具有强安全性的三方口令认证密钥交换协议

丁晓飞 马传贵
(郑州信息科技学院 郑州 450002)

摘 要 大部分口令认证密钥交换(PAKE)协议的设计者忽略了长期密钥泄露可能造成的危害.文中发现仅仅依靠口令的安全性设计可以抵抗口令泄露攻击的三方 PAKE 协议是不可能的,所以文中采取服务器通过公钥实现认证的方法,设计一个可以抵抗口令泄露攻击的强安全性协议,其在随机预示和理想密码模型下基于 ECGDH 假设具有前向安全的特性.

关键词 口令认证密钥交换;椭圆曲线理论;口令泄露攻击;前向安全;可证安全
中图法分类号 TP309 **DOI 号**: 10.3724/SP.J.1016.2010.00111

The Three-Party Password-Authenticated Key Exchange Protocol with Stronger Security

DING Xiao-Fei MA Chuan-Gui
(*Zhengzhou Information Science and Technology Institute*, *Zhengzhou* 450002)

Abstract Most password authenticated key exchange (PAKE) protocol designers ignored the attacks resulting from leakage of long-term secret keys (passwords). The authors find that it is impossible to design a secure three-party PAKE protocol only based on password against password-compromise impersonation (PCI) attack. So the authors assume that the server realizes entity authentication using his public-key, and propose a novel three-party PAKE protocol based on elliptic curve cryptosystem (ECC), which can resist PCI attack. Furthermore, the authors prove that the proposed protocol is forward secrecy under the ECGDH assumption in the random oracle and ideal cipher models.

Keywords PAKE; ECC; PCI attack; forward secrecy; provable security

1 Introduction

The password authenticated key exchange (PAKE) protocol enables two or more parties holding a memorable password to agree on a session key over an insecure public network in secure and authenticated manners. In other words, no adversary can obtain any information about session key and impersonate any participant during the protocol. Because of their easy-to-memorize property, many password-based authenticated key exchange protocols[1-4] were proposed in the last few years.

With dynamic diversity and development of communication environments such as mobile networks, it is not reasonable for client-client communications to exchange key in large scale communication environments. In recent years, one of the main concerns is to establish a secure channel between two clients with different passwords. The three-party PAKE protocols[5-10] solve this kind of questions effectively. Such protocols demand that each client shares a password with a trusted server. Thus with the help of the trusted server, two clients realize mutual authentication and secure communication.

Due to the low entropy of password, special care must be taken when designing protocols to ensure that both the password and the session key finally agreed upon are secret. In fact, the PAKE

收稿日期:2009-07-24;最终修改稿收到日期:2009-11-09.本课题得到国家"八六三"高技术研究发展计划项目基金(2007AA01Z431, SQ2008AA01Z3472853)资助.**丁晓飞**,女,1985 年生,硕士研究生,实习研究员,研究方向为信息安全. E-mail: xiaofei2812@163.com. **马传贵**,男,1962 年生,教授,博士生导师,主要研究领域为信息安全和协议分析.

protocols are always susceptible to password-guessing attacks and password-compromise impersonation (PCI) attack. Clearly, if an adversary $\mathcal{A}$ learns a client A's password, he can impersonate any other client to establish session key with A. The password (long-live secret key) compromises can lead to undesirable consequences at least until the corrupted entity A finds that his password was compromised. Thus, the PCI attack represents a serious and subtle threat, resistance to such attack is a very significant security property. Since most existing three-party PAKE protocols have many issues which incur vulnerability to PCI attack, it is worthwhile to design a protocol which can resist to PCI attack

As it is impossible to design secure three-party PAKE protocols in the symmetrical key setting against PCI attack[11], we consider to design the protocol in the public key setting. In this paper, we assume that the trusted server has a pair of public and private key to realize mutual authentication. In fact, the higher security is required by bank system, mobile E-commerce and so on which have used the public key. It is feasible under the development of hardware secure design (HSD) on network such as computation capability and EMS memory. Due to the low computation and storage costs of ECC, we present a novel three-party PAKE protocol based on elliptic curve. Under the elliptic curve gap Diffie-Hellman (ECGDH) assumption, the proposed protocol is provably secure in the random oracle and ideal cipher models. Furthermore, because the scheme is operating over a certain elliptic cure, the secret key size is relatively short. The novel protocol is efficient both in computation cost and communication cost for the client side.

The remainder of this paper is organized as follows. In Section 2, we recall the communication model, some security definitions of three-party password-authenticated key exchange protocols and the definition of the ECGDH assumption. In section 3 we propose a new three-party PAKE protocol using public-key cryptography to realize entity authentication and analyze the efficiency of protocol simply. Then in Section 4 we give rigorous security proof of the protocol in the random-oracle and ideal cipher models. Finally, we conclude this paper in Section 5.

2 Security Model

In this section we recall the formal model for PAKE protocols. It is composed of two main parts: the description of the ability of the adversary and the security definitions. The model builds upon the previous one presented in [1] and [2]. In this model, we allow the adversary to make adaptive *Corrupt* queries, which models a wider variety of attack scenarios such as PCI attack. Finally, we describe the definition of the ECGDH assumption.

2.1 Communication Model

In a three-party PAKE protocol P, there are two types of participants: clients and server. Client A and B share their passwords pw_A and pw_B with a trusted server S respectively. All passwords are chosen from the same small dictionary D of size N. We denote participant U's i-th instances as U^i. We denote the set of all clients as U, and the set of all servers as S.

In this model, it is assumed that an adversary $\mathcal{A}$ potentially control all communications in the network. During the execution of the protocol, 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. These queries from adversary are as follows:

$Execute(U_1^{i_1}, S_j, U_2^{i_2})$: This oracle query is used to simulate eavesdropping attack of the adversary. After querying the oracle, all messages interacted between participants during an execution of the protocol are returned to the adversary.

$SendClient(U^i, m)$: This query models an active attack against a client. It outputs 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 session keys by clients. It returns to the adversary the session key of client instance U^i, if the latter is defined.

$Corrupt(U^i)$: This query models the possibility of subverting a principal. It returns to the adversary the password of client instance U^i.

$Test(U^i)$: This query is used to measure the semantic security of the session key of client instance U^i, if the latter is defined. If the key is not defined, it returns $\perp$. Otherwise, it returns either the session key held by client instance U^i if $b=0$ or a random of key the same size if $b=1$.

2.2 Security Definitions

Notation. An instance U^i is said to be opened if a query $Reveal(U^i)$ has been made by the adversary, otherwise instance U^i is unopened. Instance

U^i is accepted if it goes into an accept mode after receiving the last expected protocol message, which also means it has enough information to generate the session key.

Partnering. The relationship of partnering is also defined through the session identifications *sid*. We say U_1^i and U_2^j are partners if the following conditions are satisfied: (1) Both U_1^i and U_2^j are accepted; (2) Both U_1^i and U_2^j share the same *sid*; (3) The partner identification for U_1^i is U_2^j and vice-versa; (4) No instance other than U_1^i and U_2^j accepts with a partner identification equal to U_1^i or U_2^j.

Forward Security-Fresh (FS-Fresh). We say an instance U^i is FS-Fresh if all of the following hold: it has been accepted; no *Reveal* query has been made to it or its partner and no *Corrupt* query has been asked since the beginning of the game.

FS-PAKE Security. Consider an execution of the key exchange protocol P by the adversary $\mathcal{A}$ in which the latter is given access to the *Execute*, *SendClient*, *SendServer*, *Corrupt* and *Test* oracle and asks at most one *Test* query to a FS-fresh instance of an honest client. Let b' be his output. Let S denote the event in which the adversary successfully guesses the hidden bit b used by *Test* oracle. The FS-PAKE advantage of $\mathcal{A}$ is defined as follows:

$$Adv_D^{\text{FS-PAKE}}(\mathcal{A}) = 2 \cdot Pr[S] - 1,$$
$$Adv_D^{\text{FS-PAKE}}(t,R) = \max_{\mathcal{A}} Adv_D^{\text{FS-PAKE}}(\mathcal{A}).$$

Where maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of oracle queries).

A three-party PAKE protocol is said to be FS-PAKE-secure if the advantage $Adv_D^{\text{FS-PAKE}}$ is only negligibly larger than kn/N, when n is number of active sessions and k is a security parameter.

2.3 Diffie-Hellman Assumptions

In this section, we recall the definition of the ECGDH assumptions which we use in the security proof of our protocol. Let $G=\langle P\rangle$ be a finite cyclic group of prime order q in an elliptic curve E.

Elliptic Curve Gap Diffie-Hellman assumption (ECGDH). The ECGDH assumption can be defined by considering experiment Exp^{ECGDH}: we choose two random elements $u, v \in Z_q^*$, compute $U=uP$ and $V=vP$, and then give U and V to an adversary $\mathcal{A}$. Moreover $\mathcal{A}$ can send some DDH oracle queries. Let K be the output of $\mathcal{A}$. If $Exp^{\text{ECGDH}}(\mathcal{A})$ outputs 1 if $K=ECGDH(U,V)=uvP$ and 0 otherwise. We denote $Succ_{G,D}^{\text{ECGDH}}(t)$ be the maximal success probability that $Exp^{\text{ECGDH}}(\mathcal{A})$ outputs 1 over all adversary running within time t. Fortunately, the ECGDH assumption is not stronger than the CDH assumption[7].

3 The Novel Three-party PAKE Protocol

In this section, we introduce a new three-party PAKE protocol based on ECC and evaluate the efficiency of the protocol briefly.

Description. Our three-party PAKE protocol is based on ECC. The full description is shown in Fig. 1, where pw_A and pw_B are password shared by A and the trusted server S, B and S respectively; *Pub* and *Pri* are public and private key of the trus-

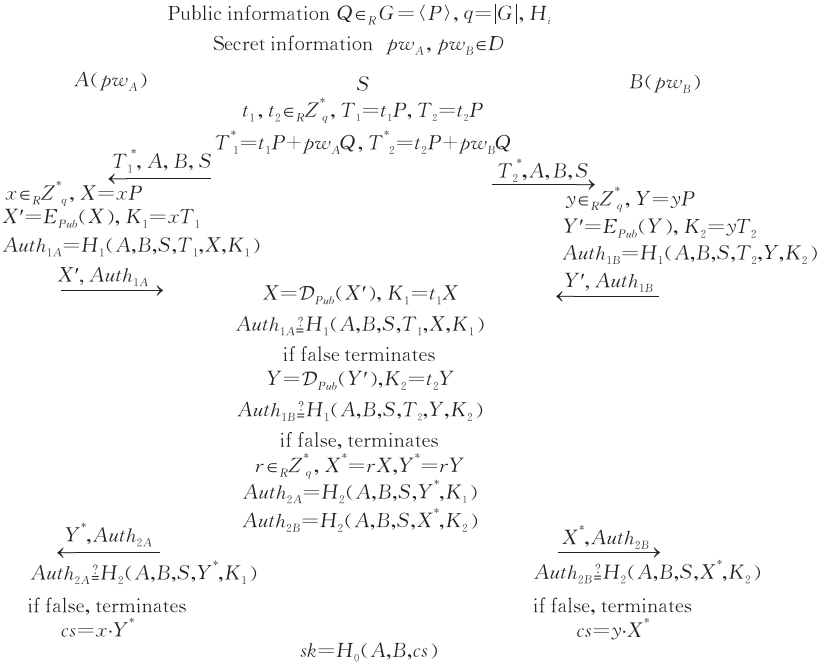


Fig. 1　Three-party PAKE protocol

ted server; $H_i:\{0,1\}^*\rightarrow\{0,1\}^{l_i}$ is hash function, $i\in\{0,1,2\}$; l_i is a security parameter.

At first, the client may send a request to the server to start the protocol. The protocol consists of the following three rounds of message.

(1) S chooses $t_1,t_2\in Z_q^*$ randomly, computes $T_1=t_1P$, $T_1^*=T_1+pw_AQ$, $T_2=t_2P$ and $T_2^*=T_2+pw_BQ$, then sends (A,B,S,T_1^*) and (A,B,S,T_2^*) to client A and B respectively.

(2) Client A obtains T_1 by decrypting T_1^*, chooses $x\in Z_q^*$ randomly and computes $X=xP$, $X'=E_{Pub}(X)$, $K_1=xT_1$ and $Auth_{1A}=H_1(A,B,S,T_1,X,K_1)$, then sends the message $(X',Auth_{1A})$ to the server S. Client B obtains T_2 by decrypting T_2^*, chooses $y\in Z_q^*$ randomly and computes $Y=yP$, $Y'=E_{Pub}(Y)$, $K_1=yT_2$ and $Auth_{1B}=H_1(A,B,S,T_1,Y,K_2)$, then sends the message $(Y',Auth_{1B})$ to the server S.

(3) S checks $Auth_{1A}$ and $Auth_{1B}$, if the authenticators are valid, chooses $r\in Z_q^*$ randomly and computes $X^*=rX$ and $Y^*=rY$, $Auth_{2A}=H_2(A,B,S,Y^*,K_1)$ and $Auth_{2B}=H_2(A,B,S,X^*,K_2)$, then sends $(Y^*,Auth_{2A})$ to the client A and $(X^*,Auth_{2B})$ to the client B.

Now A and B check $Auth_{2A}$ and $Auth_{2B}$ respectively, if the authenticator is valid, each client computes Diffie-Hellman key $cs=xY^*=yX^*=xyrP$, and then computes the session key $sk=H_0(A,B,cs)$, and accepts and terminates the execution of protocol.

Efficiency. With respect to efficiency, we denote the communication cost be measured in numbers of rounds and computational cost be measured in numbers of scalar multiplications (e.g. xP), encryptions/decryptions and hash operators. From the view of the client side, the proposed three-party PAKE protocol need only three communication rounds, whereas previous solutions require more than three communication rounds[9-10]. As for computational cost, the new protocol requires only three scalar multiplications, two encryptions/decryptions and two hash operators for each client, which saves at least one less than the solution in [10]. About the cost of certificate verification, each client only need three scalar multiplications and two hash. Because the trusted server's using the public key system, the protocol is not quite efficient for the server side.

4 Security of Protocol

From the protocol, we can find the proposed protocol is resistant to password-compromise impersonation attack. Clearly, if an adversary compromises password of client A, he wants to impersonate any other client to A, he needs to know xp. But he doesn't learn the private key of S, so he cannot obtain the xp by decrypting X'. Thus, our protocol is secure against the PCI attack. In this subsection, we prove that our protocol is FS-PAKE security under the ECGDH assumptions as the following theorem states. About the security of private key of the trusted server, we assume that the adversary cannot get it. If an adversary can obtain the private key of the trusted server, the capability of the adversary is too stronger that we don't consider.

Let $G=\langle P\rangle$ be a represent group of prime order q, where password is chosen from the dictionary D of size N. For any adversary $\mathcal{A}$ within a time bound t, with less than q_s active interactions with the parties (*Send*-queries) and q_p passive eavesdroppings (*Execute*-queries), q_e queries to encryption/decryption oracles, and asking q_h hash queries to H.

$$Adv_D^{\text{FS-PAKE}}(t,R)\leq\frac{(3q_S+3q_p+2q_e)^2}{q-1}+\frac{q_h^2+8q_S}{2^{l_1}}+\frac{q_h^2+8q_S}{2^{l_2}}+\frac{20q_S}{N}+48q_hSucc_{G,D}^{\text{ECGDH}}(t').$$

Where $t'\leq t+(q_s+q_p+q_e)\cdot\tau_e$, τ_e is the computational time for an exponentiation in G.

Proof. Security proof for the proposed three-party PAKE defines a sequence of hybrid experiments, starting with the real attack and ending in an experiment in which the adversary has no advantage. Each experiment addresses a different security aspect. For each experiment Exp_n, we define several events as followings:

(1) event S_n occurs if adversary $\mathcal{A}$ correctly guesses the bit b involved in the *Test* query;

(2) event $Encrypt_n$ occurs if $\mathcal{A}$ submits a data it has encrypted by itself using the password;

(3) event $Auth_n$ occurs if $\mathcal{A}$ submits an authenticator $Auth$ that will be accepted by the participators and that has been built by $\mathcal{A}$ itself.

Experiment Exp_0: This is the real protocol in the random oracle model. By definition, we have

$$Adv_D^{\text{FS-PAKE}}(t,R)=2\cdot Pr[S_0]-1.$$

Experiment Exp_1: In this experiment, we simulate the hash oracles (H_i, $i=0,1,2$, but also additional hash function H_i' which will be using later) as usual by maintaining hash lists Λ_{H_i} and $\Lambda_{H_i'}$ (see Figure 2). We also simulate all the instances, as the real players would do, for the *Send*-queries (see Figure 3) and for the *Execute*, *Reveal*, *Test* and *Corrupt* queries (see Figure 4). From this simulation, we easily see that the experiment is perfectly indistinguishable from the real

attack unless the permutation property of $\mathcal{E}$ and $\mathcal{D}$ does not hold. We have

$$|Pr[S_1]-Pr[S_0]|\leq\frac{q_\varepsilon^2}{2(q-1)}.$$

Where q_ε is the size of Λ_ε, which contains all encryption/decryption queries directly asked by the adversary and simulation, and thus $q_\varepsilon\leq q_e+q_s+q_p$.

For a hash query $H_i(q)$ ($H_i'(q)$) with $i=0,1,2$, for which there exists a record (i,q,r) in the list Λ_{H_i} ($\Lambda_{H_i'}$), return r. Otherwise the answer r is defined according to the following rule: **Rule $H^{(1)}$**. Choose an element $r\in\{0,1\}^{l_i}$. One adds the record (i,q,r) to the list Λ_{H_i} ($\Lambda_{H_i'}$). If the query is directly asked by the adversary, one adds (i,q,r) to $\Lambda_{\mathcal{A}}$.
For an encryption-query $\varepsilon_k(Z)$, such that a record $(k,Z,*,*,Z')$ in the list Λ_ε, return Z'. Otherwise the answer Z' is defined according to the following rule: **Rule $\mathcal{E}^{(1)}$**. Choose an element $Z'\in G$. One adds the record $(k,Z,\perp,\varepsilon,Z')$ to the list Λ_ε.
For a dncryption-query $\mathcal{D}_k(Z')$, such that a record $(k,Z,*,*,Z')$ in the list Λ_ε, return Z. Otherwise the answer Z is defined according to the following rule: **Rule $\mathcal{D}^{(1)}$**. Choose an element $\varphi\in Z_q^*$. Compute the answer $Z=\varphi P$ and add the record $(k,Z,\mathcal{D},\varepsilon,Z')$ to the list Λ_ε.

Fig. 2　Simulation of the hash oracles, and the encryption/decryption oracles

We answer to the *Send*-queries to a server as follows: A *Send* $(S^k,(U,start))$-query is processed according to the following rule: **Rule $S1^{(1)}$**. Choose two random exponent $t_1,t_2\in Z_q^*$, compute $T_1=t_1P$ and $T_1^*=t_1P+pw_1Q$, $T_2=t_2P$ and $T_2^*=t_2P+pw_2Q$, return (T_1^*,U_1,U_2,S) to U_1, (T_2^*,U_1,U_2,S) to U_2, and the instance goes to an expecting state. If the instance is in an expecting state, We apply the following rules: **Rule $S2^{(1)}$**. If the instance S^k receives the query *Send* $(S^k,((X',Auth_{1U_1}),(Y',Auth_{1U_2})))$, verify $Auth_{1U_1}$ and $Auth_{1U_2}$, if they are true, choose a random exponent $r\in Z_q^*$, compute $Y^*=rY$ and $Auth_{2U_1}=H_2(U_1,U_2,S,Y^*,K_1)$; $X^*=rX$ and $Auth_{2U_2}=H_2(U_1,U_2,S,X^*,K_2)$, return$(Y^*,Auth_{2U_1})$ to U_1, return $(X^*,Auth_{2U_2})$ to U_2. Finally, the instance terminates.
We answer to the *Send*-queries to the clients as follows: **Rule $U1^{(1)}$**. If the instance U_1^i receives the query *Send* $(U_1^i,(T_1^*,U_1,U_2,S))$, choose a random exponent $x\in Z_q^*$, then compute $X'=E_{Pub}(X)$, $K_1=xT_1$ and $Auth_{1U_1}=H_1(U_1,U_2,S,T_1,X,K_1)$, the query is answered with $(X',Auth_{1U_1})$. **Rule $U2^{(1)}$**. If the instance U_2^j receives the query *Send* $(U_2^j,(T_2^*,U_1,U_2,S))$, choose a random exponent $y\in Z_q^*$, then compute $Y'=E_{Pub}(Y)$, $K_2=xT_2$ and $Auth_{1U_2}=H_1(A,B,S,T_2,Y,K_2)$, the query is answered with $(Y',Auth_{1U_2})$. Our simulation also adds $((T_1^*,U_1,U_2,S),(T_2^*,U_1,U_2,S),(X',Auth_{1U_1}),(Y',Auth_{1U_2}),(Y^*,Auth_{2U_1}),(X^*,Auth_{2U_2}))$ to Λ_ψ. The variable Λ_ψ keeps track the exchange messages.

Fig. 3　Simulation of the *Send*-queries

On query *Reveal* (U^i), proceed as follows: If session key sk is defined for instance U^i, then return sk, else return $\perp$.
On query *Execute*(U_1^i,S^k,U_2^j), proceed as follows: $((T_1^*,U_1,U_2,S),(T_2^*,U_1,U_2,S))\leftarrow Send(S^k,(U,start)),(X',Auth_{1U_1})\leftarrow Send(U_1^i,((T_1^*,U_1,U_2,S))),$ $(Y',Auth_{1U_2})\leftarrow Send(U_2^j,(T_2^*,U_1,U_2,S)),((Y^*,Auth_{2U}),(X^*,Auth_{2U_2}))\leftarrow Send(S^k,(X',Auth_{1U_1}),(Y',Auth_{1U_2}))$. Returun $((T_1^*,U_1,U_2,S),(T_2^*,U_1,U_2,S),(X',Auth_{1U_1}),(Y',Auth_{1U_2}),(Y^*,Auth_{2U_1}),(X^*,Auth_{2U_2}))$.
A *Test* (U^i)query first gets sk from *Reveal* (U^i), and flips a coin b. If $b=1$, we return the value of the session key sk, otherwise we return a random value drawn from $\{0,1\}^{l_0}$.
On query *Corrupt* (U^i): return pw_U.

Fig. 4　Simulation of the *Reveal*, *Execute*, *Test* and *Corrupt*-queries

Experiment Exp_2: We define experiment Exp_2 by modifying the way the server process the *Send*-query so that the adversary will be the only one to encrypt data using the password. We use the following rule.

Rule $S1^{(2)}$. Choose a random $T_1^*\in G$ and $T_2^*\in G$, then compute $T_1=T_1^*-pw_1Q$ and $T_2=T_2^*-pw_2Q$. Look for the record $(pw_1,T_1,t_1,*,T_1^*)$ and $(pw_2,T_2,t_2,*,T_2^*)$ in the list Λ_ε to define t_1 and t_2 (we have $T_1=t_1P$ and $T_2=t_2P$).

The two experiments Exp_2 and Exp_1 are perfectly indistinguishable unless $t_1=\perp$ or $t_2=\perp$. This happens when T_1^* or T_2^* has been previously obtained as the ciphertext returned by an encryption-query:

$$|Pr[S_2]-Pr[S_1]|\leq\frac{2(q_S+q_p)q_\varepsilon}{q-1}.$$

Experiment Exp_3: For an easier analysis in the

following. We cancel experiments in which some collisions occur in the output of hash oracles, the encryption and decryption oracles, and the output of the *Send* queries. The probabilities are bounded by the birthday paradox:

$$|Pr[S_3]-Pr[S_2]| \leq \frac{q_h^2}{2^{l_1+1}}+\frac{q_h^2}{2^{l_2+1}}+\frac{2q_\varepsilon^2+(q_S+q_p)^2}{2(q-1)}.$$

Experiment Exp_4: We define experiment Exp_4 by aborting the executions wherein the adversary may have guessed the password and used it to send an encrypted data to the clients. We use the following rule to modify the way the clients processes the queries.

Rule $U1^{(4)}$. Look for $(pw_1, *, \perp, \varepsilon, T_1^*) \in \Lambda_\varepsilon$. If the record is found, define $Encrypt_4$ as true and abort the game. Otherwise, choose $x \in Z_q^*$, compute $X'=E_{Pub}(xP)$, $K_1=xT_1$, $Auth_{1U_1}=H_1(U_1, U_2, S, T_1, X, K_1)$.

Rule $U2^{(4)}$. Look for $(pw_2, *, \perp, \varepsilon, T_2^*) \in \Lambda_\varepsilon$. If the record is found, define $Encrypt_4$ as true and abort the game. Otherwise, choose $y \in Z_q^*$, compute $Y'=E_{Pub}(yP)$, $K_2=yT_2$, $Auth_{1U_2}=H_1(U_1, U_2, S, T_2, Y, K_2)$.

This two experiments are perfectly indistinguishable unless event $Encrypt_4$ occurs:

$$|Pr[S_4]-Pr[S_3]| \leq Pr[Encrypt_4].$$

Experiment Exp_5: We define Exp_5 by aborting the executions wherein the adversary may have been lucky in guessing the authenticator. We reach this aim by modifying the way the clients and server process the queries.

Rule $S2^{(5)}$. Check whether $Auth_{1U_1}=H_1(U_1, U_2, S, T_1, X, K_1)$ and $Auth_{1U_2}=H_1(U_1, U_2, S, T_2, Y, K_2)$. If the two equalities do hold, check if $((U_1, U_2, S, T_1, X, K_1), H_1) \in \Lambda_{\mathcal{A}}$ or $((T_1^*, U_1, U_2, S), (X', Auth_{1U_1})), H_1) \in \Lambda_\Psi$ and $((U_1, U_2, S, T_2, Y, K_2), H_1) \in \Lambda_{\mathcal{A}}$ or $(((T_2^*, U_1, U_2, S), (Y', Auth_{1U_2})), H_1) \in \Lambda_\Psi$. If these latter tests fail, then reject the authenticator: terminate, without accepting. If this rule does not make the server S to terminate, S accepts and moves on.

Rule $U1^{(5)}$. Check whether $Auth_{2U_1}=H_2(U_1, U_2, S, Y^*, K_1)$. If the equality does hold, check if $((U_1, U_2, S, Y^*, K_1), H_2) \in \Lambda_{\mathcal{A}}$ or $(((T_1^*, U_1, U_2, S), (Y^*, Auth_{2U_1})), H_2) \in \Lambda_\Psi$. If these two latter tests fail, then reject the authenticator: terminate, without accepting. If this rule does not make the client U_1 to terminate, U_1 accepts and moves on.

Rule $U2^{(5)}$. Check whether $Auth_{2U_2}=H_2(U_1, U_2, S, X^*, K_2)$. If the equality does hold, check if $((U_1, U_2, S, X^*, K_2), H_2) \in \Lambda_{\mathcal{A}}$ or $(((T_2^*, U_1, U_2, S), (X^*, Auth_{2U_2})), H_2) \in \Lambda_\Psi$. If these two latter tests fail, then reject the authenticator: terminate, without accepting. If this rule does not make the client U_2 to terminate, U_2 accepts and moves on.

The two experiments are perfectly indistinguishable unless the client rejects a valid authenticator. So we have

$$|Pr[S_5]-Pr[S_4]| \leq \frac{2q_S}{2^{l_1}}+\frac{2q_S}{2^{l_2}},$$

$$|Pr[Encrypt_5]-Pr[Encrypt_4]| \leq \frac{2q_S}{2^{l_1}}+\frac{2q_S}{2^{l_2}}.$$

Experiment Exp_6: We define Exp_6 by aborting the executions wherein the adversary may have guessed the password and then used it to build and send a valid authenticator to the clients. We reach this aim by modifying the way the clients and server process the queries.

Rule $S2^{(6)}$. Check if $((T_1^*, U_1, U_2, S), (X', Auth_{1U_1})), H_1) \in \Lambda_\Psi$ and $(((T_2^*, U_1, U_2, S), (Y', Auth_{1U_2})), H_1) \in \Lambda_\Psi$. If this is not the case, then reject the authenticator: terminate, without accepting. Check if $((U_1, U_2, S, T_1, X, K_1), H_1) \in \Lambda_{\mathcal{A}}$ and $((U_1, U_2, S, T_2, Y, K_2), H_1) \in \Lambda_{\mathcal{A}}$. If this is the case, we define the event $Auth_6$ to be true, and abort the game.

Rule $U1^{(6)}$. Check if $(((T_1^*, U_1, U_2, S), (Y^*, Auth_{2U_1})), H_2) \in \Lambda_\Psi$. If this is not the case, then reject the authenticator: terminate, without accepting. Check if $((U_1, U_2, S, Y^*, K_1), H_2) \in \Lambda_{\mathcal{A}}$. If this is the case, we define the event $Auth_6$ to be true, and abort the game.

Rule $U2^{(6)}$. Check if $(((T_2^*, U_1, U_2, S), (X^*, Auth_{2U_2})), H_2) \in \Lambda_\Psi$. If this is not the case, then reject the authenticator: terminate, without accepting. Check if $((U_1, U_2, S, X^*, K_2), H_2) \in \Lambda_{\mathcal{A}}$. If this is the case, we define the event $Auth_6$ to be true, and abort the game.

The two experiments are perfectly indistinguishable unless the event $Auth_6$ is not true. We have

$$|Pr[Encrypt_6]-Pr[Encrypt_5]| \leq Pr[Auth_6],$$
$$|Pr[S_6]-Pr[S_5]| \leq Pr[Auth_6].$$

Experiment Exp_7: In this experiment, we compute the authenticator $Auth$ and session key sk using oracles $H'_{1(2)}$ and H'_0 so that the values $Auth$ and sk are completely independent from $H_{1(2)}$ and H_0, but also pw_1, pw_2 and Diffie-Hellman key cs. Clearly, we compute them as follows: $Auth_{1U_1}=Auth_{1U_2}=H'_1(U_1, U_2, S)$, $Auth_{2U_1}=H'_2(U_1, U_2, S, Y^*)$, $Auth_{2U_2}=H'_2(U_1, U_2, S, X^*)$ and $sk=H'_0(U_1, U_2)$. Due to it, we do no longer need to compute the values K_1, K_2 and cs, we can postpone choo-

sing the values of the password $pw_{1(2)}$ until the adversary $\mathcal{A}$ asks the Corrupt query (at the very end if corruption never occurs).

The two experiments are perfectly indistinguishable unless the $AskH$ occurs: $\mathcal{A}$ queries the hash function H_1 on $(U_1, U_2, S, T_1, X, K_1)$ or $(U_1, U_2, S, T_2, Y, K_2)$, H_2 on (U_1, U_2, S, Y^*, K_1) or (U_1, U_2, S, X^*, K_2), H_0 on (U_1, U_2, cs). To avoid the trivial difference in the session on which $\mathcal{A}$ uses the password he corrupted to mount an active attack, we change the *Rule* H_i, $i=0,1,2$ as followings:

Rule $H_i^{(7)}$. If the event $AskH$ occurs before the *Corrupt* query, we choose elements $r_i \in \{0,1\}^{l_i}$ randomly. Otherwise, $\mathcal{A}$ knows the password, if $cs = ECCDH(X, Y^*) = ECCDH(Y, X^*)$ and $K_1(K_2) = ECCDH(X' - pw_1 Q(Y' - pw_2 Q), T_1(T_2))$, we set r_1 to $H_1'(U_1, U_2, S)$, r_2 to $H_2'(U_1, U_2, S, Y^*)$, r_3 to $H_2'(U_1, U_2, S, X^*)$ and r_0 to $H_0'(U_1, U_2)$, else we choose elements $r_i \in \{0,1\}^{l_i}$ randomly.

Now we still stimulates the hash oracle perfectly since we just replace some random values by other random values. Note the adversary can only ask *Test* query to instances which had been accepted before corrupting the password. Since the session key is computed with the random oracle H_0' that is private to the simulator before the corruption. We define the event $AskHBC_n$ occur if $\mathcal{A}$ ask $AskH_n$ before corrupting a participator. Thus we have

$$|Pr[Encrypt_7] - Pr[Encrypt_6]| \le Pr[AskHBC_7],$$
$$|Pr[Auth_7] - Pr[Auth_6]| \le Pr[AskHBC_7],$$
$$|Pr[S_7] - Pr[S_6]| \le Pr[AskHBC_7], Pr[S_7] = \frac{1}{2}.$$

From the lemma 2 of [8], we have:

$$Pr[Encrypt_7] \le \frac{2q_s}{N},\ Pr[Auth_7] \le \frac{4q_s}{N}.$$

Experiment Exp_8: In order to evaluate the event $AskHBC_7$, we introduce a random elliptic curve Diffie-Hellman instance (C, D). For a more convenient analysis, we consider from three aspects as follows.

Case 1. If the adversary asks H_1 on (U_1, U_2, S, T_1, X, Z) $(Z = ECCDH(X, T_1))$. We do not compute $Auth_{1A}$ with K_1, so we do not need to known the values of x and t_1.

Rule $U1^{(8)}$. Choose an random element $\alpha \in Z_q^*$ and compute $X = \alpha C$, then add the record (α, X) to Λ_C.

Rule D. Choose an random element $\beta \in Z_q^*$ and compute $T_1 = \beta D$, then add the record (β, T_1) to Λ_D. If $(*, X, *, *, X') \in \Lambda_\varepsilon$ then we abort the experiment; otherwise we add the record $(k, X, \perp, D, X')$ to Λ_ε.

By picking randomly in the Λ_A-list we can get the Diffie-Hellman secret value with probability $\frac{1}{q_h}$. We can simply look in the lists Λ_C and Λ_D to find the values α and β such that $X = \alpha C$ and $T_1 = \beta D$:

$$ECCDH(X, T_1) = ECCDH(\alpha C, \beta D) = \alpha\beta ECCDH(C, D)),$$

Thus, $Pr[\text{Case1}] \le q_h Succ_{G,D}^{\text{ECGDH}}(t')$.

Case 2. If the adversary asks H_1 on (U_1, U_2, S, T_2, Y, Z) $(Z = ECCDH(Y, T_2))$ or asks H_2 on $(U_1, U_2, S, Y^*(X^*), Y, Z(Z'))$ $((Z = ECCDH(X, T_1), (Z' = ECCDH(Y, T_2)))$. The situation is similar to Case1, so we have $Pr[\text{Case2}] \le 3q_h Succ_{G,D}^{\text{ECGDH}}(t')$.

Case 3. If the adversary asks H_0 on (U_1, U_2, Z) $(Z = ECCDH(Y^*, X) = CDH(X^*, Y))$. We do not compute sk with cs, so we do not need to known the values of x and y.

Rule $S^{(8)}$. Choose two random element $\gamma \in Z_q^*$ and $\delta \in Z_q^*$, compute $Y^* = \gamma C$ and $X^* = \delta C$, then add the record (γ, Y^*) to Λ_C and add (δ, X^*) to Λ_C.

Rule $U1(2)^{(8)}$. Choose an random element $\theta \in Z_q^*$ $(\rho \in Z_q^*)$ and compute $X = \theta D$ $(Y = \rho D)$, then add the record (θ, X) $((\rho, Y))$ to Λ_D. If $(*, X, *, *, X') \in \Lambda_E$ $((*, Y, *, *, Y') \in \Lambda_E)$ then we abort the experiment; otherwise we add the record$(k, X, \perp, D, X')$ $((k, Y, \perp, D, Y'))$ to Λ_E.

Similarly, we have:

$$ECCDH(Y^*, X) = ECCDH(\gamma C, \theta D) = \gamma\theta ECCDH(C, D)),$$
$$ECCDH(X^*, Y) = ECCDH(\delta C, \rho D) = \delta\rho ECCDH(C, D)).$$

Thus, $Pr[\text{Case3}] \le 2q_h Succ_{G,D}^{\text{ECGDH}}(t')$.

We find the experiments Exp_7 and Exp_8 are indistinguishable:

$$Pr[AskHBC_7] = Pr[AskHBC_8],$$
$$Pr[AskHBC_8] \le 6q_h Succ_{G,D}^{\text{ECGDH}}(t').$$

Combining all the above equations, one gets the announced result.

5 Conclusions

This paper presents a novel three-party password authenticated key exchange protocol based on elliptic curve cryptosystem in the public key setting, and proves its forward secrecy under the ECGDH assumption in the random oracle and ideal cipher oracles. To avoid password-compromise im-

personation attack, we assume that the trusted server already had public key system. Compared with previous solutions, the proposed three-party protocol is quite efficient both in computational cost and communication cost for the client side. But for the server side, the proposed three-party protocol is not so efficient, which is our next research issue.

References

[1] Bellare M, Rogaway P. Provably secure session key distribution: The three party case//Proceedings of the 27th ACM Symposium on Theory of Computing. New York, 1995: 57-66

[2] Bellare M, Pointcheval D, Rogaway P. Authenticated key exchange secure against dictionary attacks//Proceedings of Advances in Cryptology-Eurocrypt 2000, 2000: 139-155

[3] Boyko V, MacKenzie P, Patel S. Provably secure password authenticated key exchange using Diffie-Hellman//Proceedings of Advances in Cryptology-Eurocrypt 2000, 2000: 156-171

[4] Katz J, Ostrovsky R, Yung M. Eifficient password authenticated key exchange using human-memorable passwords//Proceedings of Advances in Cryptology-Eurocrypt 2001, 2001: 475-494

[5] Lin C, Sung H, Hwang T. Three-party encrypted key exchange: Attack and a solution. ACM Operating System Review, 2000, 34(4): 12-20

[6] Abdalla M, Chevassut O, Pointcheval D. One-time verifier-based encrypted key exchange//Proceedings of the PKC'05. Switzerland, 2005: 47-64

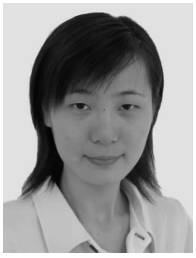
[7] Abdalla M, Pointcheval D. Simple password-based encrypted key exchange protocols//Proceedings of the RSA 2005, 2005: 191-208

[8] Bresson E, Chevassut O, Pointcheval D. Security proofs for an efficient password-based key exchange//Proceedings of the 10th CCS. New York, 2003: 241-250

[9] Wu S H, Zhu Y F. Three-party password authenticated key exchange with forward-security. Chinese Journal of Computers, 2007,30(10): 1833-1841

[10] Abdalla M, Fouque P A, Pointcheval D. Password-based authenticated key exchange in the three-party setting//Proceedings of the PKC'05. Switzerland, 2005: 65-84

[11] Xu J, Zhang Z F, Feng D G. Analysis and improvement of client-to-client password-authenticated key exchange protocols//Proceedings of the Security Protocols, 3rd SKLOIS Workshop. Beijing: Information Security and State, 2007: 119-124



DING Xiao-Fei, born in 1985, MS candidate. Her research direction is information security.

MA Chuan-Gui, born in 1962, professor, Ph. D. supervisor. His research interests include information security and protocal analysis.

Background

This work is supported by two grants from the National High Technology Research and Development Program of China (No. 2007AA01Z431, No. SQ2008AA01Z3472853).

Authenticated key exchange (AKE) protocols are importance manner to realize secure communication. AKE protocols are foundation of almost network protocols. Password-based authenticated key exchange (PAKE) protocols as one of AKE protocols get much more attention from cryptographists because their short length facilitates humans to remember and low entropy.

Password-based authenticated key exchange (PAKE) protocols enable two or more parties holding a memorable password to agree on a session key over a public network in secure and authenticated manner. With a rapid development of mobile network, PAKE protocols are widely used because their short length facilitates humans to remember. But due to absence of messages and lack of cryptogrammic protection, most PAKE protocols have hidden trouble. So how to design a secure PAKE protocol and prove its security is not only a work of great application significance, but also have wide application prospects.

With diversification of communication environment and fast application of new network technology, the ability of the adversary is greatly enhanced and attack methods are also constantly updated. There are new attacks in the PAKE protocols, such as password compromise impersonation (PCI) attack. To resist this attack, the authors mainly design novel and secure PAKE protocols in the asymmetric setting, which are competitive with the existing protocols in terms of security and efficiency.

Research issues of this paper focus on design and secrurity proof of password-based authenticated key exchange (PAKE) protocol. In this paper, the authors mainly consider the harm caused by the leakage of the ephemeral key and password compromise. In the unsymmetrical system, this paper proposes a novel three-party PAKE protocol based on elliptic curve cryptosystem, which can resist PCI attack. In the scheme, the clients can enhance encryption and authentication through the public key of the server. The computational cost and communication cost for clients is more efficient than existing alike protocols for clients. Furthermore, the authors prove that the new protocol is forward secrecy in the random oracle and ideal cipher model. As far as we know, there is not whole secure three-party PAKE protocol which can resist the PCI attack.