JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 1059-1072 (2011)

Security Analysis and Enhancements of Verifier-Based Password-Authenticated Key Exchange Protocols in the Three-Party Setting

SHUHUA WU
Department of Networks Engineering
Zhengzhou Information Science Technology Institute
He'nan, 450002 P.R. China
E-mail: wushuhua726@sina.com.cn

This paper investigates verifier-based password authenticated key exchange (PAKE) protocols in the three party setting. We first show that the protocol recently proposed by Li *et al.* is vulnerable to off-line dictionary attack and unknown key-share attack. Moreover, we also show that the direct elliptic curve (EC) analog of the DL based protocol proposed by Kwon *et al.* can't resist the off-line password guessing attack. Thereafter we present an enhanced protocol that can be securely implemented over elliptic curves. And yet, our proposal is simple and efficient. Therefore, the protocol is quite popular in low resource environments. Finally, as a result of our work, we also hope to have contributed towards a better understanding that it is important to study the precise adaptation of DL-based password authenticated protocols since direct EC analogs of DL based protocols may be susceptible to some new attacks.

Keywords: cryptanalysis, three-party, verifier-based, password authenticated key exchange, elliptic curve

1. INTRODUCTION

The password-based authenticated key exchange (PAKE) is a protocol which allows two communicating parties to prove to each other that they know the passwords (that is, password-based authentication), and to generate a fresh symmetric key securely such that it is known only to the two parties (that is, key exchange). The intrinsic problem with password-based protocols is that the memorable password, associated with each user, has low entropy, so that it is not easy to protect the password information against dictionary attacks – the notorious password guessing attacks by which attackers could search the relatively small space of human-memorable passwords. To address this problem, numerous schemes have been designed to be secure even when the secret key is a password during the last decades.

This paper is mainly interested in three-party password-based authenticated key exchange (3PAKE) protocols, in which a trusted server exists to mediate between two communication parties to allow mutual authentication and each user only shares one password with the server. Such protocols are quite useful in a large scale peer-to-peer communication system. However, most of 3PAKE protocols in the existing literature, *e.g.* [1-6], assume that each user and a server use the same knowledge related with a password to authenticate each other. There is a common problem in these protocols. That is, if the server is compromised, an adversary who gains access to the server's local data-

Received August 26, 2009; revised October 19, 2009; accepted November 5, 2009.
Communicated by Chin-Laung Lei.

base of passwords can immediately use any of these passwords to impersonate these users (without executing any off-line dictionary attack). Therefore protocols in a verifier-based model are designed to limit the damage of a server compromise. In a verifier-based protocol, a server has verifiers of the passwords instead of the passwords themselves, so the server compromise does not directly reveal the passwords, themselves. Of course, this mechanism will not prevent an adversary from mounting (off-line) dictionary attacks but it will slow him or her down and thus give the server's administrator time to react appropriately and to inform its users.

So far as we know, only the schemes in [7-10] are verifier-based PAKE schemes in the three-party setting. The verifier-based three-party PAKE scheme in [7] is based on the two-party scheme in [11]. However, the protocol in [11] is not only vulnerable to an off-line dictionary attack but it is also incomplete [12]. Therefore, it is true that similar attacks are applicable to the scheme in [7]. The authors in [12] suggested a countermeasure to the attacks for the scheme in [11], but the same countermeasure seems to be not applicable to the scheme in [7] because of their different design approach for the three-party setting. Though attractive and natural, the construction in [8] is proposed only for Discrete Logarithm (DL) settings. However, in a low resource environment, the natural choice for cryptographic protocols would be elliptic curve (EC) implementation because of the well-known advantages with regard to processing and size constraints [13, 14]. Due to that, quite recently, two verifier-based 3PAKE protocols via elliptic curves were suggested in [9, 10] respectively. They were promised to be secure against several attacks, including off-line dictionary attack.

Unfortunately, both of the protocols in [9, 10] are insecure. In this paper, we show that the protocol in [9] is still vulnerable to an off-line dictionary attack and an unknown key-share attack. And similar attacks can be mounted to attack the other protocol. Furthermore, we also show that the direct EC analog of the DL based protocol proposed by Kwon *et al.* [8] is completely insecure while the original protocol in [8] is secure in DL settings. We provide an attack to illustrate that the direct EC analog of the protocol in [8] can't resist the off-line password guessing attack. Thereafter we present an enhanced verifier-based 3PAKE protocol that can be securely implemented over elliptic curves. And yet, our proposal is simple and efficient. Therefore, the protocol is quite popular in low resource environments because of the remarkable efficiency. As stated in [8], it is worth noting that converting a three-party PAKE protocol in a symmetric model into a three-party PAKE protocol in a verifier-based model is not at all easy since a mechanism used for conversion should not reveal any redundancy information that then adversaries can use to perform an off-line dictionary attack. Problems in [7-10] illustrates that the design of such protocols remains a hard problem despite years of research. Finally, as a result of our work, we also hope to have contributed towards a better understanding that it is important to study the precise adaptation of DL-based password authenticated protocols since direct EC analogs of DL based protocols may be susceptible to some new attacks.

The remainder of this paper is organized as follows. Section 2 introduces some notations and definitions used throughout this paper. Section 3 briefly reviews Li *et al.*'s 3PAKE protocol and then gives some attacks to demonstrate its weaknesses. Section 4 provides the direct EC analog of Kwon *et al.*'s 3PAKE protocol and then demonstrates its weakness. Section 5 presents an enhanced verifier-based 3PAKE protocol along with efficiency and security analysis. Finally, conclusion is presented in section 6.

2. PRELIMINARIES

In this section, we will briefly introduce some notations and definitions.

Let $\mathcal{E}$ be an elliptic curve defined over a finite field K with large group order [13, 14] and P be a point in $\mathcal{E}$ with large order q, where q is a secure large prime. $\mathbb{G}$ denotes the cyclic additive group generated by P. Let e be an admissible bilinear map in $\mathbb{G} \times \mathbb{G}$ onto a group of the qth roots of unity in $\overline{K}$ (the algebraic closure of K [15]), satisfying the following three conditions:

- e is bilinear, *i.e.* $e(aP, bP) = e(P, P)^{ab}$ for all $a, b \in \mathbb{Z}_q$.
- e is non-degenerate, *i.e.* $e(P, P) \neq 1$.
- e is efficiently computable.

For an elliptic curve over a finite field, the map e can be derived from the Weil pairing or Tate pairing on it. We refer to [16, 17] for a more comprehensive description of it.

In this paper, we believe some variation of the computational Diffie-Hellman (CDH) assumption holds in $\mathbb{G}$. And we will introduce it in section 5.3.2.

3. LI *ET AL.*'S PROTOCOL AND ITS WEAKNESSES

This section describes the 3PAKE protocol proposed by Li *et al.* [9] and then gives attacks to demonstrate its weaknesses respectively.

3.1 Description of Li *et al.*'s Protocol

There are three entities involved in the protocol: the authentication server S, and two users A(initiator) and B(responder) who wish to establish a session key between them. And the protocol is divided into two phases: the initialization phase and the authenticated key exchange phase. Here, we just follow the description in [9]. In the initialization phase, it assumes that each user I must register to the server S so they share the verifier $Y_I = \mathcal{G}(I||S||pw_I)P$ for password pw_I and the public information, where $\mathcal{G}$: $\{0, 1\}^* \rightarrow Z_q^*$ is a hash function. In the authenticated key exchange phase, A and B authenticate each other with S's help, then A and B can share a common session key. This phase is divided into four rounds as follows.

Round 1: The initiator A broadcasts (A, B, S).

Round 2: S choose random number $s_A, s_B \in Z_q^*$, compute $S_A = s_AP$, $S_B = s_BP$, $S_A^* = S_A + Y_A$, $S_B^* = S_B + Y_B$, and send (S, S_A^*), (S, S_B^*) to A and B respectively.

Round 3:

(1) Upon receiving (S, S_A^*), A selects random number $a_0, a_1, a_2 \in Z_q^*$, computes $V_{A0} = a_0P$, $V_{A1} = a_1P$, $V_{A2} = a_2P$ and $S_A = S_A^* - Y_A$, $V_{AS} = a_0S_A$. Let T_{AS}, T_{A1} and T_{A2} be the x-coordinate of V_{AS}, V_{A1} and V_{A2}, respectively. Then, A computes Z_A by the following equation: $Z_A = T_{AS}X_A - a_1T_{A1} - a_2T_{A2}$. A sends the message $(A, V_{A0}, V_{A1}, V_{A2}, Z_A)$ to S.

(2) Upon receiving (S, S_B^*), B analogously computes V_{B0}, V_{B1}, V_{B2} and Z_B, and sends $(B, V_{B0}, V_{B1}, V_{B2}, Z_B)$ to S.

Round 4:

(1) Upon receiving (A, V_{A0}, V_{A1}, V_{A2}, Z_A), S computes V_{AS} by the equation $V_{AS} = s_A V_{A0}$ and extracts T_{AS}. Then, A checks the validation of the message by the following equation: $T_{AS}Y_A \stackrel{?}{=} Z_A P + T_{A1}V_{A1} + T_{A2}V_{A2}$. S moves to the next phase if it holds or terminates otherwise. (S analogously checks the validity of the message coming from B.)

(2) S selects a random number $x_A \in Z_q^*$, computes $V_{SA} = x_A P$ and $Q_A = T_{AS}(V_{B1} + V_{B2}) - x_A Y_A$, and sends ($S$, V_{B1}, V_{B2}, V_{SA}, Q_A) to A.

(3) S selects a random number $x_B \in Z_q^*$, computes $V_{SB} = x_B P$ and $Q_B = T_{BS}(V_{A1} + V_{A2}) - x_B Y_B$, and sends ($S$, V_{A1}, V_{A2}, V_{SB}, Q_B) to B analogously.

Key computation

(1) Upon receiving (S, V_{B1}, V_{B2}, V_{SA}, Q_A), A terminates if $T_{AS}(V_{B1} + V_{B2}) \stackrel{?}{=} Q_A + X_A V_{SA}$ does not hold or computes the four shared session keys as below: $K_{A1} = a_1 V_{B1} K_{A2} = a_2 V_{B1} K_{A3} = a_1 V_{B2} K_{A4} = a_2 V_{B2}$.

(2) Upon receiving (S, V_{A1}, V_{A2}, V_{SB}, Q_B), B checks the validation of the message by following equation: $T_{BS}(V_{A1} + V_{A2}) \stackrel{?}{=} Q_B + X_B V_{SB}$. If it holds, B computes the four shared session keys as follows: $K_{B1} = b_1 V_{A1} K_{B2} = b_1 V_{A2} K_{B3} = b_2 V_{A1} K_{B4} = b_2 V_{A2}$.

The correctness of the protocol follows from the fact that, in an honest execution of the protocol, $K_{A1} = a_1 b_1 P = K_{B1}$, $K_{A2} = a_2 b_1 P = K_{B2}$, $K_{A3} = a_2 b_1 P = K_{B3}$ and $K_{A4} = a_2 b_2 P = K_{B4}$.

3.2 Weaknesses of Li *et al.*'s Protocol

Unfortunately, Li *et al.*'s first protocol [9] described above is insecure. In this section, we will show it is vulnerable to an off-line password guessing attack and an unknown key-share attack.

Firstly, we show that the off-line password guessing attack is still effective in Li *et al.*'s protocol. In Li *et al.*'s protocol, since all transcripts are transmitted over an open network, a benign (passive) adversary, can easily obtain the valid message transcripts of (A, V_{A0}, V_{A1}, V_{A2}, Z_A) and (S, V_{B1}, V_{B2}, V_{SA}, Q_A) such that $T_{AS}Y_A = Z_A P + T_{A1}V_{A1} + T_{A2}V_{A2}$ and $T_{AS}(V_{B1} + V_{B2}) = Q_A + X_A V_{SA}$. The adversary can guess a password pw_A^* from its space $\mathcal{D}$ and derive the corresponding $Y_A^* = X_A^* P$, then verify it by checking $e(Z_A P + T_{A1}V_{A1} + T_{A2}V_{A2}, V_{B1} + V_{B2}) \stackrel{?}{=} e(Y_A^*, Q_A + X_A^* V_{SA})$. If it does hold, the adversary has guessed the correct secret password $pw_A^* = pw_A$. Otherwise, the adversary repeatedly guesses a new password pw_A^* from $\mathcal{D}$ until $e(Z_A P + T_{A1}V_{A1} + T_{A2}V_{A2}, V_{B1} + V_{B2}) \stackrel{?}{=} e(Y_A^*, Q_A + X_A^* V_{SA})$ holds. On the other hand, the adversary can analogously guess pw_B. Therefore, the above guessing attack is not a usual brute fore attack, and Li *et al.*'s protocol is still vulnerable to such an offline password guessing attack.

Secondly, we show that Li *et al.*'s protocol still falls to an unknown key-share attack in the presence of an active adversary. In particular, any legitimate user not supposedly involved in a protocol run, say C, who share the verifier $\mathcal{G}(C||S||pw_C)P$ for password pw_C and the public information, can end up sharing a session key with user A but with A thinking it is sharing with user B who is not sharing any key with A or C. A more detailed description of the attack is as follows,

(1) The protocol steps proceed as normal with A broadcasting (A, B, S) to B and S respectively notifying them that it wishes to initiate a session.
(2) Another user C intercepts the message (A, B, S) and instead sends (A, C, S) to S as if it originated from A at first, causing S to believe that A and C wish to establish a protocol session.
(3) S then outputs (S, S_A^*) and (S, S_C^*) to A and C, respectively.
(4) The rest of the steps proceed in a straightforward manner, but C will take the place of B to interact with S.
(5) Afterward, A sends (A, V_{A0}, V_{A1}, V_{A2}, Z_A) to S. At the same time, C selects random numbers c_0, c_1, $c_2 \in Z_q^*$, compute $V_{C0} = c_0P$, $V_{C1} = c_1P$, $V_{C2} = c_2P$ and $S_C = S_C^* - Y_C$, $V_{CS} = c_0S_C$. Let T_{CS}, T_{C1} and T_{C2} be the x-coordinate of V_{CS}, V_{C1}, and V_{C2}, respectively. Then, C compute Z_C by the following equation: $Z_C = T_{CS}X_C - c_1T_{C1} - c_2T_{C2}$. C then sends (C, V_{C0}, V_{C1}, V_{C2}, Z_C) to S.
(6) S then outputs (S, V_{C1}, V_{C2}, V_{SA}, Q_A) and (S, V_{A1}, V_{A2}, V_{SC}, Q_C) to A and C, respectively.
(7) After receiving (S, V_{C1}, V_{C2}, V_{SA}, Q_A), A computes the secret key K_{Ai} (i = 1, 2, 3, 4).
(8) After receiving (S, V_{A1}, V_{A2}, V_{SC}, Q_C), C computes the secret key K_{Ci} (i = 1, 2, 3, 4).

In the above attack, a malicious user C impersonates B to respond to A when A requests to initiate an instance of the protocol with B. And A ends up thinking it is sharing a key with B when it is actually sharing with C and C knows what this key $K_A = K_C$ is. Meanwhile, B need not be present at all. Through the attack, the authentication mechanism of the protocol is completely compromised. The attack also demonstrates that, when moving from two parties to three parties, the existence of malicious legitimate users needs to be taken into consideration [1, 2, 18].

Finally, we point out that the key derivation phase is deliberately omitted in Li *et al.*'s protocol. Key derivation refers to the process by which an agreed upon large random number, often named master secret, is used to derive session keys to encrypt and authenticate data. As a result, a adversary can obtain some information about the session key although a adversary is unable to obtain the whole key. More specifically, it can reliably distinguish between the session key K and a randomly chosen string of the expected length simply by checking if $e(V_{A1}, V_{B1}) = e(K, P)$ holds or not. In some sense, this is another weakness of the protocol. Indeed, the key derivation phase is a crucial step for theoretical reasons, but also practical purpose, and can not be omitted.

4. DIRECT EC ANALOG OF KWON *ET AL.*'S PROTOCOL AND ITS WEAKNESS

This section describes the direct EC analog of the DL based protocol proposed by Kwon *et al*. [8] and then shows it is susceptible to an off-line password guessing attack. There is no difference with the original version of the protocol in [8] except that the operation of the represented group is not denoted multiplicatively but additively. The original version of the DL based protocol is referred to [8].

4.1 Protocol Description

And the protocol is also divided into two phases: the initialization phase and the au-

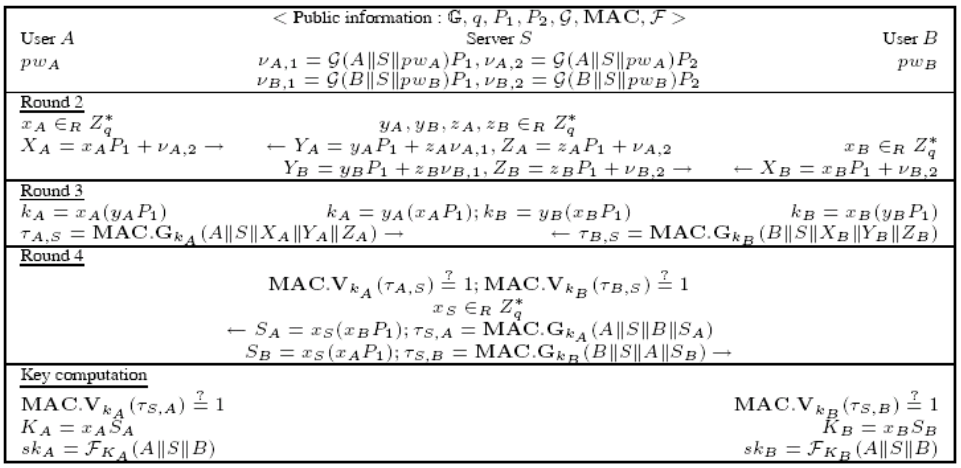

Fig. 1. Direct EC analog of Kwon *et al.*'s protocol.

thenticated key exchange phase. In the initialization phase, it assumes that each user I must register to the server S so they share the verifiers $\nu_{A,1} = \mathcal{G}(I\|S\|pw_I)P_1$, $\nu_{A,2} = \mathcal{G}(I\|S\|pw_I)P_2$ for password pw_I and the public information, where $\mathcal{G}$: $\{0, 1\}^* \rightarrow Z_q^*$ is a hash function, P_1 and P_2, are generators of $\mathbb{G}$ and must be generated so that their discrete logarithmic relation cannot be known. In the authenticated key exchange phase, A and B authenticate each other with S's help, then A and B can share a common session key. This phase is divided into four rounds, which are illustrated as in Fig. 1. In the figure, $\mathcal{F}_\sigma$ denotes a pseudo-random function (PRF) that takes an element $\sigma \in \mathbb{G}$ as its seed. In addition, **MAC** denotes a secure message authentication code (MAC) algorithm consisting of three algorithm, (**KEY.G**, **MAC.G**, **MAC.V**), where **KEY.G** generates a key k_{mac}; given k_{mac}, **MAC.G** computes a tag τ = **MAC.G**$_{k_{mac}}(M)$ for a message M; **MAC.V** verifies a message-tag pair using key k_{mac}, and returns 1 if the tag is valid or 0 otherwise. A more detailed description of the protocol follows. Here, we just follows the description in [8].

Round 1: The initiator A broadcasts (A, B, S).

Round 2:

(1) A chooses a random number $x_A \in_R Z_q^*$, computes $X_A = x_A P_1 + \nu_{A,2}$, and sends ($A$, X_A) to S.

(2) B analogously computes $X_B = x_B P_1 + \nu_{B,2}$, and sends ($B$, X_B) to S.

(3) S selects random numbers y_A, y_B, z_A, $z_B \in_R Z_q^*$, computes $Y_A = y_A P_1 + z_A \nu_{A,1}$, $Z_A = z_A P_1 + \nu_{A,2}$, and sends ($S$, Y_A, Z_A) to A. (Analogously S sends (S, Y_B, Z_B) to B.)

Round 3:

(1) Upon receiving (S, Y_A, Z_A), A computes $T_A = \mathcal{G}(A\|S\|pw_A)(Z_A - \nu_{A,2})$ and $k_A = x_A(Y_A - T_A)$.

(2) Upon receiving (S, Y_B, Z_B), B analogously computes $k_B = x_B(Y_B - T_B)$.

(3) Upon receiving (A, X_A), S computes $k_A = y_A(X_A - \nu_{A,2})$ (Analogously S computes $k_B = y_B(X_B - \nu_{B,2})$.)

(4) A computes $\tau_{A,S}$ = **MAC.G**$_{k_A}(A\|S\|X_A\|Y_A\|Z_A)$ and sends it to S. (B analogously computes $\tau_{B,S}$ and sends it to S.)

Round 4:

(1) Upon receiving $\tau_{A,S}$, S computes **MAC.V**$_{k_A}(\tau_{A,S})$. S terminates if **MAC.V** returns 0 or

moves to the next phase otherwise. (S analogously checks the validity of $\tau_{B,S}$ using k_B.)
(2) S selects a random number $x_S \in_R Z_q^*$, computes $S_A = x_S(X_B - \nu_{B,2})$ and $\tau_{S,A} = \mathbf{MAC.G}_{k_A}(A||S||B||S_A)$, and sends $(S_A, \tau_{S,A})$ to A.
(3) S computes $S_B = x_S(X_A - \nu_{A,2})$ and $\tau_{S,B} = \mathbf{MAC.G}_{k_B}(B||S||A||S_B)$, and sends $(S_B, \tau_{S,B})$ to B.

Key computation: Upon receiving $(S_A, \tau_{S,A})$, A terminates if $\mathbf{MAC.V}_{k_A}(\tau_{S,A})$ returns 0 or computes $K_A = x_A S_A$ and the session key $sk_A = \mathcal{F}_{K_A}(A||S||B)$ otherwise. (B analogously computes $K_B = x_B S_B$ and $sk_B = \mathcal{F}_{K_B}(A||S||B)$.

The correctness of the protocol follows from the fact that, in an honest execution of the protocol, $K_A = x_A x_B x_S P_1 = K_B$.

4.2 Security Weakness

Unfortunately, the direct EC analog of the DL based protocol proposed by Kwon *et al.* [8] described above is completely insecure. In this section, we will show it is vulnerable to an off-line password guessing attack. Please note the DL based protocol proposed by Kwon *et al.* [8] is still secure.

In the protocol described above, since all transcripts are transmitted over an open network, a benign (passive) adversary, can easily obtain a valid information tuple (X_A, S_A, X_B, S_B) such that $X_A = x_A P_1 + \nu_{A,2}$, $S_A = x_S x_B P_1$, $X_B = x_B P_1 + \nu_{B,2}$ and $S_B = x_S x_A P_1$. The adversary can guess two passwords pw_A^* and pw_B^* from its space $\mathcal{D}$ and derive the corresponding $\nu_{A,2}^*$ and $\nu_{B,2}^*$, then verify them by checking $e(X_A - \nu_{A,2}^*, S_A) \stackrel{?}{=} e(X_B - \nu_{B,2}^*, S_B)$. If it does hold, the adversary has guessed the correct secret passwords $pw_A^* = pw_A$ and $pw_B^* = pw_B$. Otherwise, the adversary repeatedly guesses two new passwords pw_A^* and pw_B^* from $\mathcal{D}$ until $e(X_A - \nu_{A,2}^*, S_A) \stackrel{?}{=} e(X_B - \nu_{B,2}^*, S_B)$ holds. In the attack, if the adversary is a legitimate user, say A and thus knows pw_A, it can guess pw_B much more efficiently. Therefore, the direct EC analog of Kwon *et al.*'s protocol is still vulnerable to such an offline password guessing attack.

Through the above attack, we hope show that direct EC analogs of DL based protocols may be susceptible to some new attacks. Due to it, it is important to study the precise adaptation of DL-based password authenticated protocols.

5. ENHANCED VERIFIER-BASED 3PAKE PROTOCOL USING ELLIPTIC CURVES

To avoid the off-line password guessing attack, in this section, we present an enhanced verifier-based 3PAKE protocol using EC. Thereafter, we present efficiency and security analysis for the scheme.

5.1 Description

Our protocol, as shown in Fig. 2, will also consist of four rounds. In our protocol, $\mathcal{H}$: $\{0, 1\}^* \rightarrow \{0, 1\}^l$ denotes a hash function, where l is a secure parameter.

There is a slight difference between the protocol and the previous one. In the current protocol, both $x_A P_1$ and $x_B P_1$, *i.e.* X_A and X_B, are sent in plain to S in round 2 and then are

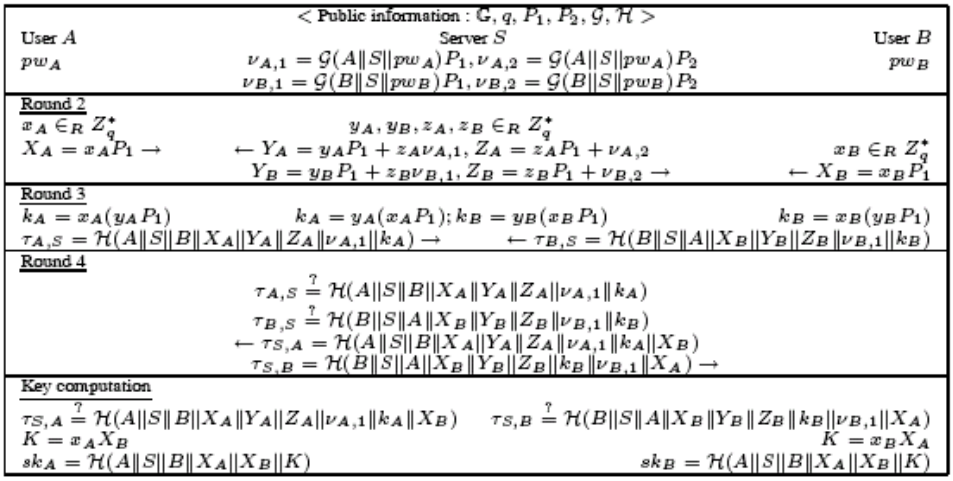

Fig. 2. Our enhanced verifier-based 3PAKE protocol.

forwarded to *B* and *A* respectively in an authenticated way in round 4. In the end, *A* and *B* will compute $K_A = x_A X_B$ and $K_B = x_B X_A$ respectively and derive a session key from that value. As a result, the relation for helping guess the password in section 4 is not available to the intruder. We can safely do so just because each user has to authenticate itself to *S* shortly after it receives the challenge from *S* and *S* will not forward X_A and X_B to *B* and *A* respectively along with its authenticator until it confirms that both users are valid. From the security view point, it is essential that *S* should wait for having received the two authenticators and check that both are valid before it communicates its own authenticator. Otherwise, an adversary could mount an off-line dictionary attack: assume that $\mathcal{A}$ tries to impersonate *A*, and thus sends $X_A = x_A P_1$. Knowing x_A, the authenticator sent back by *S* is derived from $k_A = x_A[Y_A - \mathcal{G}(A\|S\|pw_A)(Z_A - \nu_{A,2})]$, a value easily tested by the adversary, for all the passwords.

In addition, all the authenticators as well as the session key are simply computed via the hash function $\mathcal{H}$, instead of using extra message authentication scheme and pseudo-random function. Moreover, the password verifier is included as partial input of the hash function. The modification helps to achieve provable security for our scheme. Later, we will prove our scheme is a secure 3PAKE protocol.

5.2 Performance

Our protocol is efficient. One can easily remark that the communication cost remains unchanged. And the details of comparisons in computation cost between our protocol and the EC analogue of Kwon *et al.*'s protocol are shown in Table 1. Note that we only count the number of point multiplication, which entails the highest computational complexity, and neglect the computational complexity of all other operations such as Hash computation, which can be done efficiently. In one run of the enhanced protocol, the server performs two less point multiplications of elliptic curve. To be the most important, our protocol can be securely implemented over EC. In a low resource environment, the natural choice for cryptographic protocols would be EC implementation because of the well-

Table 1. Efficiency comparisons.

Schemes	Computation cost	
	User	Server
Kwon's scheme [8]	5	10
Our sheme	5	8

known advantages with regard to processing and size constraints. Therefore, the protocol is quite popular in low resource environments because of the remarkable efficiency. We should note the DL analogue of our protocols is also secure. And our design is easily understood and still efficient when compared to Kwon's solution in [8].

5.3 Security

In this section, we show that our protocol is secure in the random-oracle model (an idealized view of hash functions), starting with the formal security models and some algorithm assumption that will be used in our proof.

5.3.1 Security model for three-party password-based key exchange

In this section, we introduce the formal security models which will be used in next section when we show that our protocol is secure in the random-oracle model. The model builds upon the previous one presented in [1, 2]. In our model, we add one more oracle – *Corrupt* oracle so that the adversary capabilities in a real attack can be modelled better. Due to the omission of the *Corrupt* query in their model, the protocol proposed by Abdalla and Pointcheval in [2] was found insecure in [19] even if it was provably secure in their model.

We first introduce some definitions as follows,

Protocol Participants: Each participant in a 3-party password-based key exchange is either a client (User) $U \in \mathcal{U}$ or a trusted server $S \in \mathcal{S}$. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of the protocol. We denote U (resp. S) instances by U^i (resp. S^j).
Long-Lived Keys: Each participant $U \in \mathcal{U}$ holds a password pw_U. Each server $S \in \mathcal{S}$ holds a vector $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$ with an entry for each client, where $pw_S[U]$ is the transformed password, following the definition in [20].
Partner: An instances is said to be partner of another instance if it has accepted with the same session identifier SID as the latter's, where SID is defined as the concatenation of all messages an instance has sent and received.

The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack (see literature for more details [20, 21].) The types of oracles available to the adversary are as follows,

- *Execute*($U_1^{i_1}$, S^j, $U_2^{i_2}$): This query models passive attacks in which the attacker eavesdrops on honest executions among the client instances $U_1^{i_1}$ and $U_2^{i_2}$ and trusted server instance S^j. The output of this query consists of the messages that were exchanged dur-

ing the honest execution of the protocol.

- *SendClient*(U^i, m): This query models an active attack, in which the adversary may intercept a message and then modify it, create a new one, or simply forward it to the intended client. The output of this query is the message that client instance U^i would generate upon receipt of message m.
- *SendServer*(S^j, m): This query models an active attack against a server. It outputs the message that server instance S^j would generate upon receipt of message m.
- *Reveal*(U^i): If a session key is not defined for instance U^i or if a *Test* query (to be introduced later) was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.
- *Corrupt*(U): This query returns to the adversary the long-lived key pw_U for participant U. As in [20], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary.

In order to define a notion of security for the key exchange protocol, we consider a game in which the protocol $\mathcal{P}$ is executed in the presence of the adversary $\mathcal{A}$. In this game, we first draw a password pw from a dictionary $\mathcal{D}$, provide coin tosses and oracles to $\mathcal{A}$, and then run the adversary, letting it ask any number of queries as described above, in any order.

Forward Security: In order to model the forward secrecy (FS) of the session key, we consider a game $Game^{fs}(\mathcal{A}, \mathcal{P})$, in which one additional oracle is available to the adversary: the *Test*(U^i): oracle.

- *Test*(U^i): This query tries to capture the adversary's ability to tell apart a real session key from a random one. In order to answer it, we first flip a (private) coin b and then forward to the adversary either the session key sk held by U^i (*i.e.*, the value that a query *Reveal*(U^i) would output) if $b = 1$ or a random key of the same size if $b = 0$.

The *Test*-oracle can be queried at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the attacked instance U^i is FS-Fresh, which is defined to avoid cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key sk is FS-Fresh if all of the following hold: (1) the instance holding sk has accepted, (2) no *Corrupt*-query on the related clients has been asked since the beginning of the game; and (3) no *Reveal*-query has been asked to the instance holding sk or to its partner. In other words, the adversary can only ask *Test*-queries to instances which had accepted before the *Corrupt* query on the related clients is asked. Let **Succ** denote the event in which the adversary successfully guesses the hidden bit b used by *Test* oracle. The FS-advantage of an adversary $\mathcal{A}$ is then defined as $Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) = 2Pr[\mathbf{Succ}] - 1$, when passwords are drawn from a dictionary $\mathcal{D}$. The protocol $\mathcal{P}$ is said to be (t, ε)-FS-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running with time t. The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the games defining the security plus the code size [22].

To prevent dictionary attack, ε is usually required to be $O(n_{active}/|\mathcal{D}|) + \epsilon(l)$ for password-based protocols, where $|\mathcal{D}|$ is the size of the dictionary $\mathcal{D}$, n_{active} is the number of active attempts and $\epsilon(l)$ is a negligible function depending on the security parameter l.

5.3.2 Diffie-Hellman assumptions

In this subsection, we recall the EC computational Diffie-Hellman (CDH) assumption upon which the security of our protocol is based upon. In the CDH assumption, given $X = x \cdot P$ and $Y = y \cdot P$, where x and y are drawn randomly from Z_q^*, it is computationally infeasible to compute $xy \cdot P$.

5.3.3 Security proof

As the following theorem states, our 3PAKE is a forward-secure password-based key exchange protocol as long as the hash function closely behaves like a random oracle and the CDH problem is hard in $\mathbb{G}$. The specification of this protocol is found on Fig. 2.

Theorem 1 Let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let $\mathcal{P}$ describe the 3-party password-based authenticated key exchange protocol associated with these primitives as defined in Fig. 2. Then, $Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ is less than $O(q_s/|\mathcal{D}|) + \epsilon(l)$ under the assumptions that the hash function closely behaves like a random oracle and that the CDH assumption holds in $\mathbb{G}$, where q_s represents the number of *Send*-queries.

***Proof*:** For an easier analysis, we first exclude some unlikely evens in the game: *i.e.*, collisions on the partial transcripts (X_A, Y_A, Z_A) or (X_B, Y_B, Z_B) or on hash values. We can safely do so because the probability that such evens appear is negligible. At this moment, the sessions in the game can be split in three disjoint sub-cases:

Case A: Both X_A (resp. X_B) and (Y_A, Z_A) (resp. (Y_B, Z_B)) have been generated by a real instance of A (resp. B) and S respectively; or both X_A and X_B have been generated by A and B respectively. In the former case, the adversary can not know the secrets x_A (resp. x_B) or y_A (resp. y_B) because they are chosen by the client and server respectively. And thus she can not compute k_A (resp. k_B) based on the CDH assumption. With the same analysis, we can have similar results for the latter case. As a result, she can not validate the guessed password according to the received authenticators as well as session keys (returned by *Reveal*-query). That is to say, these executions provide no useful information to the adversary at all.

Case B: Case A did not happen and (Y_A, Z_A) (resp. (Y_B, Z_B)) has been simulated but X_A (resp. X_B) has been produced by the adversary. In this case, we just need to consider those sessions accepted before the corruption. In order to make a session accepted, the adversary has to send a correct $\tau_{A,S}$ (resp. $\tau_{B,S}$). Without collusion on hash function, each authenticator sent by the adversary has been computed with at most one pw_A (resp. pw_B) value. Thus we have: $Pr[\text{Case B}] \leq \frac{q_s}{|\mathcal{D}|}$.

Case C: Case A did not happen and X_A (resp. X_B) has been simulated but (Y_A, Z_A) (resp. (Y_B, Z_B)) has been produced by the adversary. If $\mathcal{A}$ has guessed A (resp. B)'s correct password when he sends (Y_A, Z_A) (resp. (Y_B, Z_B)) to A (resp. B), she may generate a valid authenticator $\tau_{S,A}$ (resp. $\tau_{S,B}$) upon receiving X_A (resp. X_B) from the user. But the probability is less than $\frac{q_s}{|\mathcal{D}|}$. On the other hand, if the adversary has not guessed A (resp.

B)'s correct password when she sends (Y_A, Z_A) (resp. (Y_B, Z_B)), she will not be able to know such y'_A (resp. y'_B) $\in Z_q$ that $y'_A P_1 = t_A(Z_A - t_A P_2)$ (resp. $y'_B P_1 = t_B(Z_A - t_B P_2)$) according to the results in [23], where $t_U = \mathcal{G}(U||S||pw_U)$. With no knowledge of x_A (resp. x_B) or y'_A (resp. y'_B), she can not compute k_A (resp. k_B) based on the CDH assumption. As a result, she can neither compute $\tau_{S,A}$ (resp. $\tau_{S,B}$) nor validate other possible values of the user's password according to the received value $\tau_{A,S}$ (resp. $\tau_{B,S}$). Thus we have: Pr [Case C] $\leq O(\frac{q_s}{|D|}) + \epsilon(l)$ based on the CDH assumption.

As a consequence, one gets the announced result. ❑

According to Theorem 1, our protocol can prevent off-line dictionary attacks and guarantee the forward privacy of session keys. Finally, we should note that, since mutual between server and two clients is provided, each party in our scheme can naturally detect failure of a malicious trial. In other words, our scheme can resist *undetectable online dictionary attack* [3].

6. CONCLUSION

In this paper, we have shown that the verifier-based 3PAKE protocol recently proposed by Li *et al.* [9] is still vulnerable to an off-line dictionary attack and an unknown key-share attack. Furthermore, we also have shown that the direct EC analog of the DL based protocol proposed by Kwon *et al.* [8] can't resist the off-line password guessing attack. Finally, we have presented an enhanced verifier-based 3PAKE protocol that can be securely implemented over elliptic curves. And yet, our proposal is simple and efficient. Therefore, the protocol is quite popular in low resource environments because of the remarkable efficiency. In addition, the attacks described in this paper also has highlighted direct EC analogs of DL based protocols may be susceptible to some new attacks.

REFERENCES

1. M. Abdalla, P. A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography*, LNCS 3386, 2005, pp. 65-84.
2. M. Abdalla and D. Pointcheval, "Interactive Diffie-Hellman assumptions with applications to password-based authentication," in *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, LNCS 3570, 2005, pp. 341-356.
3. Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *ACM Operating Systems Review*, Vol. 29, 1995, pp. 77-86.
4. C. L. Lin, H. M. Sun, M. Steiner, and T. Hwang, "Three-party encrypted key exchange without server public-keys," *IEEE Communication Letters*, Vol. 5, 2001, pp. 497-499.
5. C. C. Chang and Y. F. Chang, "A novel three-party encrypted key exchange proto-

col," *Computer Standards and Interfaces*, Vol. 26, 2004, pp. 471-476.
6. H. M. Sun, B. C. Chen, and T. Hwang, "Secure key agreement protocols for three-party against guessing attacks," *Journal of Systems and Software*, Vol. 75, 2005, pp. 63-68.
7. S. W. Lee, H. S. Kim, and K. Y. Yoo, "Efficient verifier-based key agreement protocol for three parties without server's public key," *Applied Mathematics and Computation*, Vol. 167, 2005, pp. 996-1003.
8. J. O. Kwon, I. R. Jeong, K. Sakurai, and D. H. Lee, "Efficient verifier based password-authenticated key exchange in the three-party setting," *Computer Standards and Interfaces*, Vol. 29, 2007, pp. 513-520.
9. W. M. Li and Q. Y. Wen, "Efficient verifier-based password-authentication key exchange protocol via elliptic curves," in *Proceedings of International Conference on Computer Science Software Engineering*, 2008, pp. 1003-1006.
10. W. M. Li, Q. Y. Wen, and H. Zhang, "Verifier-based password-authenticated key exchange protocol for three-party (In Chinese)," *TongxinXuebao/Journal on Communication*, Vol. 29, 2008, pp. 149-152.
11. S. W. Lee, H. S. Kim, and K. Y. Yoo, "Improvement of Lee and Lee's authenticated key agreement scheme," *Applied Mathematics and Computation*, Vol. 162, 2005, pp. 1049-1053.
12. J. O. Kwon, J. Y. Hwang, C. W. Kim, and D. H. Lee, "Cryptanalysis of Lee-Kim-Yoo password-based key agreement schemee," *Applied Mathmatics and Computation*, Vol. 168, 2005, pp. 858-865.
13. D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York, 2004.
14. N. Koblitz, "Elliptic curve cryptosystem," *Mathematics of Computation*, Vol. 48, 1987, pp. 203-209.
15. R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press, Cambridge, 1996.
16. I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, 1999.
17. P. S. L. M. Barreto, H. Y. Kim, and B. Lynn, "Efficient algorithms for pairing-based cryptosystems," in *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, LNCS 2442, 2002, pp. 354-369.
18. J. Katz and J. S. Shin, "Modeling insider attacks on group-key exchange protocols," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, pp. 180-189.
19. K. K. R. Choo, C. Boyd, and Y. Hitchcock, "Examining indistinguishability-based proof models for key establishment protocols," *Advances in Cryptology – ASIACRYPT*, LNCS 3788, 2005, pp. 585-604.
20. M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, LNCS 1807, 2000, pp. 139-155.
21. E. Bresson, O. Chevassut, and D. Pointcheval, "New security results on encrypted key exchange," in *Proceedings of the 7th International Workshop on Theory and Practice*

in Public Key Cryptography, LNCS 2947, 2004, pp. 145-158.
22. M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," in *Proceedings of the Cryptographers' Track at RSA Conference*, LNCS 2020, 2001, pp. 143-158.
23. M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Proceedings of the Cryptographers' Track at the RSA Conference*, LNCS 3376, 2005 pp. 191-208.

Shuhua Wu (吳樹華) is a lecturer of Networks Engineering Department, Information Science Technology Institute, Zhengzhou, China. His research interests include cryptology and communication protocols.