Arab J Sci Eng (2013) 38:2069–2077
DOI 10.1007/s13369-013-0543-z

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

ECC-Based Password-Authenticated Key Exchange in the Three-Party Setting

Tingting Liu · Qiong Pu · Yong Zhao · Shuhua Wu

Received: 10 May 2011 / Accepted: 29 September 2011 / Published online: 9 February 2013

Abstract This paper investigates three-party password authenticated key exchange protocols using elliptic curve cryptosystem (ECC). We first show that the direct elliptic curve analog of Chien's protocol proposed most recently is vulnerable to off-line dictionary attack. Thereafter, we present an enhanced protocol based on ECC. Our proposal can defeat password-guessing attacks and the stolen-verifier attacks. And yet, it is also efficient. Furthermore, we can provide the rigorous proof of the security for it. Therefore, the protocol is quite popular in low resource environments.

Keywords Password-based · Authenticated key exchange · Three-party · Dictionary attack · Elliptic curve

الخلاصة

تبحث هذه الورقة العلمية في بروتوكولات تغيّر المفتاح المصرحة بكلمة سر وذات ثلاثة أطراف باستخدام منحنى تشفير بيضوي (ECC). وقد أظهرنا أولا أن تناظر منحنى (EC) البيضوي المباشر لبروتوكول شينز المقترح حديثا هو غير حصين لهجوم القاموس خارج الخط. وعرضنا بعد ذلك بروتوكولا معززا معتمدا على ECC. ويستطيع البروتوكول المقترح من طرفنا أن يهزم هجومات تخمين كلمة السر وهجومات السرقة / التحقق. وهو فعال إلى الآن. وأكثر من ذلك نستطيع أن نعطي الإثبات الدقيق من الأمان له. لذلك فإن هذا البروتوكول معروف تماما في بيئات المصدر المنخفض.

T. Liu (✉) · S. Wu
Zhengzhou Information Science and Technology Institute, Zhengzhou, China
e-mail: ann320120@yahoo.com.cn

Q. Pu
CIMS Research Center, Tongji University, Shanghai, China

Y. Zhao
College of Computer Science, Beijing University of Technology, Beijing, China

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 (i.e., password-based authentication), and to generate a fresh symmetric key securely such that it is known only to the two parties (i.e., 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.

In practices, most of these proposed PAKE protocols are presented in the context that the two involved entities are client and server, respectively, e.g. [1–9], which are simply called 2PAKE protocols. However, there is a common problem in these 2PAKE protocols. That is, two communication parties need to previously share a password for the mutual authentication and a session key exchange. To apply 2PAKE protocols to a large scale peer-to-peer system, each pair of communication parties in a group needs to pre-share a password. This restriction causes that each user has to remember a large number of passwords for communicating with a group of users. To solve this problem, various three-party password-based authenticated key exchange (3PAKE) protocols were proposed [10–20] during the last few years, 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. However, some of them (e.g. [10–12, 14]) were already broken according to the results in [21–24], respectively. Moreover, most of them are for only RSA (e.g. [16]) or Discrete Logarithm (DL) (e.g. [10–14,17–19]), and only a few concrete Elliptic Curve (EC)-based 3PAKE protocols (e.g. [15,20]) have been proposed in the literature. In a low resource environment, the natural choice cryptographic protocols would be an EC implementation. This is due to the low computation and storage costs of EC-based protocols. It seems that most authors have presumed that the adaption of DL-based protocols to the EC environment is straightforward.

In this paper, we show that the direct EC analog of the DL-based protocol proposed most recently either by Huang [14] or Chang et al. [18] or Chien [19] or Liang et al. [30] is completely insecure. And the EC-based scheme in [15] is not efficient, because it involves costly Weil Pairing computation. We first provide an attack to illustrate that the direct EC analog of the protocol in [19] cannot resist the off-line password guessing attack. And the same attack can not only be applicable to the EC analog of the DL-based protocols in [14,18], but also the scheme based on ECC proposed by Lou and Huang [20] most recently. Thereafter, we present an enhanced 3PAKE protocol using ECC. Our proposal can defeat password-guessing attacks and the stolen-verifier attacks. And yet, it is also efficient. And yet, it is efficient. Furthermore, we can provide the rigorous proof of the security for it. Therefore, the protocol is quite popular in low resource environments because of the remarkable efficiency.

The remainder of this paper is organized as follows. Section 2 briefly reviews Chien's three-party password-based authenticated protocol and then reveals its weakness. Section 3 presents an enhanced 3PAKE protocol along with some important remarks. Section 4 provides the rigorous proof of the security for our protocol. Finally, conclusion is presented in Sect. 5.

2 Review of Chien's Protocol

This section describes the direct EC analog of the DL-based 3PAKE protocol proposed by Chien [19] 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 [19] 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 [19].

2.1 Notations

The notations used in their scheme are described as the following:

- A, B : the two clients want to establish authenticated session keys.
- S : a trusted server.
- $pw_A(pw_B)$: the password of user A (resp. B).
- E : An elliptic curve defined over a finite field F_q(q is a large prime number ($q \approx 2^{160}$)) [25,26];
- P : A point in E with large order n, where n is a large prime;
- $\mathbb{G}$: the cyclic addition group generated by P;
- kP : the point multiplication defined as $kP = \underbrace{P + P + \cdots + P}_{\text{ktimes}}$, where k is an integer in Z_n.
- h, h^t: two secure one way hash functions h, h^t:$\{0,1\}^* \to \{0,1\}^l$, where l is system security parameter.

In an elliptic curve cryptosystem (ECC), the elliptic curve equation is usually defined as the form of $E : y^2 = x^3 + ax^2 + bx + c(\text{mod}q)$ over a prime finite field F_q, where $a, b, c \in F_q$, $q > 3$, and $27c^2 + 4a^3c + 4b^3 - a^2b^2 - 8abc \neq 0(\text{mod}q)$ [25]. We look at the points on E with coordinates in F_q. Then, the points on E together with the extra point living "at infinity" $\mathcal{O}$, which we denote by $E_q = \{(x, y) : x, y \in F_a \text{ satisfy } y^2 = x^3 + ax^2 + bx + c(\text{mod}q)\} \cup \{\mathcal{O}\}$, obey the elliptic curve addition algorithm and thus form an additive group. In general, the group $\mathbb{G}$ is a subgroup of E_q. In view of shortness, we omit the details and refer to [25,26].

2.2 Protocol Description

There are three entities involved in the protocol: the authentication server S, and two users A (initiator) and B (responder). A and B authenticate each other with S's help, then A and B can share a common session key sk_{AB}. Each user $A(B)$ has a password pw_A (resp. pw_B) and the server only stores its verifier $V_A = h(A \parallel S \parallel pw_A)P$ (resp. $V_B = h(B \parallel S \parallel pw_B)P$) in its database. As illustrated on Fig. 1, "sid" denotes the session identifier used to uniquely identify a session from other sessions. The details will be described in the following steps. Here, we just follows the description in [19].

- Step 1. $A \to S : sid, A, B$ A sends the request including the two parties' (A and B) identities to S.
- Step 2. $S \to A : sid, A, B, S_1, S_2, S_3$ S chooses three random numbers z_1, z_2 and z_3, then computes $S_1 = z_3V_A \oplus z_1P$, $S_2 = z_3V_B \oplus z_2P$ and $S_3 = z_3P$, and sends the values to A.
- Step 3. $A \to B : sid, A, B, X, M_1, S_2, S_3$ Upon receiving S_3, A, uses pw_A to compute $h(A \parallel S \parallel pw_A)S_3 = z_3V_A$ and then derive $z_1P = S_1 \oplus z_3V_A$, chooses a random integer x_A, and computes one-time strong key $K_{AS} = h(x_Az_1P)$ and $X = x_AP$. It sends $A, B, X, M_1 = h(sid \parallel A \parallel B \parallel X \parallel S_1 \parallel K_{AS})$, S_2, S_3 to B.
- Step 4. $B \to S : sid, A, B, X, Y, M_1, M_2, M_{BA}$ B uses pw_B to compute $h(B \parallel S \parallel pw_B)S_3 = z_3V_B$ and derive $z_2P = S_3 \oplus z_3V_B$, randomly choose an integer y_B, computes $K_{BS} = h(y_Bz_2P)$, $K_{AB} = y_BX$, $Y =$

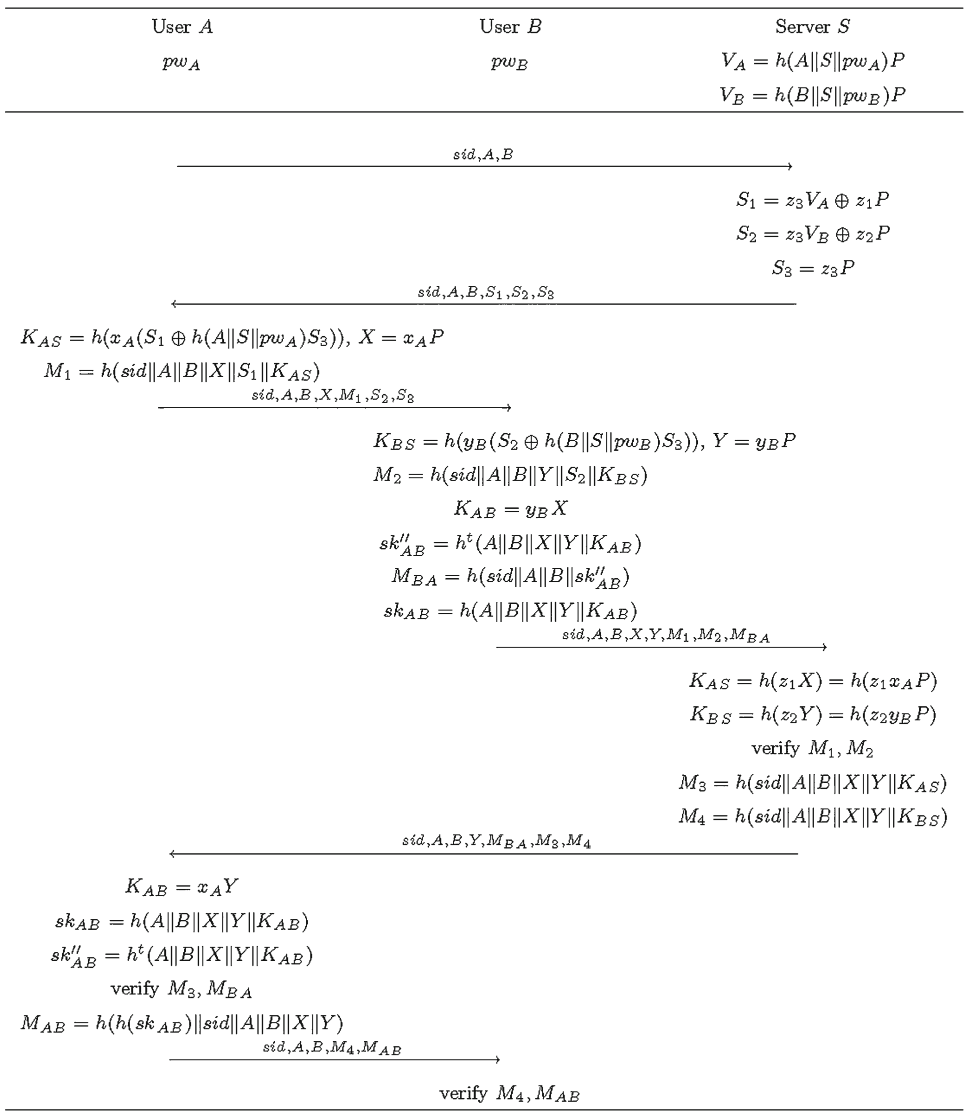
User A
User B
Server S
pw_A
pw_B
$V_A = h(A\|S\|pw_A)P$
$V_B = h(B\|S\|pw_B)P$
sid, A, B
$S_1 = z_3V_A \oplus z_1P$
$S_2 = z_3V_B \oplus z_2P$
$S_3 = z_3P$
sid, A, B, S_1, S_2, S_3
$K_{AS} = h(x_A(S_1 \oplus h(A\|S\|pw_A)S_3)),\ X = x_AP$
$M_1 = h(sid\|A\|B\|X\|S_1\|K_{AS})$
$sid, A, B, X, M_1, S_2, S_3$
$K_{BS} = h(y_B(S_2 \oplus h(B\|S\|pw_B)S_3)),\ Y = y_BP$
$M_2 = h(sid\|A\|B\|Y\|S_2\|K_{BS})$
$K_{AB} = y_BX$
$sk''_{AB} = h^t(A\|B\|X\|Y\|K_{AB})$
$M_{BA} = h(sid\|A\|B\|sk''_{AB})$
$sk_{AB} = h(A\|B\|X\|Y\|K_{AB})$
$sid, A, B, X, Y, M_1, M_2, M_{BA}$
$K_{AS} = h(z_1X) = h(z_1x_AP)$
$K_{BS} = h(z_2Y) = h(z_2y_BP)$
verify M_1, M_2
$M_3 = h(sid\|A\|B\|X\|Y\|K_{AS})$
$M_4 = h(sid\|A\|B\|X\|Y\|K_{BS})$
$sid, A, B, Y, M_{BA}, M_3, M_4$
$K_{AB} = x_AY$
$sk_{AB} = h(A\|B\|X\|Y\|K_{AB})$
$sk''_{AB} = h^t(A\|B\|X\|Y\|K_{AB})$
verify M_3, M_{BA}
$M_{AB} = h(h(sk_{AB})\|sid\|A\|B\|X\|Y)$
sid, A, B, M_4, M_{AB}
verify M_4, M_{AB}

Fig. 1 Chien's protocol [19]

y_BP, $sk_{AB} = h(A \parallel B \parallel X \parallel Y \parallel K_{AB})$, $sk''_{AB} = h^t(A \parallel B \parallel X \parallel Y \parallel K_{AB})$, $M_{BA} = h(sid \parallel A \parallel B \parallel sk''_{AB})$ and $M_2 = h(sid \parallel A \parallel B \parallel Y \parallel S_2 \parallel K_{BS})$. It sends the data in step 4 to S. sk_{AB} is the final session key.

- Step 5. $S \rightarrow A : sid, A, B, Y, M_{BA}, M_3, M_4$ S authenticates A and B by performing the following operations: (1) computes $K_{AS} = h(z_1X) = h(z_1x_AP)$ and $K_{BS} = h(z_2Y) = h(z_2y_BP)$; (2) uses its local values to verify whether M_1, M_2 are valid. If the verification succeeds, S computes $M_3 = h(sid \parallel A \parallel B \parallel X \parallel Y \parallel K_{AS})$ and $M_4 = h(sid \parallel A \parallel B \parallel X \parallel Y \parallel K_{BS})$, and sends them to A.
- Step 6. $A \rightarrow B : sid, A, B, M_4, M_{AB}$ Upon receiving the data, A first computes $K_{AB} = x_AY, sk_{AB} = h(A \parallel B \parallel X \parallel Y \parallel K_{AB})$ and $sk''_{AB} = h^t(A \parallel B \parallel X \parallel Y \parallel K_{AB})$, and then uses its local values to verify M_3 and verifies whether B's confirmation message M_{BA} is valid. If the verification succeeds, it computes $M_{AB} = h(h(sk''_{AB}) \parallel sid \parallel A \parallel B \parallel X \parallel Y)$ and sends the data in step 6 to B.

Finally, B uses its local values to verify M_4 and verifies whether A's confirmation message M_{AB} is valid. If all the verifications succeed, it accept the session key sk_{AB}.

Please note that the server in Chien's scheme only stores the user's verifier V rather than the user's bare password pw to reduce the security breach once the server is compromised. If a server directly stores users' passwords in its plaintext format, then an attacker who has stolen the password file can directly impersonate the clients. In verifier-based schemes just like Chien's protocol, an attacker, even assuming he has stolen the verifiers but has not yet cracked the corresponding passwords, cannot directly use the verifiers to impersonate the clients. This arrangement would make the attacker more time-consuming to crack the passwords to impersonate the clients and thus give the server's administrator time to react appropriately and to inform its users. However, we should point out that if one client's verifier is stolen, the attacker can still directly use this verifier to impersonate the server to this client in Chien's scheme.

2.3 Weaknesses of Chien's Protocol

Unfortunately, Chien's scheme [19] described above is completely insecure. In this section, we will show that it is vulnerable to off-line password-guessing attacks. Similar attack was first presented in [6].

Off-line password guessing attack succeeds when there is information in communications, which can be used to verify the correctness of the guessed passwords. In Chien's scheme, since all transcripts are transmitted over an open network, a benign (passive) adversary can easily obtain the valid message transcript of S_3, $S_1 = z_3V_A \oplus z_1P = h(A \parallel S \parallel pw_A)S_3 \oplus z_1P$, as well as the the identity A. The adversary can guess a password pw_A^* from the dictionary $\mathcal{D}$ and derive the corresponding point $(x^*, y^*) = z_1P \oplus h(A \parallel S \parallel pw_A)S_3 \oplus h(A \parallel S \parallel pw_A^*)S_3$, then verify it by checking $(y^*)^2 \overset{?}{=} (x^*)^3 + a(x^*)^2 + bx^* + c(\text{mod} q)$, where x^* and y^* are the x-coordinate and y-coordinate of this derived point. Clearly, if pw_A^* is not correct, the computation $S_1 \oplus h(A \parallel S \parallel pw_A^*)S_3$ should result in a random pair (x^*, y^*). In other words, x^*, y^* are two independent random numbers according to the property of $h()$. Even if $x^*, y^* \in F_q$, the probability that the point (x^*, y^*) lies on E is no larger than $\frac{2}{q}$. Typically $|\mathcal{D}|$ is much less than q. Therefore, the adversary should be able to identify the correct password pw_A given one valid message transcript of $\{S_1, S_3\}$ using such a dictionary attack, with a probability of $(1 - \frac{2}{q})^{|\mathcal{D}|-1} \approx 1 - \frac{2(|\mathcal{D}|-1)}{q} \approx 1$. Please note that the attack is a brute-force method in essence, i.e. the attacker tries offline all possible passwords in a given small set of values. Even though such attacks are not very effective in the case of high-entropy keys, they can be very damaging when the secret key is a password since the attacker has a non-negligible chance of winning [9].

Please note that the same attack can also be applicable to the EC analog of the DL-based protocols in [14,18] and the scheme based on ECC proposed by Lou and Huang [20] most recently. And the EC analog of the DL-based protocols in [30] was not secure according to the result in [31].

3 Enhanced Protocol

In this section, we present an enhanced protocol to remedy the security loopholes existing in Chien's protocol [19]. Moreover, we will make some important discussions about our improved protocol at the end of this section.

First, we define some notations used in our scheme. Let $\mathcal{H} : \{0, 1\}^\star \rightarrow \{0, 1\}^l$ be a secure hash function, where l is a security parameter. And let $E_k(\cdot)/D_k(\cdot)$ be a secure symmetric encryption/decryption algorithm [e.g., AES (Advanced Encryption Standard)], where k denotes the symmetric key. In our protocol, we assume that the two users (or clients) A and B willing to establish a common secret session key has passwords pw_A, pw_B, respectively, and the server S only stores its verifier $V_A = E_{\mathcal{H}(s\|A)}(\mathcal{G}(A \parallel S \parallel pw_A))$ (resp. $V_B = E_{\mathcal{H}(s\|B)}(\mathcal{G}(B \parallel S \parallel pw_B)))$ in its database, where $\mathcal{G} : \{0, 1\}^\star \rightarrow \mathbb{G}$ is a secure full-domain hash function (roughly, a surjective hash function) and s is a secret key of S. We also assume that S has a private/public key pair: (s, Q) with $Q = sP$. The public parameters $(\mathcal{H}, \mathcal{G}, \mathbb{G}, Q)$ have been fixed in advance and are known to all parties in the network. The simplified description of the new protocol is given in Fig. 2, where $PW_A = \mathcal{G}(A \parallel S \parallel pw_A)$ and $PW_B = \mathcal{G}(B \parallel S \parallel pw_B)$. Our protocol is based on 2PAKE in [3]. The details will be described in the following steps.

- Step 1. User A chooses two random numbers $r_A, w_A \in Z_n^*$ and computes $R_A = r_AP + PW_A$, $T_A = r_AQ$, and $W_A = w_AP$, where $PW_A = \mathcal{G}(A\|S\|pw_A)$. Then, it computes $\tau_{A,S} = \mathcal{H}(A\|B\|S\|R_A\|W_A\|PW_A\|T_A)$ and sends $(A, R_A, W_A, \tau_{A,S})$ to user B.
- Step 2. Upon receiving $(A, R_A, W_A, \tau_{A,S})$, User B also selects two random numbers $r_B, w_B \in Z_n^*$ and computes $R_B = r_BP + PW_B$, $T_B = r_BQ$, and $W_B = w_BP$. Then it compute $\tau_{B,S} = \mathcal{H}(B\|S\|A\|R_B\|W_B\|PW_B\|T_B)$. Besides, it computes $K = w_BW_A$, and uses this value to compute the common session key $SK = \mathcal{H}(A\|B\|W_A\|W_B\|K)$ and $S_B = \mathcal{H}(SK\|B)$. And it then forwards $(A, R_A, W_A, \tau_{A,S}, B, R_B, W_B, \tau_{B,S}, S_B)$ to S.
- Step 3. Upon receiving $(A, R_A, W_A, \tau_{A,S}, B, R_B, W_B, \tau_{B,S}, S_B)$, the S first computes $PW_A = D_{\mathcal{H}(s\|A)}(V_A)$, $PW_B = D_{\mathcal{H}(s\|B)}(V_B)$, $T_A = s(R_A - PW_A)$, and $T_B = s(R_B - PW_B)$ and then uses them to check $\tau_{A,S}$ and $\tau_{B,S}$, respectively, in a straight way. S detects failure of a malicious trial and thus terminates if either of them is valid or moves to the next phase otherwise. Finally, S computes $\tau_{S,A} = \mathcal{H}(S\|A\|B\|R_A\|W_B\|PW_A\|T_A)$ and $\tau_{S,B} = \mathcal{H}(S\|B\|A\|R_B\|W_A\|PW_B\|T_B)$ and then sends $(\tau_{S,A}, \tau_{S,B}, S_B)$ to user A.
- Step 4. After receiving $(\tau_{S,A}, \tau_{S,B}, S_B)$, user A first checks the validity of $\tau_{S,A}$ using T_A. If that value is invalid, A detects failure of a malicious trial and thus terminates the protocol. Otherwise, it computes $K = w_AW_B$, and uses this value to obtain the common key $SK = \mathcal{H}(A\|B\|W_A\|W_B\|K)$. Then, A also checks whether $S_B = \mathcal{H}(SK\|B)$ holds or not. If it does not hold, A terminates the protocol. Otherwise, A is convinced that SK is a valid session key and accepts it. Then, A computes $S_A = \mathcal{H}(SK\|A)$ and sends $(\tau_{S,B}, S_A)$ to user B.

Fig. 2 Our enhanced protocol

User A	User B	Server S
pw_A	pw_B	$V_A = E_{\mathcal{H}(s\Vert A)}(\mathcal{G}(A\Vert S\Vert pw_A))$ $V_B = E_{\mathcal{H}(s\Vert B)}(\mathcal{G}(B\Vert S\Vert pw_B))$ $(s, Q = sP)$
$r_A, w_A \in Z_n^*$	$r_B, w_B \in Z_n^*$	
1. $PW_A = \mathcal{G}(A\Vert S\Vert pw_A)$ $R_A = r_A P + PW_A$ $T_A = r_A Q, W_A = w_A P$ $\tau_{A,S} = \mathcal{H}(A\Vert B\Vert S\Vert R_A\Vert W_A\Vert PW_A\Vert T_A)$		
$\xrightarrow{A, R_A, W_A, \tau_{A,S}}$	2. $PW_B = \mathcal{G}(B\Vert S\Vert pw_B)$ $R_B = r_B P + PW_B$ $T_B = r_B Q, W_B = w_B P$ $\tau_{B,S} = \mathcal{H}(B\Vert S\Vert A\Vert R_B\Vert W_B\Vert PW_B\Vert T_B)$ $K = w_B W_A$ $SK = \mathcal{H}(A\Vert B\Vert W_A\Vert W_B\Vert K)$ $S_B = \mathcal{H}(SK\Vert B)$	
	$\xrightarrow{A, R_A, W_A, \tau_{A,S}, B, R_B, W_B, \tau_{B,S}, S_B}$	3. $PW_A = D_{\mathcal{H}(s\Vert A)}(V_A)$, $T_A = s(R_A - PW_A)$ $PW_B = D_{\mathcal{H}(s\Vert B)}(V_B)$,$T_B = s(R_B - PW_B)$ check $\tau_{A,S}$ and $\tau_{B,S}$ $\tau_{S,A} = \mathcal{H}(S\Vert A\Vert B\Vert R_A\Vert W_B\Vert PW_A\Vert T_A)$ $\tau_{S,B} = \mathcal{H}(S\Vert B\Vert A\Vert R_B\Vert W_A\Vert PW_B\Vert T_B)$
	$\xleftarrow{\tau_{S,A}, \tau_{S,B}, S_B}$	
4. check $\tau_{S,A}$ $K = w_A W_B$ $SK = \mathcal{H}(A\Vert B\Vert W_A\Vert W_B\Vert K)$ verify: $S_B = \mathcal{H}(SK\Vert B)$ $S_A = \mathcal{H}(SK\Vert A)$ Accept SK		
$\xrightarrow{\tau_{S,B}, S_A}$	5. check $\tau_{S,B}$ verify : $S_A = \mathcal{H}(SK\Vert A)$ Accept SK	

- Step 5. Upon receiving $(\tau_{S,B}, S_A)$, B first checks the validity of $\tau_{S,B}$ using T_B. If that value is invalid, B detects failure of a malicious trial and thus terminates the protocol. Otherwise, it verifies whether $S_A = \mathcal{H}(SK\|A)$ holds or not. If it does not hold, B terminates the protocol. Otherwise, SK is a valid session key. Both the users A and B can use this session key SK for secure communication. Here, SK is only used for one session.

Note that, in our protocol, one principal will invalidate or block the use of a password whenever a certain number of failed attempts occurs.

Rationale for the Scheme. You may wonder why we choose to compute another pair of (W_A, W_B) on the user's side and derive the session key from them rather than just follows the technique that is used in Huang's scheme or Lou's scheme to randomize the pair of $(r_A P, r_B P)$ on the server side and then derive the session key from them directly. In the latter case, the computational cost for the server will become high and it is likely to be bottleneck in the whole system, since the server is aimed at establishing sessions with many clients. In our protocol, we try to reduce the cost for the serve to improve the overall performance. That is, the server can offer its services to a large number of users. The gain in efficiency, however, comes at the cost of more computations at the user side. Besides, it also allows to achieve provable security under some simple assumptions (to be introduced in next section).

One can remark that the server in our our scheme also stores the user's verifier V rather than the user's effective password PW. Even when the adversary has acquire V stored in S. However, without knowing S's secret key s, she cannot impersonate the clients to pass the authentication, as PW is hidden in V using S's secret key s and, thus, the corresponding password PW cannot be revealed. Therefore, the proposed scheme can resist against the stolen-verifier attacks. This is a quite attractive feature because numerous customers' secrets are stored in the server's databases and the server is always the targets of attackers. In addition, if one client's verifier is stolen, the attacker cannot use this verifier to impersonate the server to this client in our scheme.

Finally, it is worth noticing that $\oplus$ is not used now, i.e. $R_A = r_A P + \mathcal{G}(A \parallel S \parallel pw_A)$ $(R_B = r_B P + \mathcal{G}(B \| S \| pw_B))$. As a result, the attacker cannot distinguish feasible and infeasible passwords by testing for subgroup membership any more because the computation $R_A - \mathcal{G}(A \| S \| pw_A^*)$ for any guessed password pw_A^* will always be in $\mathbb{G}$. Therefore, the enhanced protocol can resist against the off-line password guessing attack described in the previous section. In addition, one can easily note that the authenticators (i.e. $\tau_{A,S}$, $\tau_{B,S}$, $\tau_{S,A}$, $\tau_{S,B}$) and the session key (i.e. SK) are computed via the hash function $\mathcal{H}$. The modification is related to our technique of security proof so that we can provide a rigorous proof of security for our protocol. In next section, we will show that it indeed achieves provable security in random oracle model(ideal hash function).

Comparisons with the Related Works. Our protocol is reasonably efficient. The efficiency is measured by the following two aspects:

- Communication Cost: the number of steps during the execution of protocol.
- Computation Cost: the computation complexity of a participant.

In what concerns Computation Cost, 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 and the symmetric encryption/decryption, which can be done efficiently. The details of comparisons between our protocol and the previously proposed efficient schemes so far as I know are shown in Table 1.

As shown in Table 1, in one run of the enhanced protocol, each user performs four exponentiations and the S performs two exponentiations. The computational cost for S of our protocol is comparable to that of Huang's protocol, which is more efficient than other previous schemes [18–20]. Therefore, the server can offer its services to a large number of users. Although our protocol on the user side is not better than Huang's protocol, it is still as efficient as Chien's scheme [19]. As for communication cost, our scheme just requires four communication steps, which is more efficient than all the other schemes. Moreover, our scheme can resist against password-guessing attacks (we will prove it in next section) and the stolen-verifier attacks, while all previous schemes listed in Table 1 are broken by us. Given the better security guarantees, the performance of our scheme may be considered quite reasonable.

4 Security Proof for Our Protocol

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.

4.1 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 [11]. 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 [11] was found insecure in [22] even if it was provably secure in their model. Here, we follow the description in [11].

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 $pw_S =$

Table 1 Comparisons with related works

Schemes	Computation cost		Communication steps	Security properties	
	User	Server		Password-guessing attack	Stolen-verifier attack
Huang's scheme [14][a]	2	2	5	Insecure	Insecure
Lou's scheme [20]	3	4	5	Insecure	Insecure
Chang's scheme [18][a]	3	4	6	Insecure	Insecure
Chien's scheme [19][a]	4	7	6	Insecure	Partial
Our scheme	4	2	4	Secure	Secure

[a] For giving a fair comparison, we assume that the scheme is transferred into EC analogue version

$\langle pw_S[U]\rangle_{U\in\mathcal{U}}$ with an entry for each client, where $pw_S[U]$ is the transformed password, following the definition in [1].

Partner. An instance 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 [1,3].) The types of oracles available to the adversary are as follows:

- $Execute(U_1^{i1}, S^j, U_2^{i2})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the client instances U_1^{i1} and U_2^{i2} and trusted server instance S^j. The output of this query consists of the messages that were exchanged during 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.
- $Reveal(U^i)$: This query models the misuse of the session key by instance U^i (known-key attacks). 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 [1], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary.

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. 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 has been asked to the related clients 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 $Adv^{fs}_{\mathcal{PD}}(A) = 2Pr[\mathbf{Succ}] - 1$, 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 [27].

In password-based scenarios, ε is usually required to be $O(n_{active}/|\mathcal{D}|) + \varepsilon(l)$ for any probabilistic, polynomial time (PPT) adversary $\mathcal{A}$, where $|\mathcal{D}|$ is the size of the dictionary $\mathcal{D}$, n_{active}, is the number of active attempts and $\varepsilon(l)$ is a negligible function depending on the security parameter l. Roughly, a secure password-based protocol guarantees that an exhaustive on-line attack is the "best" possible strategy for an attacker, while off-line dictionary attacks are infeasible.

4.2 Diffie–Hellman Assumptions

In this subsection, we recall the EC computational Diffie–Hellman (ECCDH) assumption and EC decision Diffie–Hellman (ECDDH) assumption, upon which the security of our protocol is based.

In the ECCDH 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 $xy \cdot P$ (denoted by $ECCDH(X, Y)$). In ECDDH assumption, given $X = x \cdot P$, $Y = y \cdot P$, and $Z = z \cdot P$, where x, y, and z are drawn randomly from Z_q^*, it is computationally infeasible to decide whether $xy \cdot P = Z$ or not.

4.3 Security Proof

As the following theorem states, our 3PAKE is a forward-secure password-based key exchange protocol in random oracle model. The specification of this protocol is found on Fig. 2.

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 $O(q_s/|\mathcal{D}|) + \varepsilon(l)$ as long as the hash function closely behaves like a random oracle and that the ECCDH and ECDDH assumptions hold in $\mathbb{G}$, where q_s denotes the number of $Send$-queries (including $SendClient$-queries and $SendServer$-queries).

Proof For an easier analysis, we first exclude some unlikely events in the game: i.e., collisions on the partial transcripts $((R_A, W_A), (R_B, W_B))$ or on hash values. We can safely do so because the probability that such events appear is negligible.

According to the definition of forward security, we just need to consider these sessions accepted before the adversary makes $Corrupt$ query on A or B. At this moment, these sessions in the game can be split into two disjoint sub-cases:

- **Case A:** the item $(R_A, W_A, \tau_{A,S})$ (or $(R_B, W_B, \tau_{B,S})$) is newly produced by the adversary $\mathcal{A}$ and is sent to S by $\mathcal{A}$ via $SendServer$-query and, i.e. $\mathcal{A}$ does not replay to S via $SendServer$-query such an item that were ever generated previously by A (resp. B). If $\mathcal{A}$ has guessed pw_A (resp. pw_B) when she sends R_A (resp. R_B) to the server, she may generate the valid authenticator $\tau_{A,S}$ (resp. $\tau_{B,S}$). But the probability is less than $\frac{q_s}{|\mathcal{D}|}$. On the other hand, if the adversary has not guessed $pw_{A/B}$ when she sends $R_{A/B}$, she will not be able to know such $r'_{A/B} \in Z_n$ that $t'_{A/B}P = R_{A/B} - PW_{A/B}$, where $PW_{A/B} = \mathcal{G}(A/B \| S \| pw_{A/B})$ is computed via the random oracle $\mathcal{G}$ and its discrete logarithm with P as the base is unknown at all. Otherwise, we can use the classical oracle replay technique in [28] to construct an adversary to solve the discrete logarithm problem based on $\mathcal{A}$. It is contradict to hardness of the discrete logarithm problem. With no knowledge of $r'_{A/B}$ or s, she cannot compute $T_{A/B} = ECCDH(r'_{A/B}P, sP)$ based on the ECCDH assumption. As a result, she cannot query $\mathcal{H}$ to compute $\tau_{A/B,S}$. Thus, we have: $Pr[Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A})|CaseA] \le \frac{q_s}{|\mathcal{D}|} + \varepsilon(l)$ based on the ECCDH assumption.

- **Case B:** the item $(W_B, \tau_{S,A})$ (or $(W_A, \tau_{S,B})$) is sent to A (resp. B) by $\mathcal{A}$ via $SendClient$-query. In order to make A (resp. B) accept the session, the adversary has to send a correct $\tau_{S,A/B}$ along with $W_{B/A}$. Please note that the adversary has no idea of $r_{A/B}$ and s since $R_{A/B}$ is generated by A/B and Q is the public key of S. With no knowledge of $r_{A/B}$ or s, she cannot compute $T_{A/B} = ECCDH(r_{A/B}P, sP)$ based on the ECCDH assumption. As a result, she can neither query $\mathcal{H}$ on $(S\|A\|B\|R_A\|W_B\|PW_A\|T_A)$ (resp. $(S\|B\|A\|R_B\|W_A\|PW_B\|T_B)$) to compute $\tau_{S,A/B}$ nor validate any possible values of $pw_{A/B}$ according to the received value $\tau_{A/B,S}$. Thus, we have: $Pr[Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A}|CaseB)] \le \varepsilon(l)$ based on the ECCDH assumption and the ECDDH assumption.

- **Case C:** Both $(W_B, \tau_{S,A})$ and $(W_A, \tau_{S,B})$) have been generated by S and (Z_A, W_A) (resp. (Z_B, W_B)) is not newly generated by the adversary, i.e. it is generated by A (resp. B)(including the item generated previously by the user is relayed to S by the adversary). In this case, the adversary has no idea of $w_{A/B}$ since $W_{A/B}$ is generated by A/B. With no knowledge of $w_{A/B}$, she cannot compute $ECCDH(w_AP, w_AP)$ to derive the session key SK based on the ECCDH assumption. This is also the case even when later the adversary has compromised the user's passwords via $Corrupt$-query. Although the adversary may get the session keys in some executions via $Reveal$-query, it will not give any information about the targeted session key for $Test$-query to the adversary because W_A and W_B are chosen independently by the A and B, respectively. Furthermore, even if a session of this type is not an targeted session for $Test$-query, the adversary cannot validate any possible values of $pw_{A/B}$ through the received messages (including $\tau_{A/B,S}$, $\tau_{S,A/B}$, $S_{A/B}$) and the session keys returned via $Reveal$-query based on the ECCDH assumption since s is unknown to $\mathcal{A}$ either unless the adversary has compromised the user's passwords via $Corrupt$-query later. Thus, we have: $Pr[Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A})|CaseC] \le \varepsilon(l)$ based on the ECCDH assumption.

As a consequence, one gets the announced result:

$$Adv^{fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \le O(\frac{q_s}{|\mathcal{D}|}) + \varepsilon(l).$$

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 authentication 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 = 1 undetectable online dictionary attack [29].

5 Conclusion

In this paper, we have demonstrated that EC analog of Chien's 3PAKE protocol is still vulnerable to offline dictionary attacks. Thereafter, we have proposed an efficient protocol that can defeat password-guessing attacks and the stolen-verifier attacks. Furthermore, we have provided the rigorous proof of the security for it.

Acknowledgments This paper was supported in part by the Important National Science and Technology Specific Project: “Next-generation broadband wireless mobile communication network” (No. 2012ZX 03002003).

References

1. Bellare, M.; Pointcheval, D.; Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Proceedings of Advances in Cryptology: EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer (2000)
2. Boyko, V.; MacKenzie, P.D.; Patel, S.: Provably secure password-authenticated key exchange using Diffie–Hellman. In: Proceedings of Advances in Cryptology: EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer (2000)
3. Bresson, E.; Chevassut, O.; Pointcheval, D.: New security results on encrypted key exchange. In: Proceedings of PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography. LNCS, vol. 2947, pp. 145–158, Springer (2004)
4. Gennaro, R.; Lindell, Y.: A framework for password-based authenticated key exchange. In: Proceedings of Advances in Cryptology: EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer (2003)
5. Goldreich, O.; Lindell, Y.: Session-key generation using human passwords only. In: Proceedings of Advances in Cryptology: CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer (2001)
6. Boyd, C.; Montague, P.; Nguyen, K.: Elliptic curve based password authenticated key exchange protocols. In: Proceedings of 28th Australasian Conference on Information Security and Privacy: ACISP 2001. LNCS, vol. 2119, pp. 487–501, Springer (2001)
7. MacKenzie, P.D.; Patel, S.; Swaminathan, R.: Password-authenticated key exchange based on RSA. In: Proceedings of Advances in Cryptology: ASIACRYPT 2000. LNCS, vol. 1976, pp. 599–613. Springer (2000)
8. Abdalla, M.; Chevassut, O.; Pointcheval, D.: One-time verifier-based encrypted key exchange. In: Proceedings of the 8th International Workshop on Theory and Practice in Public Key (PKC ’05). LNCS, vol. 3386, pp. 47–64. Springer (2005)
9. Abdalla, M.; Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Proceedings of Topics in Cryptology: CT-RSA 2005. LNCS, vol. 3376, pp. 191–208, Springer (2005)
10. Abdalla, M.; Fouque, P.-A.; Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Proceedings of PKC’2005. LNCS, vol. 3386, pp. 65–84, Springer (2005) [Full version appeared in IEE Information Security **153**(1), 27–39 (2006)]
11. Abdalla, M.; Pointcheval, D.: Interactive Diffie–Hellman assumptions with applications to password-based authentication. In: Proceedings of FC’2005. LNCS, vol. 3570, pp. 341–356. Springer (2005)
12. Lu, R.X.; Cao, Z.F.: Simple three-party key exchange protocol. Comput. Secur. **26**, 94–97 (2007)
13. Chien, H.Y.; Wu, T.C.: Provably secure password-based three-party key exchange with optimal message steps. Comput. J. **52**(6), 646–655 (2009)
14. Huang, H.-F.: A simple three-party password-based key exchange protocol. Int. J. Commun. Syst. **22**(7), 857–862 (2009)
15. Zeng, Y.; Ma, J.; Moon, S.: An improvement on a three-party password-based key exchange protocol using weil pairing. Int. J. Netw. Secur. **11**(1), 17–22 (2010)
16. Lo, J.-W.; Lee, J.-Z.; Hwang, M.-S.; Chu, Y.-P.: An advanced password authenticated key exchange protocol for imbalanced wireless networks. J. Internet Technol. **11**(7), 997–1004 (2010)
17. Lee, T-F.; Hwang, T.: Simple password-based three-party authenticated key exchange without server public keys. Inf. Sci. **180**(9), 1702–1714 (2010)
18. Chang, T.-Y.; Hwang, M.-S.; Yang, W.-P.: A communication-efficient three-party password authenticated key exchange protocol. Inf. Sci. **181**, 217–226 (2011)
19. H.-Y. Chien.: Secure verifier-based three-party key exchange in the random oracle model. J. Inf. Sci. Eng. **27**(4), 1487–1501 (2011)
20. Lou, D.-C.; Huang, H.-F.: Efficient three-party password-based key exchange scheme. Int. J. Commun. Syst. **24**(4), 504–512 (2011)
21. Wang, W.; Hu, L.: Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. In: Proceedings of INDOCRYPT 2006. LNCS, vol. 4329, pp. 118–132. Springer (2006)
22. Choo, K.-K.R.; Boyd, C.; Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Proceedings of ASIACRYPT’2005. LNCS, vol. 3788, pp. 585–604. Springer (2005)
23. Chung, H.-R.; Ku, W.-C.: Three weaknesses in a simple three-party key exchange protocol. Inf. Sci. **178**, 220–229 (2008)
24. Yoon, E.J.; Yoo, K.Y.: Cryptanalysis of a simple three-party password-based key exchange protocol. Int. J. Commun. Syst. **24**(4), 532–542 (2011)
25. Hankerson, D.; Menezes, A.; Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Berlin (2004)
26. Koblitz, N.: Elliptic curve cryptosystem. Math. Comput. **48**, 203–209 (1987)
27. Abdalla, M.; Bellare, M.; Rogaway, P.: The oracle Diffie–Hellman assumptions and an analysis of DHIES. In: Proceedings of CT-RSA’2001, pp. 143–158. Springer (2001)
28. Pointcheval, D.: Provable Security for public key schemes. In: Contemporary Cryptology. Advanced Courses in Mathematics, CRM Barcelona, pp. 133–189 (2005)
29. Ding, Y.; Horster, P.: Undetectable on-line password guessing attacks. ACM Oper. Syst. Rev. **29**, 77–86 (1995)
30. Liang, H.; Hu, J.; Wu, S.: Re-attack on a three-party password-based authenticated key exchange protocol. Math. Comput. Model. (2012). doi:10.1016/j.mcm.2012.10.019
31. Wu, S.: Security analysis and enhancements of verifier-based password-authenticated key exchange protocols in the three-party setting. J. Inf. Sci. Eng. **27**, 1059–1072 (2011)