Provably Secure Password-based Three-party Key Exchange Protocol with Computation Efficiency

Jih-Ming Fu[1], Jeng-Ping Lin[2], Ren-Chiun Wang[3*]

1 Department of Computer Science & Information Engineering, Cheng Shiu University, No.840, Chengcing Rd., Niaosong Dist., Kaohsiung City 83347, Taiwan (R.O.C.)
2 Department of Commerce Technology & Management, Chihlee Institute of Technology, 313, Sec. 1, Wunhua Rd., Banciao District, New Taipei City, 22050 Taiwan, R.O.C
3 Project Resource Division, Institute for Information Industry
rcwang@icst.org.tw

Abstract: Going along with the rapid development of web technologies, people can make a great quantity of service requests to service providers using mobile devices anytime and anywhere. However, the service requester and the service providers may not trust each other and they may locate at different domain. They require a communal trusted third party to help them establish a shared session key for secure communications. It is so-called three-party key exchange. Recently, many password-based three-party key exchange protocols were proposed against various well-known security threats. In those protocols, to prevent the password guessing attack, a widely used way is to employ public-key and/or symmetric-key cryptosystems to protect the exchanged messages. As we known, the encrypted and decrypted operations in a public-key cryptosystem are time-consuming. In this paper, we propose a password-based three-party key exchange protocol with the computation-efficiency without using public-key systems. Finally, we prove the security of the proposed protocol in the random oracle model.

[Jih-Ming Fu[1], Jeng-Ping Lin[2], Ren-Chiun Wang. **Provably Secure Password-based Three-party Key Exchange Protocol with Computation Efficiency.** Life Science Journal. 2011; 8(4):394-401] (ISSN:1097-8135). http://www.lifesciencesite.com.

Keywords: cryptography; discrete logarithm problem; on-line undetectable password guessing attack; three-party key exchange.

1. Introduction

Today, people have many opportunities to obtain services or resources from application servers by using their mobile devices through the Internet. However, both of the clients and the servers may be distributed over different network domains and do not win the trust each other. A secure mechanism has to make sure that the identity of the clients and the server can be authenticated each other and the communications are secure against an unauthorized user from eavesdropping the delivery contents[1-2,5]. The client and the application server require a 3. communal trusted third party[3-4,17].

Password is widely employed to construct a secure key exchange protocol since password-based protocols are easily to be developed and to be maintained. However, users have to worry about whether their passwords (have low entropies) have been guessed or not. The password guessing attacks can be divided into three kinds[11-12]:

1. **On-line detectable guessing attack.** Attacker can enumerate all the candidature passwords and pick up one from the list. Then the attacker sends the chosen password to connect the server and verifies the server's response in on-line. Most password-based protocols can prevent this attack by the server limits the fail times.
2. **On-line undetectable guessing attack.** Attacker can enumerate all the candidature passwords and pick up one from the list. Then the attacker sends the chosen password to connect the server and verifies the server's response in on-line. Since the server cannot discriminate whether the request is malicious or honest, therefore the server always replies a honest response. The attacker can catch this chance to guess the password until the password is correctly obtained[23].

Off-line guessing attack. Since the communicated channel is open, any eavesdropper can collect all the communications. Then the attacker can enumerate all the candidature passwords to launch the attack off-line until a hit is obtained without the help of the server.

Many password-based three-party key exchange protocols were proposed and addressed to overcome the above guessing attacks by using the concept of public-key and symmetric-key techniques[10-11,19-20,26]. For enhancing the efficiency dramatically, in 2007, Lu and Cao proposed a simple three-party key exchange protocol[21] without using the server's public key. Unfortunately, Lu-Cao's key exchange protocol suffered from the unknown key sharing[1], the on-line undetectable guessing, and the impersonation

attacks[12,15,18,23]. For guaranteeing the quality of communication services, low communication and computation cost is required in a three-party key exchange protocol. In 2009, Huang[16] proposed an efficiency-enhanced password-based three-party key exchange protocol. Huang claimed that the proposed protocol is also more efficient than Lu-Cao's protocol and can be applied in practice. However, Huang's protocol is still not secure against the on-line undetectable guessing attack[25].

We propose a provably secure password-based three-party key exchange protocol to withstand various well-known security threats by using the random oracle model[3,11,22]. Compared with the related protocols [10-11,20], our proposed protocol is computation-efficient.

In the next section, we first give a notation of security. In Section 3, we propose a novel three-party key exchange protocol. In Section 4, we analyze the security of the proposed protocol. In Section 5, we analyze the efficiency among our proposed protocol and the related protocols.

1 An unknown key-sharing attack on a key Finally, we conclude this paper in Section 6.exchange protocol which provides the key confirmation property is an attack whereby an entity A believes that she shares a session key with the communicated entity B. Unfortunately, it is fact that if the entity B mistakenly believes that the session key is instead shared with another entity E, where $E \neq A$. A secure key exchange protocol should be against this threat[6,8].

2. Notations of Security

We first define some hard mathematical problems and security of a password-based three-party protocol.

2.1 Hard Problems

1. **Definition 1. Discrete Logarithm Problem (DLP).** Given two elements g and g^a, it is computationally infeasible to find a, where p is a large prime number, g is a generator with order q in 2. $GF(p)$ and $a \in Z_q^*$.

2. **Definition 2. Computational Diffie-Hellman Problem (CDHP).** Given three elements g, g^a, and g^b, it is computationally infeasible to calculate g^{ab}, where p is a large prime number, g is a generator 3. with order q in $GF(p)$ and both of a and $b \in Z_q^*$.

3. **Definition 3. Decisional Diffie-Hellman Problem (DDHP).** Given four elements g, g^a, g^b, and g^c, it is difficult to decide whether c mod q is equal 4. ab mod q, where p is a large prime number, g is a generator with order q in $GF(p)$ and all of a, b and $c \in Z_q^*$.

2.2 Security Definitions

The concrete security of a three party-based protocol is built up both the property of the session key indistinguishability and the protection of the password[7,22]. In a password-based protocol, an on-line detectable guessing attack[14] is inherent and is inevitable. However, this attack can be prevented by locking the account after some reasonable failed attempts in most password-based protocols. A more dangerous attack is the off-line guessing attack after an adversary copies a transcript of executions in a password-based protocol. The mission of a password-based protocol is to rule out the off-line guessing attack and to limit the adversary only to the on-line detectable guessing attack. For thwarting the online detectable guessing attack, the service requesters' requests are required to be authenticated for the operations of the trusted server from distinguishing malicious attempts from real requests. Also, for deterring the on-line undetectable and the off-line guessing attacks, the proposed protocol has to live up to the requirement of attackers that they may pick up the correct password but cannot verify their guessing from the eavesdropped messages.

We denote the proposed protocol, a service requester C_A and a service provider $C_B \in \hat{C} = \{C_1, ..., C_{NC}\}$ and a trusted server S. Each service requester C_A and a service provider $C_B \in \hat{C}$ hold memorial passwords pw_A and pw_B, and the server S maintains a password table $<P_1, ..., P_{NC}>$. We also assume that an adversary AD who controls all the communications that take place by C_A^i, C_B^j and S is a probabilistic machine, where we denote that C_A^i is the ith instance of the service requester C_A and C_B^j is the jth instance of the service provider C_B. AD can interact with all the participants (C_A, C_B, S) through the following oracle queries.

1. Execute(C_A^i, C_B^j), Execute(C_A^i, S), Execute(C_B^j, S): We use this query to model passive attacks where an attacker can eavesdrop all the communications between the instances (C_A^i, C_B^j) and between the instances (C_A^i, S), and (C_B^j, S) respectively.

2. SendClient(C_A^i, m): We use this query to model an active attack against that the attacker sends a message m to a participant C_A at the ith instance. Then query outputs the result of C_A from receiving the message m to generate.

3. SendServer(m): We use this query to model an active attack against that the attacker sends a message m to the server S. Then query outputs the result of S from receiving the message m to generate.

4. Reveal(C_A^i): We use this query to model an active attack against the known-key attack at the ith instance C_A. The query says that if the instance does not accept the session key, the output is $\perp$; otherwise, the output is the real session key.

5. Corrupt(C_A): We use this query to allow that an attacker AD can corrupt the complete internal state of an entity C_A.
6. Test(C_A^i): If an attacker AD queries this oracle and no session key for $C_A^i \in \hat{C}$ is accepted, this oracle outputs $\perp$; otherwise, the oracle flips a coin b. If $b = 1$, returns the real session key; if $b = 0$; returns a random key which has the same key with the real session key.

The security definition of the proposed protocol depends on the partnership and freshness of oracles, where the partnership of the oracles is defined using the session identifiers *sids* and the partnership is defined to restrict the adversary's Reveal and Corrupt queries. If the partnership is not accepted by the oracles, the adversary is trying to guess the session key.

1. Partnership: We say that two oracles C_A^i and C_B^j are partners, if and only if both of the oracles have accepted the same session key with the same session identifier and they have agreed on the same set of exchanging messages. Besides C_A^i and C_B^j, no other oracles have accepted with the same session identifier.
2. Freshness: We say that two oracles C_A^i and C_B^j are fresh if and only if the oracle C_A^i has accepted another partner oracle C_B^j, the oracle C_B^j has accepted another partner oracle C_A^i, and all the oracles C_A^i and C_B^j have not been sent a Reveal query a Corrupt query.
3. Session key security: We use the standard semantic security notation to model this property[22]. The security of session key is defined that the adversary who wants to discriminate a real key from a random one in the game G is indistinguishable, where the game played between the adversary AD and a collections of U_x^i oracles. The players $U_x \in \hat{C}$ and S and instances $i \in \{1, ..., N_I\}$. AD runs the game G with the following stages.

- Stage 1: AD is allowed to send the queries (Execute, SendClient, SendServer, Reveal and Corrupt) in the game.
- Stage 2: During the game G, at some point, AD can choose a fresh session and end a Test query to one of the fresh oracles C_A^i and C_B^j for the testing. Depending on the unbiased coin b, AD is given ether the actual session key K or a random one from the session key distribution.
- Stage 3: AD can continue to send the queries to the oracles Execute, SendClient, SndServer, Reveal and Corrupt for its choice. However, AD is restricted to send the Reveal and Corrupt queries to the oracles for its test session.
- Stage 4: Eventually, AD winds up the game simulation and decides to output its guess bit b'.

The success of AD from breaking the protocol in the game depends on passwords which are drawn from a dictionary D and is measured in terms of the advantage of AD from distinguishing whether the received value is the real key or a random one. Let $Adv_{P,D}^{G,AD}(k, q_{fake\text{-}C})$ be the advantage of AD and the advantage function be be defined as follows.

$$Adv_{P,D}^{G,AD}(k,q_{fake\text{-}C})=|Pr[b'\text{-}b]\text{-}q_{fake\text{-}C}/N\text{–}1/2*(N\text{-}q_{fake\text{-}C})| \quad (1)$$

where k is a security parameter, N denotes the size of the dictionary D and $q_{fake\text{-}C}$ denotes the number of attempts of the adversary from faking the client. After $q_{fake\text{-}C}$ times of faking the client, the intuition of the formulation is that the advantage of the adversary from finding the correct password and from faking the session key successfully should have the probability at most $q_{fake\text{-}C}/N$. The rest of non-successful faking cases could have the successful probability 1/2.

Password protection: An adversary may try to guess the password of a valid client and verify its guess through the interaction with the server or the client or from the intercepted messages. We require that the protocol has to provide the explicit authentication of a client's request for thwarting the online detectable guessing attack in which the server can do some actions such that the limitation of invalid request attempts cannot exceed the pre-defined threshold. Security against the adversary from launching the off-line guessing and the online undetectable guessing attacks, the protocol should not provide any advantageous information to outsiders or to a curious partner to verify its guess.

Definition 4. We say that a password-based three-party key exchange protocol is secure in our model when the following requirements are satisfied:

1. Validity: Among three oracles (C_A^i, C_B^j, S), the oracles (C_A^i, C_B^j) accept the same session key in the absence of an active adversary.
2. Session key indistinguishability: For all probabilistic, the advantage of the adversary AD is negligible within a polynomial time.
3. Explicit authentication: As the above mentioned, the protocol should make sure that the explicit authentication of two communicated parties is done for being against the online detectable guessing attacks.
4. Password protection: As the above mentioned, the protocol should not provide any advantageous information to outsiders or to a curious partner to verify its guess for being against the off-line guessing and the undetectable online guessing attacks.

3. Our Proposed Protocol

In our protocol, we define $h_1()$ and $h_2()$ are

secure cryptographic one-way hash functions and we will model the functions as random oracles in the security proof. The other parameters are introduced as follows:

A. The system selects a large prime number p, where $(p - 1)$ has a prime factor q.

B. Let g be a generator with order q in $GF(p)$.

C. TS denotes the trusted third party.

D. A and B denote two communicated parties.

E. pw_A and pw_B denote the passwords that A shared with TS and B shared with TS, respectively.

F. $\oplus$ denotes an exclusive OR operation.

G. For simplicity, all the exponentiation operations are under the modular p such as $g^x \bmod p \rightarrow g^x$.

1. Request that initiator A selects a random number x, calculates $R_A = g^x \oplus h_1(pw_A, A, B, sid)$, and sends (A, sid, R_A) to the responder B, where the sid denotes the session identity.

2. Upon receiving the request, B also selects a random number y, calculates $R_B = g^y \oplus h_1(pw_B, A, B, sid)$, and sends (B, R_B) with A's request to the trusted server TS.

3. (a) Upon receiving (A, B, sid, R_A, R_B), TS employs the passwords pw_A and pw_B to extract the exchanged information g^x and g^y, respectively. Then T selects three random numbers (z_1, z_2, z_3) and calculates (a, b, c, d), where $a = g^{xz_1}$, $b = g^{yz_1}$, $c = g^{z_2}$, and $d = g^{z_3}$.

(b) TS sends (A, sid, Z_{A1}, Z_{A2}) and (B, sid, Z_{B1}, Z_{B2}) to A and B in parallel, where $Z_{A1} = b \oplus h_1(pw_A+1, A, B, sid)$, $Z_{A2} = c \oplus h_1(pw_A+2, A, B, sid)$, $Z_{B1} = a \oplus h_1(pw_B+1, A, B, sid)$, and $Z_{B2} = d \oplus h_1(pw_B+2, A, B, sid)$.

4. Do in parallel

(a) Upon receiving (B, sid, Z_{B1}, Z_{B2}), B employs $h_1(pw_B+1, A, B, sid)$ and $h_1(pw_B+2, A, B, sid)$ to recover a and d. B then calculates the session key $K = h_2(A, B, sid, a^y)$, $S_{B1} = h_1(A, B, sid, K)$ and $S_{B2} = h_1(A, B, sid, d^y, a)$. B sends S_{B1} to A and S_{B2} to TS for identifying the validation of its identity and the session key.

(b) Upon receiving (A, sid, Z_{A1}, Z_{A2}), A employs $h_1(pw_A+1, A, B, sid)$ and $h_1(pw_A+2, A, B, sid)$ to recover b and c. A then calculates the session key $K = h_2(A, B, sid, b^x)$, $S_{A1} = h_1(A, B, sid, K+1)$ and $S_{A2} = h_1(A, B, sid, c^x, b)$. A sends S_{B1} to B and S_{A2} to TS for identifying the validation of its identity and the session key.

5. Do in parallel

(a) Both of A and B can authenticate each other by checking the validation of S_{B1} and S_{A1} and believe that the owned session key is fresh.

(b) Upon receiving A and B's responses, TS can check the validation of S_{B2} and S_{A2}. If any of the conditions does not hold, TS will return "*connection failure*" message to the corresponding parties and increase the fail times by one. We introduce the proposed protocol in Figure 1.

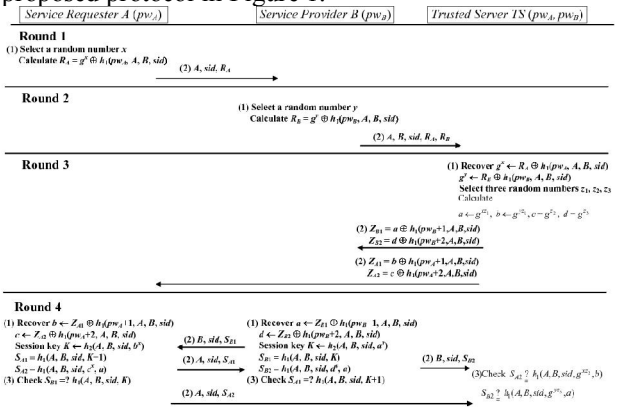

Figure 1. The proposed protocol

4. Security Analysis

In this section, we analyze that the proposed protocol is secure against some well-known attacks. Before our analysis, we first assume that the following mathematical problems are hard to be solved[9,13].

4.1 Analysis

Session Key Security.

(a) Even if $a = g^{xz_1}$ and $b = g^{yz_1}$ are known by an adversary, based on the difficulty of the CDHP, the adversary cannot derive the session key $K = g^{xyz_1}$ except the parties A and B.

(b) Based on the properties of one-way hash function and the exclusive-OR operator, the adversary is useless to derive (g^x, b, g^y, a) without the knowledge of A and B's passwords. The reason is that the extracted values cannot be verified. The adversary wants to discriminate (g^x, b, g^y, a) from $(R_A, R_B, Z_{A1}, Z_{B1})$, the probability of obtaining the session key K is equivalent to solve the CDHP on $(Z_{A1}, S_{A1}, Z_{B1}, S_{B1})$.

Replay Attack. An adversary who wants to imitate the requester A can resend the used messages $(R_A = g^x \oplus h_1(pw_A, A, B, sid))$ to B or to TS and expect to obtain some useful information from TS such as $(Z_{A1} = g^{yz_1} \oplus h_1(pw_A+1, A, B, sid), Z_{A2} = g^{z_2} \oplus h_1(pw_A+2, A, B, sid))$. Based on the CDHP assumption, the adversary not only cannot derive new session key $K = g^{xyz_1}$ without the knowledge of the ephemeral keys x, but also cannot win the trust of TS without the knowledge of the passwords pw_A since g^{z_2} is encrypted using the password pw_A.

Impersonation Attack. In Round 3 of our

proposed protocol, when someone sends the exchanged messages to *TS*, *TS* always returns the messages (Z_{A1}, Z_{A2}, Z_{B1}, Z_{B2}) back. The adversary can catch this chance to launch the attack. Note that *TS* waits the responses in Round 4. Since all the exchanged messages must be encrypted using the password independently, the adversary cannot know whether the guessed password is correct or not and also cannot judge whether the received message S_{B1} and the computed results (S_{A1}, S_{A2}) are correct or not. Based on the difficult of the CDHP, this way is blocked.

4. **Password Guessing Attack.**

(a) On-line detectable guessing attack. In current systems, there is a standard mechanism to defeat this attack. The solution is that the remote server logs and counts the number of trial failures. If the number is larger than the pre-defined threshold values, the server stops the connection. This concept can be applied to our protocol since *TS* verifies whether *A* and *B*'s responses (S_{A2}, S_{B2}) are correct or not in Round 4 and records the failure times.

(b) On-line undetectable guessing attack. To launch the attack successfully, the attacker has to get some useful information in advance for manipulating the data and verifying their guess on *TS*'s response (or *B*'s response). The attack cannot work on our protocol since all the requests have to be sent to *TS* and *TS* will wait the feedbacks from both of *A* and *B*. It implies that any trial process will be detected by *TS*. The attack fails.

(c) Off-line guessing attack. All the exchanged messages are encrypted using the passwords independently. The goal of the adversary is to guess the password and to verify the correctness on the intercepted messages. Based on the difficult of the CDHP, the adversary cannot employ the guessed password and derive messages to obtain any results on the messages (S_{A1}, S_{A2}, S_{B1}, S_{B2}) in Round 4.

5. **Forward/Backward Secrecy.**

(a) In each session, *A*, *B* and *TS* select their ephemeral keys (x, y, z_1, z_2) to construct ($R_A = g^x \oplus h_1(pwA, A, B, sid)$, $R_B = g^y \oplus h_1(pw_B, A, B, sid)$, $z_{A1} = b \oplus h_1(pwA+1, A, B, sid)$, $z_{B1} = a \oplus h_1(pw_B+1, A, B, sid)$). Based on the difficult of the CDHP, the adversary cannot calculate the session key $K = h_2(A, B, sid, g^{xyz_1})$ in all the sessions even if the passwords are guessed correctly. The property of the forward secrecy is provided.

(b) Even if one of the used session key $K = h_2(A, B, sid, g^{xyz_1})$ is compromised by the adversary, the adversary cannot obtain any useful information on the corresponding messages. For instance, the adversary may guess the password to get $g^{x'}$ and g^{yz_1}'. Based on the difficult of the CDHP, the adversary cannot verify the guessed password. As the above mentioned, without the knowledge of the password, the adversary cannot launch any attacks. Hence, the backward secrecy is also kept in our protocol.

Theorem 1. We claim that the proposed password-based three-party key exchange protocol is secure in the random oracle model if the CDHP is hard.

Proof. We then give the detailed proof in the appendix.

5. Efficiency Analysis

In this section, we analyze the computation cost of a service requester because the requester could use personal mobile devices to obtain the desirable services. Also, as introduced in[24], we can learn a relationship as follows: the time of one modular exponentiation is faster 5/3 times than the time of one public-key en/decryption operation, the time of one modular multiplication computation is faster 240 times than the time of one modular exponentiation operation, and the time of one one-way hash function operation is faster 600 times than the time of one modular exponentiation.

In Round 1, *A* calculates $R_A = g^x \oplus h_1(pwA, A, B, sid)$. The cost is one modular exponentiation plus one hash function operation. In Round 4, *A* recovers $b = Z_{A1} \oplus h\_1(pwA+1, A, B, sid)$ and $c = Z_{A2} \oplus h_1(pwA+2, A, B, sid)$. The cost is two hash function operations. Then *A* calculates the session key $K = h_2(A, B, sid, b^x)$, $S_{A1} = h_1(A, B, sid, K+1)$ and $S_{A2} = h_1(A, B, sid, c^x, a)$. The cost is two modular exponentiation plus 4 hash function operations. By the above, the computation cost of *A* is 3 modular exponentiations plus 6 hash function operations.

In the communication cost, we denote that:

1. Message Step denotes that one entity has sent data to the communicated party.
2. Communication Round means that if the sent data are independent between each message steps, one or more message steps can be integrated into the same communication round due to the sent data can be performed in parallel. The burden of the communication cost can be reduced.

We summarize the results in Table 1 and we can see that our protocol is more efficient than the related protocols[10-11,16,20-21].

5. Conclusions

In this paper, we have proposed a provably secure password-based three-party key exchange protocol to overcome some well known security

threats. Compared with the related protocols, the computation efficiency is still kept in our proposed protocol.

Table 1. Comparisons of the Computation Cost At Requester Side and the Communication Cost

	Our	Lu-Cao[21]*3	Huang[16]*4	Chien-Wu[11]	Chen *et al.*[10]*	Lo-Yeh[20]
T_{EXP}	3	4	2	2	3	3
T_{MUL}	0	2	0	0	0	0
T_H	7	3	4	4	4^{*1}	4^{*1}
T_{PKC}	0	0	0	1	1^{*2}	1^{*2}
T_{SYM}	0	0	0	0	1	1
Total(T_{MUL})	722.8	963.2	481.6	881.6	$1121.6+1T_{SYM}$	$1121.6+1T_{SYM}$
Rounds/Steps	4/8	5/5	5/5	4/4	5/5	4/6

T_{EXP} denotes the time of one modular exponentiation operation; T_{MUL} denotes the time of one modular multiplication computation; T_H denotes the time of one hash function operation; T_{PKC} denotes the time of one public-key en/decryption operation; T_{SYM} denotes the time of one symmetric-key en/decryption operation;
*: the protocol has been proven that the on-line undetectable guessing attack still exists[20].
*1: the computation cost of pseudo-random hash function is similar to the cost of one-way trapdoor hash function.
*2: the computation cost of one-way trapdoor function is similar to the cost of public key en/decryption.
*3: The protocol which is not secure against the unknown key sharing, the on-line undetectable guessing, and the impersonation attacks has been proven by[12,15,18,23].
*4: Wu had have shown that the protocol is not secure against the on-line undetectable guessing attack[25].

Correspondence:
Ren-Chiun Wang
E-mail: rcwang@icst.org.tw

References

1. M. Abdalla, E. Bresson, O. Chevassut, B. MÖoller, D. Pointcheval, Strong Password-Based Authentication in TLS using the Three-Party Group Diffie-Hellman Protocol, International Journal of Security and Networks. 2007, 2(3/4):284-296.
2. M. Abdalla, D. Catalano, C. Chevalier, D. Pointcheval, Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework, in: Topics in Cryptology - CT-RSA 2008, LNCS 4964, 2008, 335-351.
3. M. Abdalla, P.-A. Fouque, D. Pointcheval, Password-Based Authenticated Key Exchange in the Three-Party Setting, in: Public Key Cryptography - PKC 2005, LNCS 3386, 2005, 65-84.
4. M. Abdalla, P.-A. Fouque, D. Pointcheval, Password-Based Authenticated Key Exchange in the Three-Party Setting, IEE Proceedings, 2006, 153 (1):27-39.
5. M. Abdalla, D. Pointcheval, A Scalable Password-based Group Key Exchange Protocol in the Standard Model, in: Advances in Cryptology - ASIACRYPT 2006, LNCS 4284, 2006, 332-347.
6. J. Baek, K. Kim, Remarks on the unknown key-share attacks, IEICE Trans. on Fundamentals, 2000, E83-A(12):2766-2769.
7. M. Bellare, P. Rogaway, Provably secure session key distribution the three party case, in: Proc. of the 27th ACM Annual Symposium on the Theory of Computing, 1995, 57-66.
8. S. Blake-Wilson, A. Menezes, Unknown key-share attacks on the stationto-station (STS) protocol, in: Public Key Cryptography (PKC '99) Proceedings, LNCS 1560, 1999, 154-170.
9. S. Boneh, B. Lynn, H. Shacham, Short Signatures from theWeil Pairing, in: Advances in Cryptology - ASIACRYPT 2001, LNCS 2284, 2001, 514-532.
10. H.-B. Chen, T.-H. Chen, W.-B. Lee, C.-C. Chang, Security enhancement for a three-party encrypted key exchange protocol against undetectable on-line password guessing attacks, Computer Standards & Interfaces, 2008, 30(1-2):95-99.
11. H.-Y. Chien, T.-C. Wu, Provably secure password-based three-party key exchange with optimal message steps, The Computer Journal, 2009, 52(6):646-655.
12. H.-R. Chung, W.-C. Ku, Three weaknesses in a simple three-party key exchange protocol, Information Sciences, 2008, 178(1):220-229.
13. W. Diffiee, M. Hellman, New directions in cryptology, IEEE Transations on Information Theory, 1976, IT-22(6):644-654.
14. Y. Ding, P. Horster, Undetected on-line password guessing attacks, ACM Operating Systems Review, 1995, 29(4):77-86.
15. H. Guo, Z. Li, Y. Mu, X. Zhang, Cryptanalysis of simple three-party key exchange protocol, Computers & Security, 2008, 27(1-2):16-21.
16. H.-F. Huang, A simple three-party password-based key exchange protocol, International Journal of Communication Systems, 2009, 22(7):857-862.
17. W.-S. Juang, Efficient three-party key exchange using smart cards, IEEE Trans. on Consumer Electronics, 2004, 50(2):619-624.
18. H.-S. Kim, J.-Y. Choi, Enhanced password-based simple three-party key exchange protocol, Computers & Electrical Engineering, 2009, 35(1):107{114.
19. T.-F. Lee, J.-L. Liu, M.-J. Sung, S.-B. Yang, C.-M.

Chen, Communication-efficient three-party protocols for authentication and key agreement, Computers & Mathematics with Applications, 2009, 58(4):641-648.
20. N.-W. Lo, K.-H. Yeh, Cryptanalysis of two three-party encrypted key exchange protocols, Computer Standards & Interfaces, 2009. 31(6):1167-1174.
21. R. Lu, Z. Cao, Simple three-party key exchange protocol, Computers & Security, 2007, 26(1):94-97.
22. D. P. M. Bellare, P. Rogaway, Authenticated and key exchange secure against dictionary attacks, Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, 2000, 139-155.
23. R. C.-W. Phan, W.-C. Yau, B.-M. Goi, Cryptanalysis of simple threeparty key exchange protocol (S-3PAKE), Information Sciences, 2008, 178(13):2849-2856.
24. B. Schneier, Applied cryptography, 2nd edition, John Wiley & Sons Inc., 1996.
25. S. Wu, Weakness of a three-party password-based authenticated key exchange protocol, Report 2009/535, CryptEAr (Nov. 2009). URL http://eprint.iacr.org/2009/535.pdf
26. E.-J. Yoon, K.-Y. Yoo, Improving the novel three-party encrypted key exchange protocol, Computer Standards & Interfaces, 2008, 30(5):309-314.

Appendix

A. Security Proof

We prove that our protocol provides the session key indistinguishability property in the random oracle model under the CDHP assumption.

Proof. We use a contradiction way to prove it. We assume that an adversary AD can gain a non-negligible advantage to distinguish the test key in the game and AD can construct a breaker AD" to solve the CDHP problem, where the advantage of AD from differentiating the real session key from a random key as follows:

$\mathrm{Adv}_{P,D}^{G,\mathrm{AD}}(k,q_{fake\text{-}C}) = |\Pr[b'\text{-}b]\text{-}q_{fake\text{-}C}/N\text{–}1/2*(N\text{-}q_{fake\text{-}C})|$.

We suppose that an oracle C_A has accepted the session key of the form $K = h_2(A, B, sid, g^{xyz_1})$ with another fresh and partnership oracle C_B. We say that AD is successful if AD picks an oracle C_A or C_B to ask a Test query and can output the bit guess correctly. Thus, we have Pr[AD succeeds] = $q_{fake\text{-}C}/N + 1/2 * (N - q_{fake\text{-}C})/N + \eta(k)$, where $\eta(k)$ is non-negligible.

Let Q_h be the event that $h_1()$ or $h_2()$ has been queried on (A, B, sid, g^{xyz_1}) by AD or some oracles. Then Pr[AD succeeds] = $q_{fake\text{-}C}/N$ + Pr[AD succeeds | Q_h] * Pr[Q_h] + Pr[AD succeeds | $\overline{Q_h}$] * Pr[$\overline{Q_h}$]. Since $h_1()$ and $h_2()$ are random oracles and C_A and C_B are fresh oracles, it implies Pr[AD succeeds | $\overline{Q_h}$] = 1/2. Hence, $q_{fake\text{-}C}/N + 1/2 * (N - q_{fake\text{-}C})/N + \eta(k) \le q_{fake\text{-}C}/N + 1/2 * (N - q_{fake\text{-}C})/N + \Pr[Q_h]$. We then have $\Pr[Q_h] \ge \eta(k)$.

The adversary AD selects a fresh oracle C_A which has accepted a session key. Then the probability of $h_2()$ being queried on (A, B, sid, g^{xyz_1}) by AD or some oracles other than C_A and C_B is non-negligible. As mentioned before, we have assumed that AD constructs a breaker AD" which can solve the CDHP with non-negligible probability. The task of AD" is that: Given $X = g^x$ and $Y = g^y$, AD" outputs g^{xy}, where x and y are chosen randomly.

AD" executes the following process:

1. Randomly select C_A and C_B from $\hat{C} = \{C_1, C_2, ..., C_{NC}\}$ and instances u and i from $\{1, 2, ..., N_I\}$, where N_C and N_I denote the number of service requesters and service providers and the instances per entity. Note that all these parameters are polynomial on the security parameter.
2. Determine two oracles C_A^u and C_B^v who are partnership.
3. Guess that AD will choose one of C_A^u and C_B^v who have accepted the session to ask its Test query after AD decides to terminate the game.

Given the challenge $(X^* = g^x, Y^* = g^y)$ to AD", AD" sets the public parameters as (g, p). AD" also maintains the lists L_{h1} and L_{h2} for the random oracles $h_1()$ and $h_2()$ queries, L_{Send} for the communicated transcripts, and L_{Key} for the corresponding keys of each session. AD" selects the passwords pw for each C_A and $C_B \in \{C_1, C_2, ..., C_{NC}\}$ at random and lets $pw_{\hat{C}}$ be the password file of TS.

During the game, AD will ask some queries to AD". The answers are given as follows:

1. Hash query: AD" randomly responses $h_1()$ and $h_2()$ queries which are like real random oracles do, and records all the inputs and the corresponding outputs in L_{h1} and L_{h2}, respectively.
2. Corrupt(C) query: If C is one of C_A and C_B, AD" gives up; otherwise, AD" answers all the internal state of C to AD.
3. SendClient(C_X^i, m) query:

(a) If $(C_X = C_A)$ && $(i = u)$ && $(m = start)$, then AD" sets $N_X = X^*$ and responds the protocol says $\{CA, sid, N_X \oplus h_1(pw_{C_A}, C_A, C_B, sid)\}$. Finally, the oracle records the responsive transcript and the random exponent (?) in the L_{Send} list and $(h_1(pw_{C_A}, C_A, C_B, sid), (pw_{C_A}, C_A, C_B, sid))$ in the L_{h1} list, where ? denotes the corresponding exponent of X^* and is unknown.

(b) If $(C_X = C_B)$ && $(i = v)$ && (m has the form of $(C_A, sid, N_X \oplus h_1(pw_{C_A}, C_A, C_B, sid))$, then AD" sets $N_Y =$

Y^* and responds the protocol says $\{C_A, C_B, sid, N_X \oplus h_1(pw_{C_A}, C_A, C_B, sid), N_Y \oplus h_1(pw_{C_B}, C_A, C_B, sid)\}$. Finally, AD" records the responsive transcript and the random exponents (?) in the L_{Send} list and ($h_1(pw_{C_B}, C_A, C_B, sid)$, ($pw_{C_B}, C_A, C_B, sid$)) in the L_{h1} list, where ? denotes the corresponding exponent of Y^* and is unknown.

(c) If ($C_X \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of ("start", $C_Y \in \{C_1, C_2, ..., C_{NC}\}$ && $C_Y \neq C_X$)), then AD" selects an integer x' at random, calculates $X^* = g^{x'}$, and responds with the transcript $\{C_X, sid, X^* \oplus h_1(pw_{C_X}, C_X, C_Y, sid)\}$. Finally, AD" records the transcript and the randomly secret exponent x' in its L_{Send} and L_{h1} lists.

(d) If ($C_X \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of ($C_Y, sid, Y^* \oplus h_1(pw_{C_Y}, C_Y, C_X, sid)$), then AD" selects an integer x' at random, calculates $X^* = g^{x'}$, and responds with the transcript $\{C_Y, C_X, sid, Y^* \oplus h_1(pw_{C_Y}, C_Y, C_X, sid), X^* \oplus h_1(pw_{C_X}, C_X, C_Y, sid)\}$. Finally, AD" records the transcript and the randomly secret exponent x' in its L_{Send} list.

(e) If ($C_X = C_A \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of (C_X, sid, Z_{C_X1}, Z_{C_X2}) for $C_Y = C_B \in \{C_1, C_2, ..., C_{NC}\}$), then AD" consults its L_{Send} list by using sid to find a matched entry. If the matched entry can be found, AD" extracts the local value from L_{Send} to recover the received data and to calculate K, S_{C_X1} and S_{C_X2}. AD" responds with the transcript $\{C_X, sid, S_{C_X1}, S_{C_X2}\}$. Finally, AD" records corresponding data in its L_{Send}, L_{h1}, L_{h2} and L_{Key} lists respectively. Otherwise, AD" responses with error messages.

(f) If ($C_X \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of (C_X, sid, S_{C_X1})), then AD" consults its L_{Send} list by using sid to find a matched entry. If the matched entry can be found, AD" extracts the local values from L_{h1}, L_{h2} and L_{Key} lists and uses them to verify S_{C_X1}. If the verification does not hold, AD" gives up; AD" records corresponding data in its L_{Send} list.

(g) AD" responses with error messages for all the other cases.

4. SendServer(m) query:

(a) If (C_X and $C_Y \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of ("start", C_X, C_Y, sid, $X^* \oplus h_1(pw_{C_X}, C_X, C_Y, sid)$, $Y^* \oplus h_1(pw_{C_Y}, C_Y, C_X, sid)$)), then AD" uses pw_{C_X} and pw_{C_Y} to recover the received data. AD" selects three integers z_1, z_2 and z_3 at random and responds with the transcript $\{C_X, sid, Z_{C_X1}, Z_{C_X2}\}$ and $\{C_Y, sid, Z_{C_Y1}, Z_{C_Y2}\}$. Finally, AD records all the transcripts and the randomly secret exponents z_1, z_2 and z_3 in its L_{Send} list, L_{h1} list and L_{Key} list respectively.

(b) If ($C_X \in \{C_1, C_2, ..., C_{NC}\}$) && ($m$ has the form of (C_X, sid, S_{C_X2} for C_Y) $\in \{C_1, C_2, ..., C_{NC}\}$), then AD" consults its L_{Send} list by using sid to find a matched entry. If the matched entry can be found, AD" extracts the local values from L_{h1}, L_{h2} and L_{Key} lists and uses them to verify S_{C_X2}. If the verification does not hold, AD" responds an error message to C_Y and records corresponding data in its L_{Send} list.

(c) AD" responses with error messages for all the other cases.

Reveal(C_X^i) query: After receiving the query, AD" consults the records in the list of L_{Key} and reveals all the internal state and the session keys.

AD then answers its guess and requires AD" to searches its L_{h1} and L_{h2} list for the entry, where the entry has the input of the form (C_X, C_Y, sid, $(recovered\ data)^{secretexponent}$) for some K. Finally, AD" outputs K as the Diffie-Hellman key of C_X and C_Y. There are the two possible results for the above experiment:

1. AD" gives up if AD does not make its queries where C_A^u or C_B^v has accepted their session.
2. If AD does make its queries, then C_A^u or C_B^v will accept their session and hold the key formed $h_2(C_A, C_B, sid, (recovered\ data)^{secretexponent})$. It is the fact that the session key g^{xyz_1} is unknown to AD", AD" cannot calculate this key actually.

AD" will search its L_{h1} and L_{h2} lists for the entry and certainly wins its experiment if Case 2 does happen really. Hence, the probability of AD" outputting the correct value on $g^{xyz_1} \bmod p$ is: $\Pr[Q_h]/(N_C^2 N_I^2)\} \geq \eta(k)/(N_C^2 N_I^2)$, where the probability is non-negligible and the result contradicts our CDHP assumption. Hence, we can conclude that $\eta(k)$ must be negligible and is the advantage of $Adv_{P,D}^{G,AD}(k, q_{fake\text{-}C})$. The theorem is proven.