his password pw_A to S; if S accepts the request, S computes $PW_A = H_0(A, pw_A)$, $x_A = H_1(A,s)$, $x'_A = H_2(A,s)$, $y_A = PW_A \oplus x_A$, $y'_A = PW_A \oplus x'_A$, and $h_A = H_3(x_A, x'_A)$. It then stores (A, h_A, y_A, y'_A) into the user's smart card, and destroys (pw_A, PW_A, x_A, x'_A).

Login phase. When the client A wants to establish a session key with the client B, A and B respectively insert their smart cards and input their passwords, and then their smart cards respectively perform the following steps. A's smart card computes $PW_A = H_0(A, pw_A)$, $x_A = y_A \oplus PW_A$, $x'_A = y'_A \oplus PW_A$, and $h_A = H_3(x_A, x'_A)$. It then verifies whether the computed h_A equals the pre-stored one in the card. If not, it stops; if the verification succeeds, then it further performs the key agreement phase. Likewise, the client B's smart card performs the operations and verifications similar to A's operations.

Key agreement phase. Suppose A and B want to establish a connection and a session key, they perform the following three rounds.

Round 1:

$A \rightarrow B$: A, $V_A, Auth_A$

A's smart card chooses a random number $a \in_R Z_q^*$, computes $R_A = g^a$, $V_A = R_A^{x_A}$ and $Auth_A = H_4(A,B,V_A,R_A)$. It then sends (A, $V_A, Auth_A$) to B.

Round 2:

$B \rightarrow S$: $A, V_A, Auth_A$, $B, V_B, Auth_B$

When B's smart card receives the request, it chooses a random number $b \in_R Z_q^*$, and computes $R_B = g^b$, $V_B = R_B^{x_B}$ and $Auth_B = H_4(A,B,V_B,R_B)$. It sends ($A, V_A, Auth_A$, B, V_B , $Auth_B$) to S.

Round 3:

$S \rightarrow A$: $S, B, w_A, R'_B, Auth'_A$

$S \rightarrow B$: $S, A, w_B, R'_A, Auth'_B$

When S receives the request from B, S computes $x_A = H_1(A,s)$, $x_B = H_1(B,s)$, $R'_A = V_A^{x_A^{-1}}$, $R'_B = V_B^{x_B^{-1}}$, $Auth_A = H_4(A,B,V_A,R'_A)$ and $Auth_B = H_4(A,B,V_B,R'_B)$. It then verifies whether both the computed $Auth_A$ and $Auth_B$ equal the received ones. If the verification succeeds, then it continues the following steps; otherwise, it rejects the request. It chooses a random number $t \in_R Z_q^*$, and computes $w_A = t \cdot H_2(A,s)^{-1} \bmod q$ $w_B = t \cdot H_2(B,s)^{-1} \bmod q$, $W_A = R'^{\,t}_A$, $W_B = R'^{\,t}_B$, $Auth'_A = H_5(S,B,w_A,R'_B,W_A)$ and $Auth'_B = H_5(S,A,w_B,R'_A,W_B)$. It sends ($S, B, w_A, R'_B, Auth'_A$) to A and (S, A, w_B, $R'_A, Auth'_B$) to B.

When the client A's smart card receives the response, it computes $W_A = g^{a \cdot w_A \cdot x'_A}$ and $Auth'_A = H_5(S,B,w_A,R'_B,W_A)$, and verifies whether the computed $Auth'_A$ equals the received one. If so, it accepts the connection and computes the session key $sk_A = H_6(A,B,R_A,R'_B,R'^{\,a}_B)$; otherwise, it rejects the connection. Similarly, client B's smart card performs its corresponding computations and verifications. If the verification succeeds, then it computes its session key $sk_B = H_6(A,B,R'_A,R_B,R'^{\,b}_A)$.

4 Exponent Derivation Attacks on Yang et al.'s Scheme

This section shows the exponent derivation attack and discusses the design pitfall. Yang et al.'s scheme sends the blinded exponents to secure the exponents. Unfortunately, these not-well-formed DH exponents are vulnerable to our Exponent Derivation Attacks (EDA), and the attackers can derive the secret keys and the session keys, and then impersonate the clients.

Deriving the secret keys from the blinded DH exponents. In this approach, we just eavesdrop the transmissions, and try to derive clients' secret keys from the blinded DH exponents. We can plot this attack either as an outside attacker or an inside attacker.

Outside attacker. We observe that the transmission ($S, B, w_A, R'_B, Auth'_A$) and ($S, A, w_B, R'_A, Auth'_B$) in

Round 3 could be used to derive clients' secret keys $x'_A = H_2(A,s)$ and $x'_B = H_2(B,s)$. An attack for deriving A's secret key is shown as follows.

(1) First we eavesdrop on several communications in which A is involved- $(S,B_1,w_{A,1},R'_{B_1},Auth'_{A,1})$, $(S,A,w_{B_1},R'_{A,1},Auth'_{B_1})$, $(S,B_2,w_{A,2},R'_{B_2},Auth'_{A,2})$, $(S,A,w_{B_2},R'_{A,2},Auth'_{B_2})$, …, $(S,B_n,w_{A,n},R'_{B_n},Auth'_{A,n})$, where $B_{i,1\le i\le n}$ denotes some clients that A has interacted with and these $B_{i,1\le i\le n}$ need not to be same one. That is, we just eavesdrop on some communications of which A is involved. Note that $w_{A,1} = t_1 \cdot x'^{-1}_A \bmod q$, $w_{A,2} = t_2 \cdot x'^{-1}_A \bmod q$, …, $w_{A,n} = t_n \cdot x'^{-1}_A \bmod q$ for some unknown integers $t_1,t_2,...,t_n$.

(2) Take $\gcd(w_{A,1},w_{A,2},...,w_{A,n}) = x'^{-1}_A \cdot \gcd(t_1,...,t_n)$, where gcd denotes the greatest common divisor. According to Cohen [4], the probability that n randomly chosen integers being co-prime is $1/\zeta(n)$, where $\zeta(n)$ is the Riemann zeta function [24]. We have $1/\zeta(2)$ =0.60792…, $1/\zeta(3)$=0.83190…, $1/\zeta(4)$ =0.924047…, and $1/\zeta(5)$ =0.964388. That is, by simply eavesdropping 5 sessions, we have $\gcd(w_{A,1},w_{A,2},...,w_{A,n}) = x'^{-1}_A \cdot \gcd(t_1,...,t_4) = x'^{-1}_A$ with probability 0.964388. We could eavesdrop more sessions to increase the probability if necessary.

(3) When one candidate x'_A is got, we compute the corresponding $t_1 = w_{A,1} \cdot x'_A$ and $W_{A,1} = R'^{t_1}_{A,1}$. Then we use the derived values to verify the candidate by testing whether the computed $Auth'_{A,1} = H_5(S,B_1,w_{A,1},R'_{B_1},W_{A,1})$ equals the eavesdropped $Auth'_{A,1}$. If so, this candidate x'_A is the real one; otherwise, we eavesdrop more sessions and go to Step 2.

Once the secret x'_A is found, it is very easy to cheat A into accepting the wrong data and wrong session key. We show one example of such attacks as follows.

(1) We initiate a new session with A or wait for a new session in which A involves; We eavesdrop on messages $(A,V_A,Auth_A, B,V_B,Auth_B)$ in Round 1 and Round 2;

(2) We intercept S's message $(S,B,w_A,R_B,Auth'_A)$ and $(S,A,w_B,R_A,Auth'_B)$ in Round 3.

(3) We randomly choose two integers t and b, and compute $\overline{w}_A = t \cdot x'^{-1}_A \bmod q$, $\overline{R}_B = g^b$, $\overline{W}_A = R_A^t$ and $Auth''_A = H_5(S,B,\overline{w}_A,\overline{R}_B,\overline{W}_A)$. We send, on behalf of S, the data $(S,B,\overline{w}_A,\overline{R}_B,Auth''_A)$ to A in Round 3. Apparently, A will accept this forged message because the attacker uses the correct x'_A to forge the data. Finally A computes the session key $sk_A = H_6(A,B,R_A,\overline{R}_B,\overline{R}_B^{\,a} = g^{ab})$. The attacker could compute the same session key because he can use the eavesdropped value R_A to compute $R_A^{\,b}$ and $sk_B = H_6(A,B,R_A,\overline{R}_B,R_A^{\,b} = g^{ab})$. The impersonation attack succeeds and the attacker shares the same session key with A.

Probability of deriving the secrets and impersonating the client. In the above attack, the attacker only needs 5 sessions to have a successful rate approaching 0.9643. We could eavesdrop more sessions to increase the success rate.

Inside attacker. A client (say A) can easily derive its communicating party's (say B) secrets using the responses from the server. We notice that the server will respond $(S,B,w_A,R'_B,Auth'_A)$ and $(S,A,w_B,R'_A,Auth'_B)$ in Round 3. Since $w_A = t \cdot H_2(A,s)^{-1} \bmod q$ and $w_B = t \cdot H_2(B,s)^{-1} \bmod q$, both the client A and the client B can derive the value t and then derive the secret of the communicating party's secrets $H_2(A,s)^{-1}$ and $H_2(B,s)^{-1}$ respectively.

For an insider, the above attack only needs one session to derive communicating party's secrets.

Importance of the demonstrated attacks. As more and more resource-limited devices have been deployed in our daily life, the security threats are becoming larger and larger. How to design secure protocols that meet the resource constraints is very crucial. Yang et al. have proposed one possible mechanism to reducing the number of the costly exponentiation computations by blinding the exponent components. However, we have demonstrated the secret key disclosure attack, the session key disclosure attack, and the impersonation attack against this approach. We argue that this approach is not secure at all, and should be avoided.

5 Countermeasure to Exponent Derivation Attacks and the Improved 3PAKA Scheme

5.1 Countermeasures to Exponent Derivation Attacks

Conventional DH key agreement schemes require the participants send their ephemeral DH public keys and keeping the secret exponents private. This approach aims to establish the session keys, based on the CDHP problem. It requires each participant at least two modular exponentiations, which could be a burden for many resource-limited devices. Yang et al. use random numbers to blind the transmission of the secret exponents $w_A = t \cdot H_2(A,s)^{-1} \bmod q$. This approach could save one modular exponentiation for the sender and the receiver; however, the security of their approach does not depend on any hard problem. That is

why we can plot the attacks in Section 4.

Luckily, Chien [21] has formulated a new hard problem called the MCDHP, and has proved its hardness being equivalent to that of the CDHP [21]. We review the MCDHP as follows and will design a new 3PAKA scheme, based on the MCDHP. Interested readers are referred to the publication [21].

Definition 1 [21]**.** The *Modified Computational Diffie-Hellman Problem* (the MCDHP) over G is defined as follows: Given $x+t$, g, g^t and g^y, where t, x and y are random numbers and g is a generator for the group G, the challenge is to compute g^{xy}.

We briefly quote the proof as follows. Interested readers are referred to the publication [21] for the details.

Theorem 1. The MCDHP problem is as hard as the CDHP problem [21].

Proof. We prove this by reduction.

(1) The MCDHP problem is reduced to the CDHP.

Given an instance of the MCDHP problem- ($x+t$, g, g^t and g^y), then we can compute $g^x = g^{(x+t)} / g^t$ and get the instance (g, g^x and g^y) for the CDHP problem. Assume there is one oracle that can answer the CDHP problem. Now we input the instance (g, g^x and g^y) to the oracle, and we get the answer g^{xy}.

(2) The CDHP problem is reduced to the MCDHP problem.

Assume there is one oracle that can answer the MCDHP problem: given ($x+t$, g, g^t and g^y), it outputs g^{xy}.

Now given an instance of the CDHP problem- (g, g^x and g^y), we then choose a random value t, and input the instance (t, g, g^x and g^y) to the MCDHP oracle. The oracle will answer $(g^t / g^x)^y = g^{ty-xy}$. Using the response, we can derive $(g^{ty-xy} \cdot g^{-ty})^{-1} = g^{xy}$. That is, we get the answer for the CDHP problem- (g, g^x and g^y).

Based on the above arguments, we prove the theorem.

5.2 The Proposed Scheme

We, based on the MCDHP, propose an improved 3PAKA scheme which not only improves the security properties but also reduces the computational load of the clients and the communication steps.

The scheme consists of three phases: the initialization phase, the authentication phase, and the password updating phase. There are two kinds of entities: one is the Server (S) and the others are registered clients. Figure 1 depicts the scheme. We respectively describe the detailed phases as follows.

The initialization phase: the S publishes the parameters (G, g, p, q), and the clients register their public keys and get the issued smart cards as follows. The following steps are conducted in a secure channel.

1. $A \Rightarrow S$: $ID_A, h(pw_A) \oplus h_{bio}(bio_A), Y_A$

A chooses his private key pr_A and computes the public key $Y_A = g^{pr_A}$. He then inputs its password pw_A, scans his biometric information bio_A, and computes $h(pw_A)$ and $h_{bio}(bio_A)$. Finally he sends $ID_A, h(pw_A) \oplus h_{bio}(bio_A), Y_A$ to S.

2. $S \Rightarrow A$: smart card(ID_A, Z_A)

S computes $K_A = h(ID_A, s)$ and issue a smart card which contains ID_A, Z_A, where $Z_A = K_A \oplus h(pw_A) \oplus h_{bio}(bio_A)$.

3. A:

Upon receiving the card, A derives K_A, and writes Z_A, $pr_A \oplus h_{bio}(bio_A)$ and $W_A = h(ID_A \| K_A \| h(pw_A))$ into the card.

The authentication phase:

(1) $A \rightarrow S$: $sid, A, B, x + pr_A, M1$

A inputs his password pw_A and scans his biometric information bio_A. The card derives $K_A = Z_A \oplus h(pw_A) \oplus h_{bio}(bio_A)$ and pr_A, and verifies whether $W_A ?= h(ID_A \| K_A \| h(pw_A))$ holds. If it does not hold, the card terminates the operations; otherwise, it chooses an ephemeral private key x and a randomly chosen number R_A, and then encrypts its message as $M1 = E_{K_A}[A \| B \| x + pr_A \| R_A]$. It sends the message "$sid, A, B, x + pr_A, M1$" to S to initiate the request.

(2) $S \rightarrow B$: $sid, A, B, M2$

S first derives $K_A = h(ID_A, s)$, decrypts $E_{K_A}[A \| B \| x + pr_A \| R_A]$ and checks the correct format and the existence of $x + pr_A$. It then computes $g^x = g^{x+pr_A} / Y_A$. It finally computes $M2 = E_{K_B}[A \| B \| x + pr_A \| g^x]$ and sends it to B.

(3) $B \rightarrow S$: $sid, A, B, y + pr_B, M3$

B inputs his password pw_B and biometric information bio_B. The card derives $K_B = Z_B \oplus h(pw_B) \oplus h_{bio}(bio_B)$ and pr_B, and verifies whether $W_B ? = h(ID_B \| K_B \| h(pw_B))$ holds. If so, it continues the following steps. It decrypts $E_{K_B}[A \| B \| x + pr_A \| g^x]$, chooses an ephemeral private key y and a randomly chosen number R_B, and compute $Seed = h((g^x)^y)$. It computes $M3 = E_{K_B}[A \| B \| x + pr_A \| g^x \| y + pr_B \| Seed]$ and sends "$sid, A, B, y + pr_B, M3$" to S.

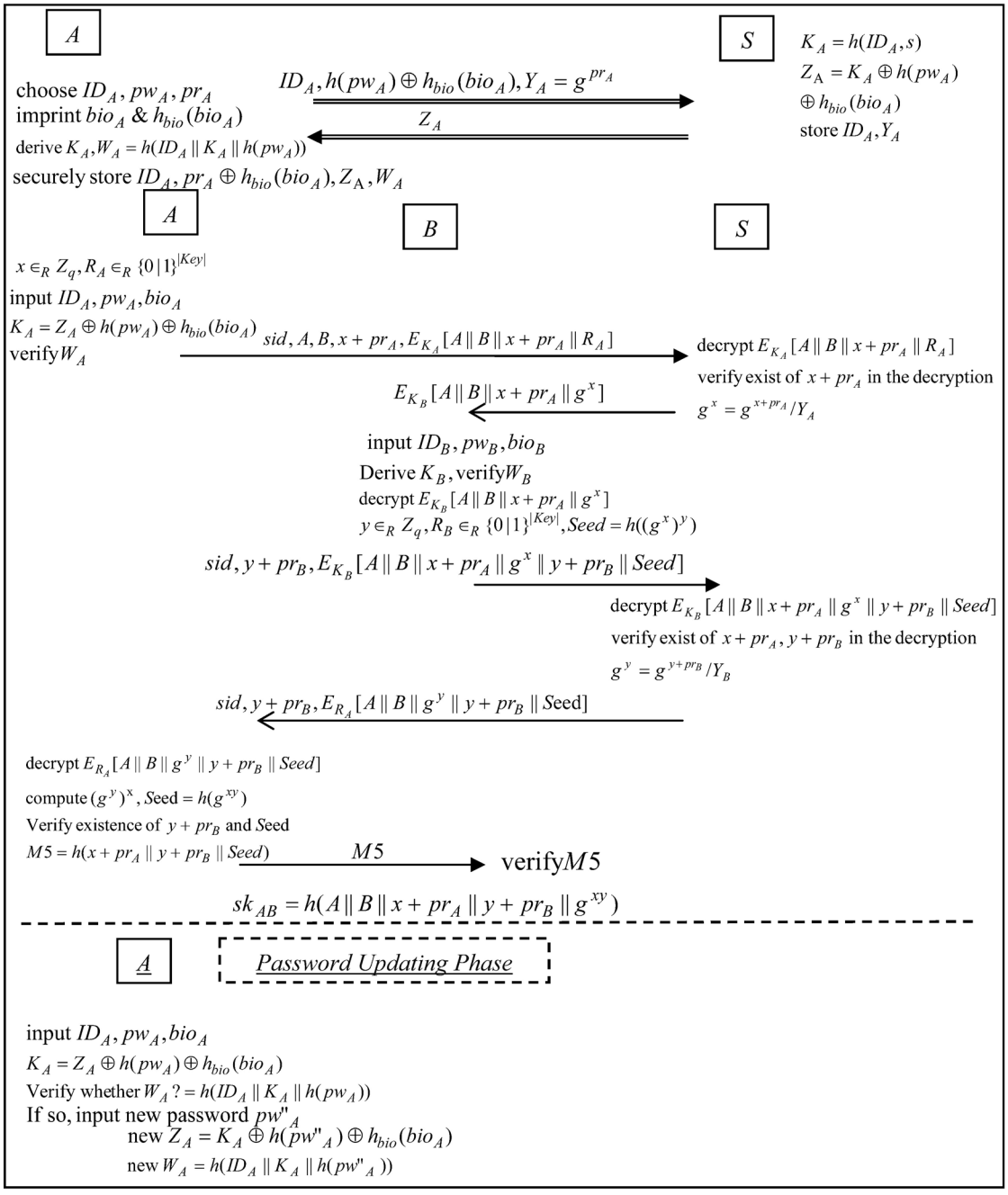

Figure 1. The MCDHP-based 3PAKA scheme

(4) $S \rightarrow A$: $sid, A, B, y + pr_B, M4$

S decrypts $M3$, derives $Seed$, and checks the existence of $x + pr_A$, $y + pr_B$, g^x. If the verification succeeds, it then computes $g^y = g^{y+pr_B} / Y_B$ and $M4 = E_{R_A}[A \| B \| g^y \| y + pr_B \| Seed]$. It sends "$sid, A, B, y + pr_B, M4$".

(5) $A \rightarrow B$: $sid, A, B, y + pr_B, M5$

A uses R_A to decrypt $M4$, computes $Seed = h((g^y)^x)$, and checks the existence of $y + pr_B$ and $Seed$. If it succeeds, then it computes $M5 = h(x + pr_A \| y + pr_B \| Seed)$

Upon receiving the message, B verifies the correctness of the received $M5$. Finally, A and B share the session key $sk_{AB} = h(A \| B \| x + pr_A \| y + pr_B \| g^{xy})$.

The password updating phase. A inputs his password pw_A and scan his biometric information bio_A. The card derives $K_A = Z_A \oplus h(pw_A) \oplus h_{bio}(bio_A)$ and pr_A, and verifies whether $W_A ? = h(ID_A \| K_A \| h(pw_A))$ holds. If it does not hold, the card terminates the operations; otherwise, it commands the user to input the new password pw''_A and updates Z_A as $K_A \oplus h(pw''_A) \oplus h_{bio}(bio_A)$.

6 Performance Evaluation and Security Analysis

We evaluate the performance of the related works in Section 6.1, analyze the security of the proposed

scheme in Section 6.2, and show the results from the AVISPA checkers' verifications.

6.1 Performance Evaluation

Table 1 summarizes several key features of several recent publications. Regarding the communication cost, Yang et al.'s scheme [1] requires the less numebr of steps and rounds; however, we should notice that the scheme provides *only key distribution* instead of key confirmation. Compared to to a key-distribution scheme, its extension to achieving key confirmation would demand at least one more step and one more round. That is, if we extend Yang et al.'s scheme to provide key confirmation, then it would at least require 4 rounds and 5 steps.

Table 1. Performance comparison of related works

	[1]			Ours			[20]			[18]			[19]		
TI	×			×			×			√			×		
NO	√			√			×			×			√		
ST	×			×			√			×			×		
#S	4(5)[3]			5			5(6)[1]			5(6)[2]			6		
#R	3(4)[3]			5			5(6)[1]			5(6)[2]			5		
KD	√			×			×			×			×		
KC	×			√			√			√			√		
Entity	A	B	S	A	B	S	A	B	S	A	B	S	A	B	S
#SE	0	0	0	2	2	4	0	0	0	0	0	0	4	4	4
#AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
#RN	1	1	2	2	1	0	1	1	1	1	1	0	2	2	0
#H	3	3	6	2	2	0	10	10	7	3	3	2	1	1	0
#E	4	4	4	1	1	2	0	0	0	3	3	2	1	1	2
#M	2	2	2	0	0	2	0	0	0	0	0	0	0	0	2
SW	SD, SKD, IMA			none			DOS			none			SVP		

Both our scheme and Chen et al.'s scheme [18] require only 5 steps in 5 rounds; however, the scheme [18] depends on timestamp synchronization to ensure the message freshness and to conquer the possible replay attacks; unfortunately, it is very difficult to synchronize timestamp in large distributed netwrks; that is why amlost all internet protocols do not depend on time synchronization to ensure the message freshness. Generally, a nonce-based scheme requires one more step than its timestamp-based counterparts. That is, if we amend Chen et al.'s scheme to its equalivent nonce-based version, then it would requires 6 steps in 6 rounds.

Even though Amin and Biswas [20] only specify 5 steps in their protocol description, their scheme should require *at least 6 steps*; this is because they assume that the two clients, without any interactions, will initiate their requests at the same time to the server; but, practically, the scheme needs one more interaction from the initiator (say *A*) to inform the responeder (say *B*) to respond. Furthermore, the scheme requires the synchronization of the secret state (the keys); without the synchronization, the authnetication fails, and *this weakness make it vulnerable to the Denial-Of-Service (DOS) attacks*.

In a short summary of communication performance comparison, our scheme requires the least number of steps when we amend or extend the compared schems [1, 18-20] to achieve the same functions. Chien and Wu [2] has proved that the optomal number of the communication steps is three for those CDHP-based 3PAKA schemes that use nonce to ensure the message freshness and provide the key confirmation. Up to now, the academia still do not know the optimal number of the steps for the MCDHP-based schemes that achieve the key confirmation and use nonce for the message freshness. It is still an open question.

Regarding the various computations of these 3PAKA schemes, the exponentiation is the most expensive computation. From the table, we can see that the scheme [20] does not require any exponentiations; that is beacause its security does not depend on any hard problems, and it only depends on hashing operations. We should exclude this scheme from the comparison when we consider only those schemes of which the security is based on hard problems like the CDHP, the MCDHP, and so on. Conventionally, a client in any CDHP-based key agreement schemes would requires at least two exponentiations. Here, we can see that our scheme and the scheme [19] require each client only one exponentiation, owing to the MCDHP. This enhancement is significant for those resource-constrained devices.

Regarding the security, Yang et al.'s scheme [1] is vulnerable to the secret key disclosure, the session key disclosure, and the impersonation attacks. The scheme [20] is vulnerable to the DOS attacks because its authentication depends on the sychronization of the updated secrets. The scheme [19] does not consider the stolen vreifier problem: the server needs to store each client's secret, and an attacker with a disclosed secret can impersonate the client. On the contrary, the server in our scheme only stores the clients' public keys but not any verifiers. This makes our scheme more securely robust.

To summarize all the performance comparison, we can see that our proposed scheme owns the least number of the communication steps and the least compuations among the related works under the same security requirements and the same criteria.

- A, B, S respectively denotes the client A, the client B, and the server S.
- TI: timestamp; NO: nonce; ST: Synchronization of State; KD: Key Distribution; KC: Key Confirmation; #SE: number of Symmetric Encryption; #AE: number of Asymmetric Encryptions; #RN: number of Random Numbers; #H: number of Hashing; #E: number of Exponentiations; #S: number of communication steps; #R: number of message rounds.
- SW: Security Weaknesses; SD: Secret Disclosure; SKD: Session Key Disclosure; DOS: Denial Of

Service; SVP: Stolen Verifier Problem; IMA: IMpersonation Attacks.

(1) Amin and Biswas [20] only specify 5 steps in their protocol description, but their scheme should require at least 6 steps.

(2) The scheme depends on the timestamp synchronization to ensure message freshness, and its nonce-based equivalent version would require at least one more step and one more round.

(3) The scheme only provides key distribution, and its key-confirmation version would demand at least one more step/round.

6.2 Security Analysis

In this section, we define the security notation and the model where the hash function is modeled as a random oracle and the symmetric encryption is INDistinguishable against Chosen Plaintext Attacks (IND-CPA). We prove the session key indistinguishability in the model. Following that, we further analyze other security properties.

For a 3PAKA protocol, $C_A \in \overline{C} = \{C_1,...,C_{N_C}\}$ denotes a client from the set of the clients and S denotes the server, where N_C denotes the number of clients. Each client $C_A \in \overline{C}$ shares a secret K_A with the server S. The adversary AD is a probabilistic machine that controls all the communications that take place between C_A^i, C_B^j, and S, where C_A^i and C_B^j denote the i-th instance of C_A and the j-th instance of C_B, respectively. AD interacts with the participants through the following oracle queries.

$Execute(C_A^i, C_B^j)$, $Execute(C_A^i, S)$: This query models passive attacks in which the attacker eavesdrops on all the communications between the instances (C_A^i, C_B^j) and between the instances (C_A^i, S) respectively.

$SendClient(C_A^i, m)$: This query models an active attack against the ith instance C_A by sending the message m. The output of this query is the message that C_A^i would generate, upon receiving the message m.

$SendServer(m)$: This query models an active attack against the server S by sending the message m. The output of this query is the message that S would generate, upon receiving the message m.

$Reveal(C_A^i)$: This query models an active attack against the ith instance C_A. The output of this query is either $\perp$ if the instance does not accept a session key or the real session key if it accepts a key.

$Corrupt(C_A)$**:** This query allows AD to corrupt the entity C_A at will, and thereby learns the complete internal state of the entity. This query can be used to model real-world scenarios of an insider co-operating with the adversary or an insider who has been completely compromised by the adversary.

$Test(C_A^i)$: If $C_A^i \in \overline{C}$ does not accept any session keys, this query outputs the undefined symbol $\perp$; otherwise, it returns either the session key of C_A^i if the random bit $b = 1$ or returns a random key of the same size if $b = 0$.

The definition of security depends on the notations of partnership/freshness of oracles [3-4]. The definition of security restricts the adversary's *Reveal and Corrupt* queries to those oracles that are not partners (two oracles are partners if, and only if, the two oracles have accepted the same session key with the same *sid*, have agreed on the same set of entities, and no other oracles have accepted the same session key with the same *sid*.) of the oracles whose key the adversary is trying to guess. An oracle C_A^i is called fresh (or it holds a fresh session key) at the end of execution, if, and only if, oracle C_A^i has accepted with or without partner oracles C_B^j, all the oracles C_A^i and C_B^j (if such a partner oracle exists) have not been sent a *Reveal* query, and the entities C_A and C_B of oracles C_A^i and C_B^j (if such a partner exists) have not been sent a *Corrupt* query.

Session key security. The session key between the two clients sk_{AB} should be secure against adversaries. We model this property, using the standard semantic security notation [3-4]. Security of the session key is defined by the adversary's in-distinguishability in telling a real key from a random one in the game G_a, played between the adversary AD and a collections of U_x^i for players $U_x \in \overline{C} \cup S$ and instances $i \in \{1,...,N_I\}$, where N_I is the maximum number of instances for each player. The adversary AD runs the game simulation G_a with the settings as follows.

- **Stage 1:** AD can send *Execute*, *SendClient, SendServer*, *Reveal*, and *Corrupt* queries in the simulation.
- **Stage 2:** At some point during G_a, AD will choose a fresh session and send a *Test* query to the fresh oracle (C_A^i or C_B^j) associated with the test session. Depending on the randomly chosen bit b, AD is given either the actual session key sk_{AB} or a random session key drawn from the session key distribution. In addition, sk_{AB} denotes the session key between the two clients.
- **Stage 3:** AD continues making any *Execute, SendClient, SendServer*, *Reveal*, and *Corrupt* oracle queries to its choice. In addition, AD is restrained from sending a *Reveal* query to the oracles of the test session, and from sending a *Corrupt* query to the entities corresponding to the test session.
- **Stage 4:** Eventually, AD terminates the game simulation and outputs its guess bit *b'*.

Success of AD in G_a against a protocol P is

measured in terms of AD's advantage in distinguishing whether it receives the real key sk_{AB} or a random one. Let the advantage function of AD be denoted by $Adv^{G_a,AD}(k)$, where k is the security parameter. Then, $Adv^{G_a,AD}(k) = |\Pr[b'=b]-1/2|$.

Definition 2 - A 3PAKA protocol is session-key secure in our model if the session key in-distinguishability is satisfied: for all probabilistic, polynomial-time adversaries AD, $Adv^{G_a,AD}(k)$ is negligible.

In addition to the general session key in-distinguishability security, our scheme also provides the key escrow-less-ness property: this property requires that even an honest but curious server cannot distinguish a session key from a random one with the same distribution.

Theorem 2. The proposed schemes own the key escrow-less-ness property.

Proof: An honest but curious server who owns the secret keys K_A and K_B can decrypt the messages and learn all the transcripts $\{x+pr_A, y+pr_B, g^x, g^y\}$ and the value $Seed=h(g^{xy})$. Now, we will prove that the server still cannot learn the session key.

We prove this by contradiction. In our scheme, the server that knows $\{x+pr_A, y+pr_B, Y_A=g^{pr_A}, Y_B=g^{pr_B}, Seed=h(g^{xy})\}$ can derive $\{g^x, g^y\}$ and would like to derive the session key $sk_{AB}=h(A\|B\|x+pr_A\|y+pr_B\|g^{xy})$. Due to the on-way property of the hash functions, the server should learn g^{xy} to attain the key, since the session key is not transmitted during the protocol.

Now, we assume that the server in our scheme can derive the value g^{xy}. Then, we can use this server as an oracle to solve the CDHP problem as follows.

Given an instance of the CDHP problem- $\{g, g^x, g^y\}$, we choose two random numbers $\bar{x}$ and $\bar{y}$, and prepare the parameters $\{\bar{x}, \bar{y}, Y_A=g^{\bar{x}}/g^x, Y_B=g^{\bar{y}}/g^y\}$ for the server. Then, the server can derive $g^{\bar{x}}/Y_A=g^x$ and $g^{\bar{y}}/Y_B=g^y$, and returns the value g^{xy}, which is the DH value for (g^x, g^y). This contradicts the hardness property of the CDHP problem. That is, our initial assumption is wrong, and the server cannot learn the session key.

Theorem 3. The proposed 3PAKA schemes satisfy the session key in-distinguishability if the MCDHP problem is a hard problem.

Proof:

Based on the result of Theorem 1, the server who knows $\{x+pr_A, y+pr_B, Y_A, Y_B, Seed=h(g^{xy}), g^x, g^y\}$ still cannot derive the session key. Since an outsider could not be more powerful than the server, the outsider cannot learn the session key. This proves Theorem 3.

Theorem 4. The proposed scheme can resist the threats and attacks caused by the stolen verifier problem.

Proof:

In our scheme, the server only keeps the registered clients' identities and public keys, for examples $Y_A=g^{pr_A}$ and $Y_B=g^{pr_B}$. If we assume that an attacker get the stolen verifiers, the public keys Y_A and Y_B in our scheme, then the attacker can only derive the values ($x+pr_A, y+pr_B, g^x, g^y, Y_A, Y_B$) but nothing else. So, even given the values ($x+pr_A, y+pr_B, g^x, g^y, Y_A, Y_B$), the attacker still cannot derive any secret keys or any session keys, owing to the hardness of the CDHP and the MCDHP.

Theorem 5. The proposed scheme can resist the threats and attacks caused by the lost smart card problem.

Proof:

For a lost smart card of the client A, an attacker can get the values $Z_A = K_A \oplus h(pw_A) \oplus h_{bio}(bio_A), pr_A \oplus h_{bio}(bio_A)$ and $W_A=h(ID_A\|K_A\|h(pw_A))$ inside the card. Based on the data, the attacker may try to guess and verify the password or try to derive any secret keys (and the session keys). However, both the secret key K_A and the private key pr_A are well protected by the user's $h_{bio}(bio_A)$. Without the secret key K_A, the attacker cannot verify his guessed passwords. That is, the proposed scheme can resist the password guessing attacks and any secret value derivation attacks, even if he captures and compromise the lost card.

Table 2 summarizes the security-related properties of the proposed scheme and the counterparts. In addition to the session key indistinguishability and the session-key escrow-less-ness property, our scheme is also immune to the stolen verifier problem and the lost smart card problem. Our scheme also does not depend on any synchronization of updated secrets: that makes it robust against the DOS attacks. Based on Table 2, we can see that both our scheme and the scheme [20] show the security robustness against all the security threats. However, from the previous table, Table 1, the scheme [20] needs one more message step than our scheme, and the scheme also demands more expensive exponentiation computations.

6.3 Security Verifications Using the AVISPA

Using the HLPSL specification [27], we specify three roles: "mobile", "gateway", and "server". The "mobile" and the "gateway" model the two clients, and the "server" models the trusted server. The specified security goals in our specifications include authentication among the parties and the secrecy of the session keys.

Table 2. Summary of the security properties of the related schemes

	[1]	Ours	[20]	[18]	[19]
TI	X	X	X	V	X
NO	V	V	X	X	V
ST	X	X	√	X	X
KD	V	X	X	X	X
KC	X	V	V	V	V
Session Key Disclosure	V	X	X	X	X
Denial of Service attack	X	X	V	X	X
Password guessing attack	V	X	X	X	X
Server get the session keys	V	X	X	X	X
Impersonation attack	V	X	X	X	X
Stolen Verifier Problem	X	X	X	X	V
Lost Smart Card Problem	V	X	X	X	X

Note. TI: timestamp; NO: nonce; ST: Synchronization of State; KD: Key Distribution; KC: Key Confirmation.

We run simulation (in Figure 2) and the OFMC/ CL-AtSe checkers. The results verify "SAFE" (Figure 3 shows the results from OFMC.

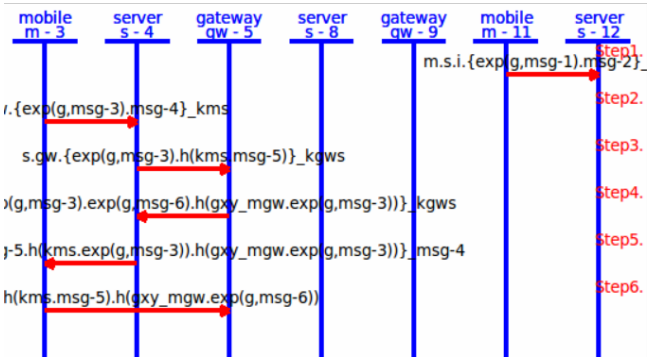

Figure 2. The protocol simulation of our scheme using AVISPA

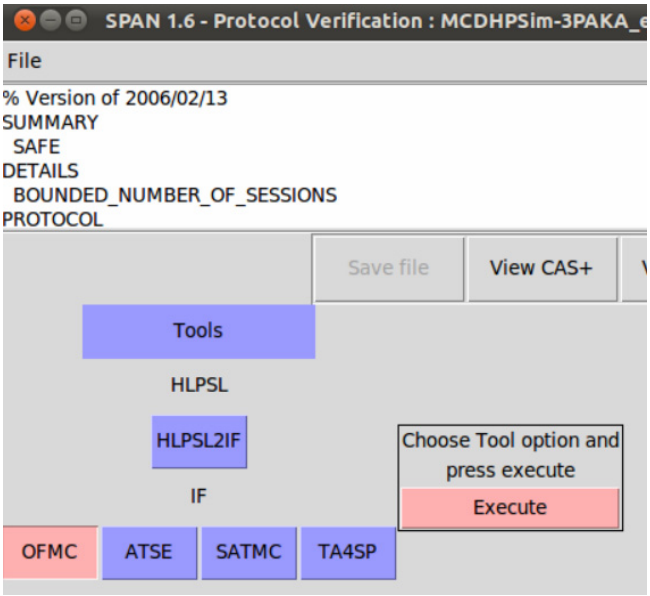

Figure 3. OFMC verifies the security of our scheme

7 Conclusions and Future Work

In this paper, we have shown a novel attack, the exponent derivation attack, on Yang et al.'s scheme; this attack is very powerful that it can derive the clients' secret keys and the session keys by only eavesdropping the communications. As more and more resource-limited devices have been deployed, it is crucial to design more efficient Diffie-Hellman key agreement schemes for these devices. Here, we have demonstrated that Yang et al.'s approach of reducing the number of the exponentiation is not secure, and this approach should be avoided for any key agreement designs.

To conquer the security weaknesses and to enhance the computation performance, we, based on the MCDHP, have proposed an improved scheme. Both the security analysis and the AVISPA verification have verified the security properties of our scheme. The proposed scheme shows the best performance in terms of security, communications and computations, when we evaluate the related works under the same criteria. Especially, its reduction of exponentiation computation makes it very attractive to those resource-limited devices. Up to now, we are still not sure whether the communication steps of the MCDHP-based 3PAKA scheme with key confirmation could be further reduced. It is still an open question and it is our future work.

Acknowledgements

This project is partially supported by the Ministry of Science and Technology, Taiwan, R.O.C., under grant No. MOST 106-3114-E-260-001, MOST 107-2218-E-260-001.

References

[1] H. Yang, Y. X. Zhang, Y. Z. Zhou, X. M. Fu, H. Liu, A. V. Vasilakosc, Provably Secure Three-party Authenticated Key Agreement Protocol Using Smart Cards, *Computer Networks*, Vol. 58, pp. 29-38, January, 2014.

[2] C. C. Chang, Y. F. Chang, A Novel Three-party Encrypted Key Exchange Protocol, *Computer Standards and Interfaces*, Vol. 26, No. 5, pp. 471-476, September, 2004.

[3] H. Y. Chien, T. C. Wu, Provably Secure Password-based Three-party Key Exchange with Optimal Message Steps, *The Computer Journal*, Vol. 52, No. 6, pp. 646-655, August, 2009.

[4] H. R. Chung, W. C. Ku, Three Weaknesses in a Simple Three-party Key Exchange Protocol, *Information Sciences*, Vol. 178, No. 1, pp. 220-229, January, 2008.

[5] L. Gong, Lower Bounds on Messages and Rounds for Network Authentication Protocols, *Proc. of the 1st ACM Conference on Computer and Communications Security*, Fairfax, VA, USA, 1993, pp. 26-37.

[6] W. S. Jaung, Efficient Three-party Key Exchange Using Smart Cards, *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 2, pp. 619-624, May, 2004.

[7] J. O. Kwon, I. R. Jeong, D. H. Lee, Three-round Smart Card-based Key Exchange Scheme, *IEICE Transactions on Communications*, E90-B, No. 11, pp. 3255-3258, November, 2007.

[8] T. F. Lee, I. P. Chang, C. C. Wang, Efficient Three-party Encrypted Key Exchange Using Trapdoor Functions, *Security and Communication Networks*, Vol. 6, No. 11, pp. 1353-1358, November, 2013.

[9] T. F. Lee, T. Hwang, C. L. Lin, Enhanced Three-party Encrypted Key Exchange without Server Public Keys, *Computers and Security*, Vol. 23, No. 7, pp. 571-577, October, 2004.

[10] C. H. Lim, P. J. Lee, A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup, *Advances in Cryptology (Crypto'97)*, Santa Barbara, CA, USA, 1997, pp. 249-263.

[11] C. L. Lin, H. M. Sun, T. Hwang, Three-party Encrypted Key Exchange: Attacks and a Solution, *ACM Operating System Review*, Vol. 34, No. 4, pp. 12-20, October, 2000.

[12] C. L. Lin, H. M. Sun, M. Steiner, T. Hwang, Three-party Encrypted Key Exchange without Server Public-keys, *IEEE Communications Letters*, Vol. 5, No. 12, pp. 497-499, December, 2001.

[13] R. Lu, Z. Cao, Simple Three-party Key Exchange Protocol, *Computers & Security*, Vol. 26, No. 1, pp. 94-97, February, 2007.

[14] J. Nam, S. Kim, D. Won, Attack on the Sun-Chen-Hwang's Three-Party Key Agreement Protocols Using Passwords, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E89-A, No.1, pp. 209-212, January, 2006.

[15] M. Steiner, G. Tsudik, M. Wainder, Refinement and Extension of Encrypted Key Exchange, *ACM Operation Systems Review*, Vol. 29, No. 3, pp. 22-30, July, 1995.

[16] H. M. Sun, B. C. Chen, T. Hwang, Secure Key Agreement Protocols for Three-party against Guessing Attacks, *The Journal of Systems and Software*, Vol. 75, No. 1-2, pp. 63-68, February, 2005.

[17] Z.W. Tan, A Communication and Computation-efficient Three-party Authenticated Key Agreement Protocol, *Security and Communication Networks*, Vol. 6, No. 7, pp. 854-863, July, 2013.

[18] C. M. Chen, L. L. Xu, W. C. Fang, T. Y. Wu, A Three-party Password Authenticated Key Exchange Protocol Resistant to Stolen Smart Card Attacks, in: J. S. Pan, P. W. Tsai, H. C. Huang (Eds.), *Advances in Intelligent Information Hiding and Multimedia Signal Processing. Smart Innovation, Systems and Technologies*, Vol. 63. Springer, 2017, pp. 331-336.

[19] H. Y. Chien, Using the Modified Diffie-Hellman Problem to Enhance Client Computational Performance in a Three-party Authenticated Key Agreement, *Arabian Journal for Science and Engineering*, Vol. 43, No. 2, pp 637-644, February 2018.

[20] R. Amin, G. P. Biswas, Cryptanalysis and Design of a Three-party Authenticated Key Exchange Protocol Using Smart Card, *Arabian Journal for Science and Engineering*, Vol. 40, No. 11, pp. 3135-3149, November, 2015.

[21] H. Y. Chien, A Generic Approach to Improving Diffie-Hellman Key Agreement Efficiency for Thin Clients, *The Computer Journal*, Vol. 59, No. 4, pp. 592-601, April 2016.

[22] A. T. B. Jin, D. N. C. Ling, A. Goh, Biohashing: Two Factor Authentication Featuring Fingerprint Data and Tokenised Random Number, *Pattern Recognition*, Vol. 37, No. 11, pp. 2245-2255, November, 2004.

[23] A. Lumini, L. Nanni, An Improved BioHashing for Human Authentication, *Pattern Recognition*, Vol. 40, No. 3, pp. 1057-1065, March, 2007.

[24] E. Cohen, Arithmetical Functions Associated with Arbitrary Sets of Integers, *Acta Arithmetica*, Vol. 5, pp. 407-415, 1959. DOI: 10.4064/aa-5-4-407-415.

[25] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganó, L. Vigneron, The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications, *17th International Conference on Computer Aided Verification (CAV'2005)*, Edinburgh, Scotland, 2005, pp. 281-285.

[26] Avispa - a tool for Automated Validation of Internet Security Protocols, http://www.avispa-project.org, May, 2019.

[27] D. von Oheimb, The High-Level Protocol Specification Language Hlpsl Developed in the EU Project Avispa, *APPSEM 2005 Workshop*, Frauenchiemsee, Germany, 2005.

[28] D. Basin, S. Mödersheim, L. Viganó. OFMC: A Symbolic Model-Checker for Security Protocols, *International Journal of Information Security*, Vol. 4, No. 3, pp.181-208, June, 2005.

[29] M. Turuani, The CL-Atse Protocol Analyser, *Proc. of 17th International Conf. on Rewriting Techniques and Applications (RTA)*, Seattle, WA, USA, 2006, pp. 277-286.

[30] SPAN: A Security Protocol ANimator for AVISPA, http:// people.irisa.fr/Thomas.Genet/span/; https://raweb.inria.fr/ rapportsactivite/ RA2007/lande/uid30.html, May, 2019.

[31] R. Amin, SK H. Islam, G. P. Biswas, M. S. Obaidat, A Robust Mutual Authentication Protocol for WSN with Multiple Base-stations, *Ad Hoc Networks*, Vol. 75-76, pp. 1-18, June, 2018.

[32] H.-Y. Chien, An Effective Approach to Solving Large Communication Overhead Issue and Strengthening the Securities of AKA Protocols, *International Journal of Communication Systems*, Vol. 31, No. 1, e3381, January, 2018.

[33] H.-Y. Chien, Group-oriented Range-bound Key Agreement for Internet-of-Things Scenarios, *IEEE Internet of Things Journal*, Vol. 5, No. 3, pp. 1890-1903, June, 2018.

Biography

Hung-Yu Chien received the B.S. degree from NCTU, Taiwan, 1988, the M.S. degree from NTU, Taiwan, 1990, and the doctoral degree in applied mathematics at NCHU 2002. He is a professor of National Chi Nan University since 199808. His research interests include cryptography, networking, network security, ontology, and Internet-of-Things.