2012 26th IEEE International Conference on Advanced Information Networking and Applications

Efficient and Secure Cross-Realm Client-to-Client Password-Authenticated Key Exchange

Po-Jen Chuang and Yi-Ping Liao

Department of Electrical Engineering, Tamkang University
Tamsui, New Taipei City, Taiwan 25137, R.O.C.
E-mail: pjchuang@ee.tku.edu.tw

***Abstract*—To conduct secure communications in wireless networks, clients must create safer keys from the recorded less secure passwords – known as Password-Authenticated Key Exchange (PAKE). As attacker capability has evolved quickly, PAKE protocols must progress with time to fight against possible attacks. This paper makes an analytical survey on current cross-realm client-to-client (C2C) PAKE protocols and based on the Smart Card Framework Agreement develops a new and stronger C2C PAKE protocol to deal with malicious attacks. The new protocol involves client passwords, Smart Card information and server private keys to build a security protection mechanism which maneuvers by Mod calculation, Asymmetric encryption and Diffie-Hellman operations and is able to maintain communication security even when client passwords and server private keys are snatched. To verify the security of various C2C PAKE protocols – including ours, we employ Yoneyama's Security Model which can verify even Key-Compromise Impersonation (KCI) and Leakage of Ephemeral Private (LEP) attacks. Cost comparisons – covering calculation times and complexity – are also provided. The results show that our protocol achieves notably better security at reasonable cost.**

Keywords-client-to-client password-authenticated key exchange (C2C PAKE); cross-realm; smart cards; security models; performance evaluation.

I. INTRODUCTION

In a wireless network where packet transmission is carried out in an open environment, an adversary can easily eavesdrop, tamper or intercept routing packets to launch any forms of attacks. To secure communications, clients must establish more secure keys from the recorded less secure passwords by Password-Authenticated Key Exchange (PAKE). Figs. 1-3 present three basic PAKE structures – 2-party, 3-party and C2C. In 2-party PAKE, any two clients must use previously recorded passwords to authenticate with each other before setting up communication keys. If the system has numerous clients, this may produce burdens as clients need to record lots of client passwords. To improve it, 3-party PAKE lets two clients conduct mutual authentication via the server (which has all client passwords) to create communication keys. C2C PAKE advances one step further. It provides cross-realm communication: When two clients of different servers are to conduct communication, they will authenticate each other's identity via servers to establish communication keys. Compared with 3-party PAKE in which close clients also need to perform authentication via the remote server, C2C PAKE apparently works better.

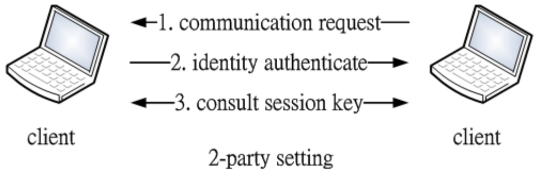


Fig. 1. 2-party PAKE.

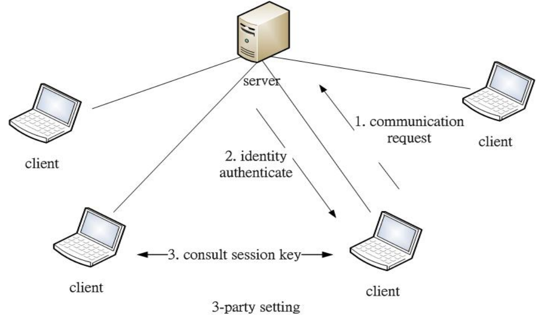


Fig. 2. 3-party PAKE.

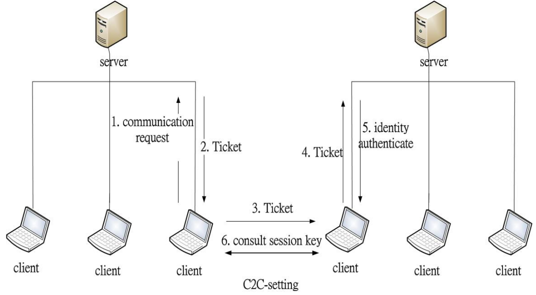


Fig. 3. C2C PAKE.

As the scope of malicious attacks has evolved quickly, we need more robust security-maintenance mechanisms, i.e., we need stronger communication protocols to resist malicious attacks even when critical information (such as passwords or server private keys) is grabbed. To tackle security problems in wireless networks, we first conduct an analysis on possible adversary attacks and also on recent C2C PAKE protocols to see their advantages/disadvantages. Based on the obtained analytical results, we then develop an advanced C2C PAKE protocol which employs client passwords, Smart Card information and server private keys to build a security protection mechanism. The key advantage of our new C2C PAKE protocol is, when both client passwords and the server private keys are snatched by adversaries, communication security can still be sustained.

1550-445X/12 $26.00 © 2012 IEEE
DOI 10.1109/AINA.2012.128

II. BACKGROUND STUDY

2.1 Possible Attacks

***dictionary attacks* (DAs):** An attacker continues to guess the password and verify the correct one through the return message. There are

(1) online DAs: An attacker guesses the password directly using key authentication with the server, and successfully interpret the return message when guessing right.

(2) undetectable online DAs: Similar to online DAs except that the server can not detect the ongoing attack.

(3) offline DAs: An attacker collects the client authentication packet by eavesdropping and calculates the client's password from the collected packet. (Both offline DAs and undetectable online DAs can lead to the more serious impersonation attacks.)

man-in-the-middle attacks: An attacker can join communication between two parties by tampering the authentication information, without being detected.

unknown key-share attacks: An attacker can authenticate with the server by modifying a client's identity and when the client thinks he is communicating with a fixed target, he is actually communicating with the attacker.

known-key attacks: An attacker fetching a communication key from a specific communication (e.g., by Denning-Sacco attacks) can actually use it to snatch information of other communications. A protocol able to resist such attacks is said to satisfy forward security.

Denning-Sacco attacks: When an (insider) attacker knows how to make a communication key, he can use the key and a target client's information (such as ID) to produce the client's communication key.

replay attacks: An attacker may intercept a client's authentication packet and use it to re-authenticate with the server. If passing certification, he can impersonate the client.

denial-of-service attacks: An attacker can paralyze a server by sending lots of meaningless messages to it.

impersonation attacks: An attacker obtains a client's password and uses the fake identity to attack other clients or the server. Without advanced identity checking mechanisms, such an attack is hard to resist.

password-compromise impersonation attacks: An attacker obtains the password of a client (say A), uses it to masquerade as other clients and communicate with A. It may happen to clients of general protocols who communicate based only on passwords. (Both password-compromise impersonation attacks and impersonation attacks are key-compromise impersonation (KCI) attacks).

2.2 Existing C2C PAKE Protocols

Some C2C PAKE protocols perform/support authentication (i.e., verify identity) by smart cards which can modify data by physical contact or induction, store clients' authentication information and perform encryption/decryption operations. A basic C2C PAKE protocol operates as follows. To initiate communication, client A first sends a communication request to the server. The server will send A a Ticket packet after verifying his identity. A then passes the Ticket to his target client at this communication attempt (say B). B will, in turn, send his authentication information along with the Ticket to the server. The server then verifies the identity of B by the received information and sends A negotiation information for him to negotiate a communication key with B. To facilitate later discussions, a brief introduction on major C2C PAKE protocols is provided below.

Byun's Protocol [6]: Being the first C2C PAKE protocol, it (Fig. 4) has two major problems:

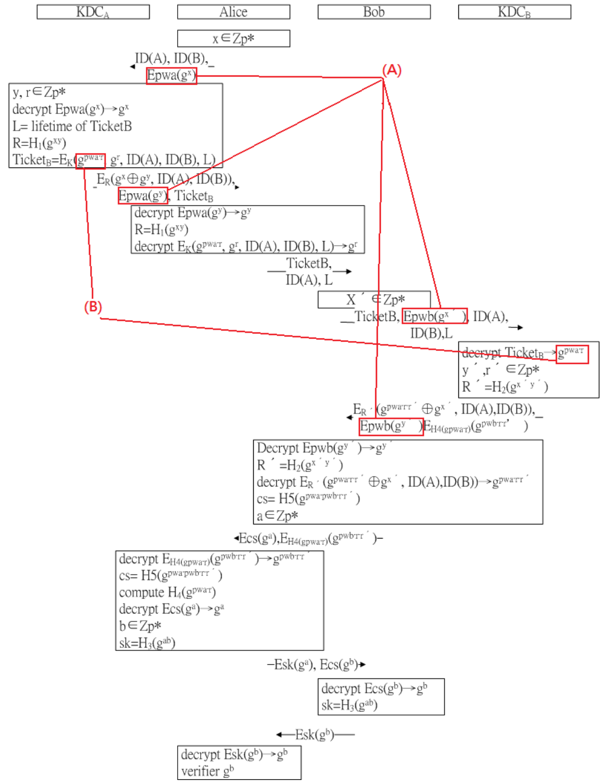
(i) As a large number of packets contain password information, an adversary can grab Epwa(gx), Epwa(gy), Epwb(gx') and Epwb(gy') from communication (Fig. 4 (A)) and use the information to conduct offline DAs.

(ii) An adversary can also use Ticket B which contains gpwa · r (Fig. 4 (B)) and subsequent negotiation information to conduct offline DAs.

Feng's Protocol [11]: To solve the problems, this protocol lets packets carry less password information and adds asymmetric encryption in Tickets and certification (Fig. 5 (A), involving both private and public keys). This protocol uses the password only once for authentication between the server and client, and because the packet comes from the server to the client (Fig. 5 (B)), an adversary can not guess the password by it, largely reducing possible offline DAs. Feng's Protocol has its own problems:

(i) Asymmetric encryption increases the operation cost.

(ii) In order to pass a packet that contains the password



KDC_A
Alice
Bob
KDC_B
$x \in Zp^*$
ID(A), ID(B), Epwa(g^x)
(A)
$y, r \in Zp^*$
decrypt Epwa(g^x) → g^x
L= lifetime of TicketB
$R=H_1(g^{xy})$
$Ticket_B=E_K(g^{pwa \cdot r}, g^r, ID(A), ID(B), L)$
$E_R(g^x \oplus g^y, ID(A), ID(B))$, Epwa($g^y$), $Ticket_B$
decrypt Epwa(g^y) → g^y
$R=H_1(g^{xy})$
decrypt $E_K(g^{pwa \cdot r}, g^r, ID(A), ID(B), L) \rightarrow g^r$
TicketB, ID(A), L
$X' \in Zp^*$
(B)
TicketB, Epwb($g^{x'}$), ID(A), ID(B),L
decrypt $Ticket_B \rightarrow g^{pwa \cdot r}$
$y', r' \in Zp^*$
$R'=H_2(g^{x'y'})$
$E_{R'}(g^{pwa \cdot r \cdot r'} \oplus g^{x'}, ID(A),ID(B))$, Epwb($g^{y'}$) $E_{H4(gpwa \cdot r)}(g^{pwb \cdot r \cdot r'})$
Decrypt Epwb($g^{y'}$) → $g^{y'}$
$R'=H_2(g^{x'y'})$
decrypt $E_{R'}(g^{pwa \cdot r \cdot r'} \oplus g^{x'}, ID(A),ID(B)) \rightarrow g^{pwa \cdot r \cdot r'}$
cs= H5($g^{pwa \cdot pwb \cdot r \cdot r'}$)
$a \in Zp^*$
Ecs(g^a), $E_{H4(gpwa \cdot r)}(g^{pwb \cdot r \cdot r'})$
decrypt $E_{H4(gpwa \cdot r)}(g^{pwb \cdot r \cdot r'}) \rightarrow g^{pwb \cdot r \cdot r'}$
cs= H5($g^{pwa \cdot pwb \cdot r \cdot r'}$)
compute $H_4(g^{pwa \cdot r})$
decrypt Ecs(g^a) → g^a
$b \in Zp^*$
$sk=H_3(g^{ab})$
Esk(g^a), Ecs(g^b)
decrypt Ecs(g^b) → g^b
$sk=H_3(g^{ab})$
Esk(g^b)
decrypt Esk(g^b) → g^b
verifier g^b

Fig. 4. Byun's Protocol.

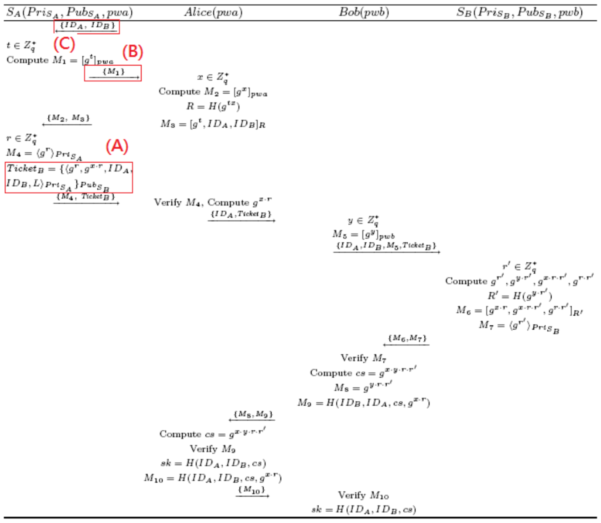

$S_A(Pris_A, Pubs_A, pwa)$
$Alice(pwa)$
$Bob(pwb)$
$S_B(Pris_B, Pubs_B, pwb)$
$\{ID_A, ID_B\}$
(C)
(B)
$t \in Z_q^*$
Compute $M_1 = [g^t]_{pwa}$
$\{M_1\}$
$x \in Z_q^*$
Compute $M_2 = [g^x]_{pwa}$
$R = H(g^{tx})$
$M_3 = [g^t, ID_A, ID_B]_R$
$\{M_2, M_3\}$
$r \in Z_q^*$
$M_4 = \langle g^r \rangle_{Pris_A}$
(A)
$Ticket_B = \{(g^r, g^{x \cdot r}, ID_A, ID_B, L)_{Pris_A}\}_{Pubs_B}$
$\{M_4, Ticket_B\}$
Verify M_4, Compute $g^{x \cdot r}$
$\{ID_A, Ticket_B\}$
$y \in Z_q^*$
$M_5 = [g^y]_{pwb}$
$\{ID_A, ID_B, M_5, Ticket_B\}$
$r' \in Z_q^*$
Compute $g^{r'}, g^{y \cdot r'}, g^{x \cdot r \cdot r'}, g^{r \cdot r'}$
$R' = H(g^{y \cdot r'})$
$M_6 = [g^{x \cdot r}, g^{x \cdot r \cdot r'}, g^{r \cdot r'}]_{R'}$
$M_7 = \langle g^{r'} \rangle_{Pris_B}$
$\{M_6, M_7\}$
Verify M_7
Compute $cs = g^{x \cdot y \cdot r \cdot r'}$
$M_8 = g^{y \cdot r \cdot r'}$
$M_9 = H(ID_B, ID_A, cs, g^{x \cdot r})$
$\{M_8, M_9\}$
Compute $cs = g^{x \cdot y \cdot r \cdot r'}$
Verify M_9
$sk = H(ID_A, ID_B, cs)$
$M_{10} = H(ID_A, ID_B, cs, g^{x \cdot r})$
$\{M_{10}\}$
Verify M_{10}
$sk = H(ID_A, ID_B, cs)$

Fig. 5. Feng's Protocol.

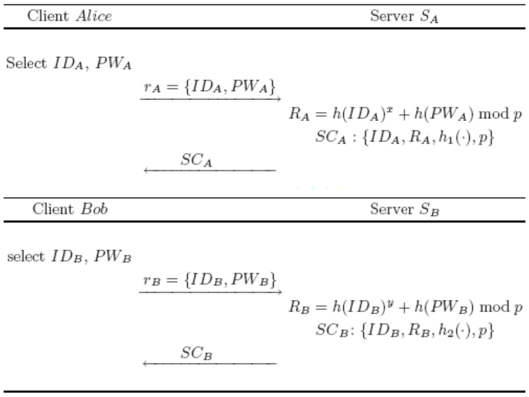


Fig. 6. Jin's Protocol (the Registration Phase).

from the server to the client, the client must uplink a request to the server – taking two extra transmissions.

(iii) The two extra transmissions, conducted to initiate negotiation only, are not certified and are hence prone to denial-of-service attacks.

Jin's Protocol [12]: This protocol (Figs. 6-7) does not use asymmetric encryption to satisfy KCI security. Instead, it employs the Smart Cards. It exchanges Smart Card information in the Registration Phase and uses the information to set up a communication key in the Login-and-Authentication Phase. Its advantages:

(i) As authentication in the Login-and-Authentication Phase is conducted based on Smart Card information which is previously established in the Registration Phase (Fig. 6) and involves no packets containing the passwords, offline dictionary attacks can be effectively avoided (Fig. 7 (A) contains no password PW).

(ii) Authentication is carried out not based on the password, so an adversary holding only the password but no Smart Card information can not forge server-client authentication. The design can effectively resist KCI attacks.

(iii) The server authenticates a client by storing only parameter x, not the password. Thus, the safety of the password can be secured even when the server is attacked (Fig. 7 (B): the server uses only x to calculate R_A").

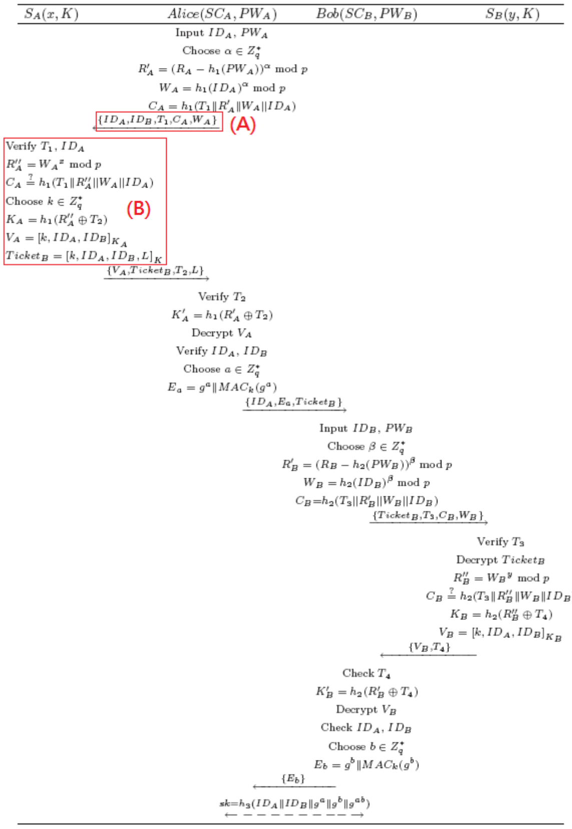

$S_A(x, K)$
$Alice(SC_A, PW_A)$
$Bob(SC_B, PW_B)$
$S_B(y, K)$
Input ID_A, PW_A
Choose $\alpha \in Z_q^*$
$R'_A = (R_A - h_1(PW_A))^{\alpha} \bmod p$
$W_A = h_1(ID_A)^{\alpha} \bmod p$
$C_A = h_1(T_1 \| R'_A \| W_A \| ID_A)$
$\{ID_A, ID_B, T_1, C_A, W_A\}$
(A)
Verify T_1, ID_A
$R''_A = W_A{}^x \bmod p$
$C_A \stackrel{?}{=} h_1(T_1 \| R''_A \| W_A \| ID_A)$
Choose $k \in Z_q^*$
(B)
$K_A = h_1(R''_A \oplus T_2)$
$V_A = [k, ID_A, ID_B]_{K_A}$
$Ticket_B = [k, ID_A, ID_B, L]_K$
$\{V_A, Ticket_B, T_2, L\}$
Verify T_2
$K'_A = h_1(R'_A \oplus T_2)$
Decrypt V_A
Verify ID_A, ID_B
Choose $a \in Z_q^*$
$E_a = g^a \| MAC_k(g^a)$
$\{ID_A, E_a, Ticket_B\}$
Input ID_B, PW_B
Choose $\beta \in Z_q^*$
$R'_B = (R_B - h_2(PW_B))^{\beta} \bmod p$
$W_B = h_2(ID_B)^{\beta} \bmod p$
$C_B = h_2(T_3 \| R'_B \| W_B \| ID_B)$
$\{Ticket_B, T_3, C_B, W_B\}$
Verify T_3
Decrypt $Ticket_B$
$R''_B = W_B{}^y \bmod p$
$C_B \stackrel{?}{=} h_2(T_3 \| R''_B \| W_B \| ID_B)$
$K_B = h_2(R''_B \oplus T_4)$
$V_B = [k, ID_A, ID_B]_{K_B}$
$\{V_B, T_4\}$
Check T_4
$K'_B = h_2(R'_B \oplus T_4)$
Decrypt V_B
Check ID_A, ID_B
Choose $b \in Z_q^*$
$E_b = g^b \| MAC_k(g^b)$
$\{E_b\}$
$sk = h_3(ID_A \| ID_B \| g^a \| g^b \| g^{ab})$

Fig. 7. Jin's Protocol (the Login-And-Authentication Phase).

Jin's Protocol faces two primary problems:

(i) Before it is put to work, each client must employ secure approaches to set up Smart Card information.

(ii) An attacker can still launch KCI attacks – using argument x, instead of client passwords (Fig. 7 (B): x alone can calculate R_A").

Ding's Protocol [10]: This protocol (Fig. 8) involves less cost to protect client passwords. The server will broadcast g^{α} and g^{β} for authentication, i.e., server-client authentication uses the password only once (Fig. 8 (A)) and the packet contains a random number so that an adversary can not guess at the password. Offline DAs can be effectively avoided. Its disadvantages include (i) the server needs to broadcast g^{α} and g^{β} constantly to inform the clients, and (ii) in the authentication phase from clients to the server, the server can not authenticate clients and is therefore vulnerable to denial-of-service attacks.

III. THE PROPOSED NEW PROTOCOL

This paper presents a new Smart Card protocol based on Jin's Protocol [12]. The operation of the new C2C PAKE protocol is introduced below.

3.1 Involving Mod Calculation

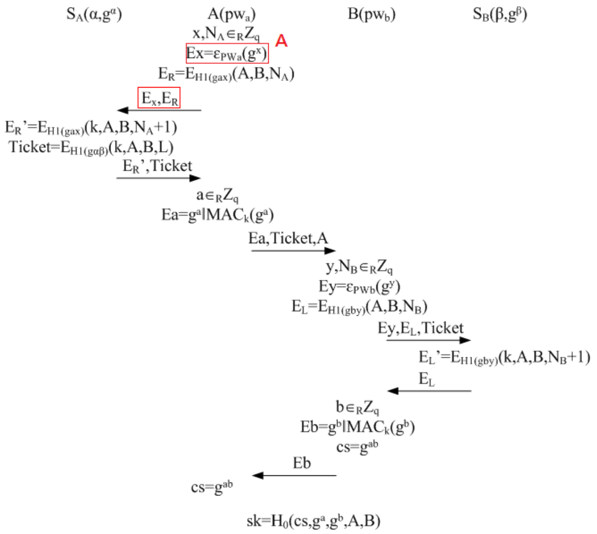


Fig. 8. Ding's Protocol.

As Fig. 9 exhibits, our protocol can resist off-line DAs because (1) $R_A' = R_A'' = h1(ID_A)^{\alpha x}$ mod p (the value of Mod Calculation remains equal after addition, multiplication and exponentiation – see the following calculation details), and (2) the transmission packets containing R_A' and W_A also contain random numbers and never use passwords after this phase.

$R_A' = (R_A - h1(PW_A))^{\alpha}$ mod p = $((h(ID_A)^x + h(PW_A)$ mod p) - $h1(PW_A))^{\alpha}$ mod p

$RA'' = W_A{}^x$ mod p = $(h1(ID_A)^{\alpha}$ mod p$)^x$ mod p

$RA' = RA'' = h1(ID_A)^{\alpha x}$ mod p

3.2 Employing the Diffie-Hellman Operations

Our protocol uses a set of Diffie-Hellman operations to enhance transmission security (Fig. 9) because

(1) an adversary with no password can not launch attacks (to intercept a transmission packet, it needs the password to get the result of the Diffie-Hellman operations), and

(2) the number of packets containing passwords will not grow, reducing the risk of DAs. (In Fig. 9 (E), (A) adds a Diffie-Hellman operation and uses G to encrypt packets. For easier cross-reference, the original Smart Card authentication packet [12] is shown in Fig. 10.)

By adding the Diffie-Hellman calculation and using parameters obtained from such a calculation to encrypt the authentication packet, our protocol can keep an adversary with no passwords from launching attacks. The involved additional cost will be 2 Diffie-Hellman calculations and 4 times of symmetric encryption.

3.3 Exchanging Parameters

Our protocol uses the same way as Ding's Protocol to exchange parameters. In Fig. 9 (D), the parameters that Ding's Protocol broadcasts are placed in the initial Smart Cards, reducing two times of extra transmissions when compared with Feng's Protocol (Fig. 11).

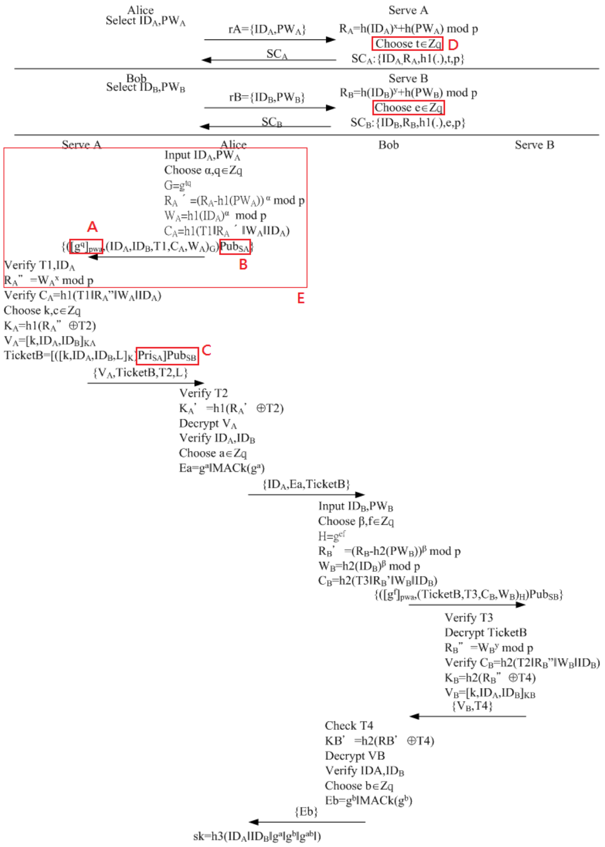


Fig. 9. Our new Smart Card Protocol.

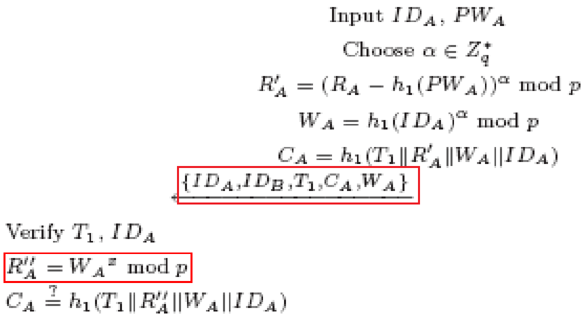

Input ID_A, PW_A
Choose $\alpha \in Z_q^*$
$R_A' = (R_A - h_1(PW_A))^{\alpha}$ mod p
$W_A = h_1(ID_A)^{\alpha}$ mod p
$C_A = h_1(T_1\|R_A'\|W_A\|ID_A)$
$\{ID_A,ID_B,T_1,C_A,W_A\}$
Verify T_1, ID_A
$R_A'' = W_A{}^x$ mod p
$C_A \overset{?}{=} h_1(T_1\|R_A''\|W_A\|ID_A)$

Fig. 10. The original Smart Card authentication packet.

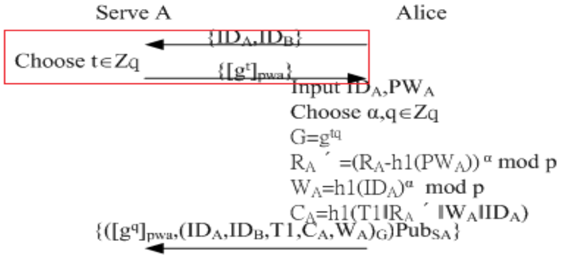

Serve A
Alice
$\{ID_A,ID_B\}$
Choose t∈Zq
$\{[g^t]_{pwa}\}$
Input ID_A,PW_A
Choose α,q∈Zq
$G=g^{tq}$
$R_A'=(R_A-h1(PW_A))^{\alpha}$ mod p
$W_A=h1(ID_A)^{\alpha}$ mod p
$C_A=h1(T1\|R_A'\|W_A\|ID_A)$
$\{([g^q]_{pwa},(ID_A,ID_B,T1,C_A,W_A)_G)Pub_{SA}\}$

Fig. 11. Reducing extra transmissions.

3.4 Using Asymmetric Encryption

Our protocol adds asymmetric encryption in both the authentication packet (Fig. 9 (B)) and the Ticket (Fig. 9 (C)) to ensure that, without the private key, an adversary can not crack the safety. In the Ticket, Pub_{SB} makes sure only server B can decrypt the packet, while Pri_{SA} guarantees the packet is sent from server A and an attacker can not decrypt it even when the shared key between servers (k) is compromised.

That is, our protocol can prevent an attacker from launching attacks without the private key at the cost of 4 asymmetric encryptions (2 pairs of encryption and decryption). On the other side, adding asymmetric encryption in the Ticket will make it difficult for adversaries to disguise as servers (at the cost of another 4 times of asymmetric encryption, i.e., 2 pairs of encryption and decryption).

3.5 Major Advantages

(1) An adversary cannot crack the security of server-client authentication without the password, the Smart Card (with parameter x) and the private key (Fig. 9 (A) and (B)).

(2) An adversary needs the private key and the shared key between servers (k) to decrypt the Ticket (Fig. 9 (C)).

IV. PERFORMANCE EVALUATION

4.1 Employing Appropriate Security Models

The performance of security protocols is subject to factors including length of the cipher text, encryption designs or password complexity. As different evaluation mechanisms may yield different results, a security protocol tends to assume an ideal calculation method and adopt a security model to simulate the attacker's acts – to prove its chance of having security cracks nears zero. Employing proper security models to attain fair performance comparison between different protocols is indeed essential. But, current models, which define attacker capability based mainly on random oracles and test only known attacks, can not verify unknown attacks or specific security loopholes. This paper decides to employ Yoneyama's Security Model in [17] to evaluate the performance of various protocols – because it can verify a number of attacks that the frequently used BR (Bellare-Rogaway) [1-4] and CK (Canetti-Krawczyk) [7,13] models fail to verify, such as KCI, LEP (Leakage of Ephemeral Private), UDonDA (undetectable online DAs), and offDA (offline DAs). A brief introduction on attacker capability is given below to facilitate later discussions.

Execute**:** An attacker disguises as clients A and B to create a communication connection with the server and uses the obtained information to initiate a DA.

SendClient**:** An attacker sends a forged packet to a client who then sends the calculated results back to the attacker following the specified requirements in the forged packet.

SendServer**:** An attacker sends a forged packet to the server who then returns the calculated results to the attacker following the specified requirements in the forged packet.

(*Execute*, *SendClient* and *SendServer* together can verify BR security, i.e., if an attacker can strike a general attack.)

StaticKeyReveal**:** An attacker can obtain the target static key information, such as (1) the password between a client and the server – to achieve a KCI attack or (2) the server's private key – to achieve an LEP attack.

SessionKeyReveal**:** An attacker can obtain a client's session key after the session is completed.

EphemeralKeyReveal**:** An attacker can obtain the target temporary key information, such as information to generate session keys.

EstablishParty**:** An attacker can directly register as a client on the server, attaining complete control over the client. A client not attacked by this oracle is called an honest client.

(*SessionKeyReveal*, *EphemeralKeyReveal* and *EstablishParty* together can verify if a protocol reaches forward security.)

Test**:** To test if an attacker can get the client's session key by guessing. The security model will randomly select an authentication bit: Test will return a session key if bit = 1 or a random number if bit = 0.

TestPassword**:** To test if an attacker can guess and get the client's password. If the guessed password is right, return 1; otherwise, return 0.

4.2 Evaluating the security of our C2C PAKE protocol

We use Yoneyama's Model to evaluate the security of our C2C PAKE protocol. In our assumption, if an attacker obtains the private key, he will not obtain K, pw or x at the same time; if unable to get the private key, he can get x, pw and K. Below are some experiments (Exp).

Exp0**:** Let Succ0 be the case that an attacker has guessed the correct authentication bit.

$$\mathrm{Adv}^{pake}{}_{t,D}(A) = 2\Pr[\mathrm{Succ0}] -1 \qquad \mathbf{(1)}$$

Exp1**:** Authenticate h1, h2, h3 and the ideal encryption and decryption ε and D. The probability that the attacker uses Send, Reveal and Execute to find the random numbers is $(q_E{}^2+q_{h1}{}^2+q_{h2}{}^2+q_{h3}{}^2)/2(q-1)$.

$$\Pr[\mathrm{Succ0}]-\Pr[\mathrm{Succ1}]|\leq(q_E{}^2+q_{h1}{}^2+q_{h2}{}^2+q_{h3}{}^2)/2(q-1)\mathbf{(2)}$$

Exp2**:** Replace W_A by a random number. The probability that an attacker can distinguish W_A from the random number equals the probability he can verify W_A by obtaining x, which will be $\mathrm{Adv}^{dl}{}_G(T_{dl})$. Then, there are two cases.

Case1: The attacker obtains the private key → The probability of successfully verifying W_A = the probability of using the password to get g^t and crack DDH = $q_{StaticKey} \cdot \mathrm{Adv}^{dl}{}_G(T_{dl}) \cdot \mathrm{Adv}^{ddh}{}_G(T_{ddh})$.

Case2: The attacker obtains the password → The probability of successfully verifying W_A = the probability of cracking DDH and asymmetric encryption = $q_{StaticKey} \cdot \mathrm{Adv}^{dl}{}_G(T_{dl}) \cdot \mathrm{Adv}^{cca2}{}_G(T_{cca2})$.

By the above two formula, we have

$$|\Pr[\mathrm{Succ1}]-\Pr[\mathrm{Succ2}]|\square q_{StaticKey} \cdot \mathrm{Adv}^{dl}{}_G(T_{dl}) \cdot (\mathrm{Adv}^{ddh}{}_G(T_{ddh}) + \mathrm{Adv}^{cca2}{}_G(T_{cca2})) \qquad \mathbf{(3)}$$

Exp2 is conducted mainly to verify the probability that an attacker can crack the authentication packet $\{([g^q]_{pwa},(ID_A,ID_B,T1,C_A,W_A)_G)Pub_{SA}\}$. In Case1, the attacker gets the private key by $q_{StaticKey}$, breaks symmetric encryption by $\mathrm{Adv}^{dl}{}_G(T_{dl})$, and cracks DDH by $\mathrm{Adv}^{ddh}{}_G(T_{ddh})$. In Case2, the attacker gets the password by $q_{StaticKey}$ and cracks asymmetric encryption by $\mathrm{Adv}^{cca2}{}_G(T_{cca2})$.

Exp3**:** We now use a random number to replace k in the Ticket and get 2 cases.

Case1: The attacker gets the private key → the probability of successfully verifying k = the probability of

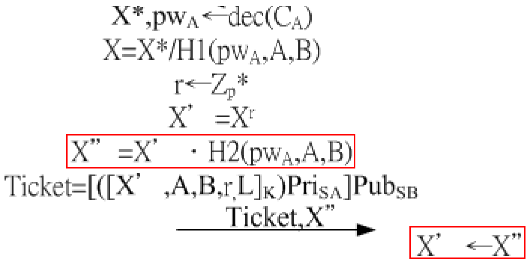

$X^*, pw_A \leftarrow dec(C_A)$
$X = X^*/H1(pw_A, A, B)$
$r \leftarrow Z_p^*$
$X' = X^r$
$\boxed{X'' = X' \cdot H2(pw_A, A, B)}$
$Ticket = [([X', A, B, r, L]_K)Pri_{SA}]Pub_{SB}$
$Ticket, X''$
$\boxed{X' \leftarrow X''}$

Fig. 12. AidkeyC2C Protocol's weakness.

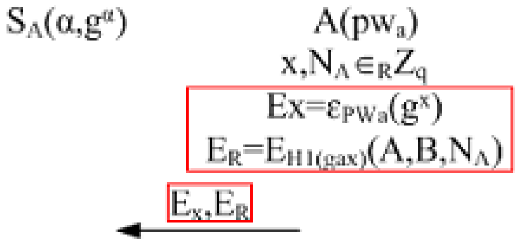

$S_A(\alpha, g^\alpha)$
$A(pw_a)$
$x, N_A \in_R Z_q$
$Ex = \varepsilon_{PW_a}(g^x)$
$E_R = E_{H1(gax)}(A, B, N_A)$
E_X, E_R

Fig. 13. Ding's Protocol's weakness.

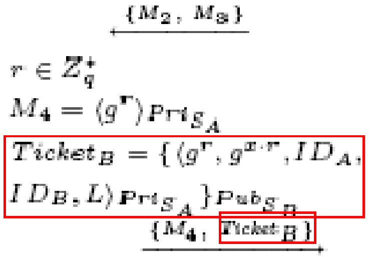

$\{M_2, M_3\}$
$r \in Z_q^*$
$M_4 = (g^r)Pri_{S_A}$
$Ticket_B = \{(g^r, g^{x \cdot r}, ID_A, ID_B, L)Pri_{S_A}\}Pub_{S_B}$
$\{M_4, Ticket_B\}$

Fig. 14. Feng's Protocol's weakness.

breaking the encryption and decryption $q_{StaticKey} \cdot Adv^{cca}_{SE}(T_{se}, q_e, q_d)$.

Case2: The attacker gets K → the probability of successfully verifying k = the probability of breaking the asymmetric encryption $q_{StaticKey} \cdot Adv^{cca2}_{G}(T_{cca2})$.

As the Ticket is used 3 times, we have

$$|Pr[Succ2] - Pr[Succ3] \square 3q_{StaticKey} \cdot (Adv^{cca}_{SE}(T_{se}, q_e, q_d) + Adv^{cca2}_{G}(T_{cca2})) \qquad \textbf{(4)}$$

Exp3 may verify the probability that an attacker can crack the Ticket. In Case1, the attacker gets the private key by $q_{StaticKey}$ and cracks the symmetric encryption of K by $Adv^{cca}_{SE}(T_{se}, q_e, q_d)$. In Case2, the attacker gets K by $q_{StaticKey}$ and cracks on asymmetric encryption $Adv^{cca2}_{G}(T_{cca2})$.

Exp4: Used to verify the probability that an attacker can counterfeit MAC, which will be $Adv^{cma}_{MAC}(T_{mac}, q_t, q_v)$. MAC is used 2 times (Ea and Eb), so we get

$$|Pr[Succ3] - Pr[Succ4]| \square 2Adv^{cma}_{MAC}(T_{mac}, q_t, q_v) \qquad \textbf{(5)}$$

Exp5: Replace DDH by $U = g^u$, $V = g^v$ and $Z = g^r$. The probability that an attacker can distinguish DDH from U, V and Z equals the probability he can crack DDH, i.e., $Adv^{ddh}_{G}(T_{ddh})$. Thus

$$|Pr[Succ4] - Pr[Succ5]| \square Adv^{ddh}_{G}(T_{ddh}) \qquad \textbf{(6)}$$

The probability that *Exp5* succeeds actually equals the probability that the attacker guesses sk by ways not mentioned above – including at least the probability of using Corrupt(C^1,2) and online DAs $q_{send}/2|D|$. Thus,

$$Pr[Succ5] \le (q_{SendClient} + q_{SendServer})/2|D| + 1/2 \qquad \textbf{(7)}$$

By(1)~(7), we have

$Adv^{ake}_{D}(t,R) \le (q_E^2 + q_{h1}^2 + q_{h2}^2 + q_{h3}^2)/(q-1) + q_{StaticKey} \cdot Adv^{dl}_{G}(T_{dl}) \cdot (Adv^{ddh}_{G}(T_{ddh}) + Adv^{cca2}_{G}(T_{cca2})) + 3q_{StaticKey} \cdot (Adv^{cca}_{SE}(T_{se}, q_e, q_d) + Adv^{cca2}_{G}(T_{cca2})) + 2Adv^{cma}_{MAC}(T_{mac}, q_t, q_v) + Adv^{ddh}_{G}(T_{ddh}) + (q_{SendClient} + q_{SendServer})/2|D| + 1/2$.

In ideal conditions, the chance for an attacker to break down all encryption mechanisms is nearly none. Note that both the dictionary/password length and the number of oracles the attacker uses are as large as infinite. When q and D approach *infinite* while $Adv^{dl}_{G}(T_{dl})$, $Adv^{ddh}_{G}(T_{ddh})$, $Adv^{cca2}_{G}(T_{cca2})$, $Adv^{cca}_{SE}(T_{se}, q_e, q_d)$ and $Adv^{cma}_{MAC}(T_{mac}, q_t, q_v)$ are approximately zero, $Adv^{pake}_{t,D}(A) = 2 \cdot 1/2 - 1$ – a near zero value, indicating an attacker has almost no chance to break this protocol under ideal conditions. According to this security model, an attacker can eavesdrop, send packets and obtain sk to launch a general attack or break BR security, or obtain a static key (that also contains the server's private key) to reach KCI and LEP attacks. That is, this model can verify if a protocol is tough enough to maintain BR+ KCI+LEP security, not just the general BR security.

Table I. Calculation times and security comparisons

	Transmission Times	Symmetric Calculation	Asymmetric Calculation	Security
AidkeyC2C	6	18	8	DA+BR+FS
Ding	6	36	0	DA+BR+FS
Jin	6	30	4	DA+BR+FS+KCI(Incomplete)
Feng	9	34	8	DA+BR+FS+KCI
OURs	6	44	12	DA+BR+FS+KCI+LEP

Table II. Calculation cost comparison – complexity

Calculation Type	Version	Complexity	
Asymmetric encryption	RSA[16]	encryption	$O((\log_2 e)(\log_2 n)^2)$
		decryption	$O((\log_2 d)(\log_2 n)^2)$
Symmetric encryption	AES[9]	$O(\log_2 n)$	
Hash	SHA1[15]	$O(\log_2 n)$	
Mod	[8]	$O(\log^2 n)$	
Diffie-Hellman	[5]	$O(vp)$	
MAC	MD5[14]	$O(\log_2 n)$	

Table III. Cost comparison among protocols – complexity

	Complexity
AidkeyC2C	$14O(\log_2 n) + 4(vp) + 4O((\log_2 d)(\log_2 n)^2) + 4O((\log_2 e)(\log_2 n)^2)$
Ding	$22O(\log_2 n) + 14O(vp)$
Jin	$24O(\log_2 n) + 12O(vp) + 6O(\log^2 n)$
Feng	$18O(\log_2 n) + 16O(vp) + 4O((\log_2 d)(\log_2 n)^2) + 4O((\log_2 e)(\log_2 n)^2)$
Ours	$32O(\log_2 n) + 18O(vp) + 4O((\log_2 d)(\log_2 n)^2) + 4O((\log_2 e)(\log_2 n)^2) + 6O(\log^2 n)$

4.3 The Security of Other Protocols

AidkeyC2C attains BR but not KCI security as password leakage may cause attacks (an attacker can counterfeit X" by the password – Fig. 12), and so does Ding's Protocol (an attacker can use the obtained password to decrypt E_X and E_R – Fig. 13). Jin's Protocol can reach KCI security only if its Smart Card is secure; if not (e.g., password leakage), it is vulnerable to KCI attacks. For Feng's Protocol, a safe

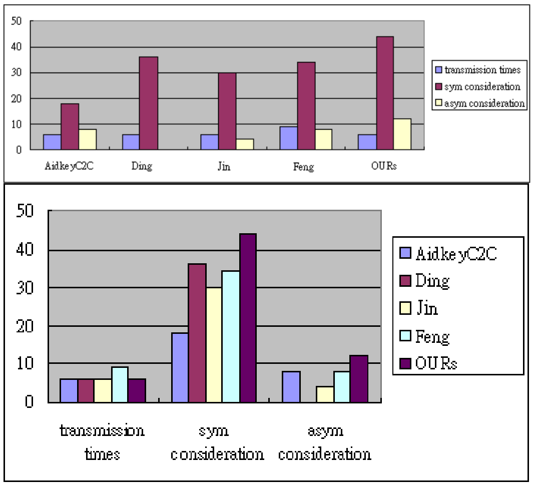


Fig. 15. Calculation cost comparison.

private key will ensure KCI security, but a snatched private key will lead to possible LEP attacks (the Ticket in Fig. 14 is encrypted by both private and public keys).

4.4 Security and Cost Comparisons

Table I gives a handy look on security and cost of various protocols, including AidkeyC2C, Ding's, Jin's, Feng's and ours. Note that, when considering cost, we take both calculation times and complexity into account. The cipher text length and cost for symmetric/asymmetric encryption are each in linear/exponential relationship. Mod is similar to asymmetric encryption – its cipher text length and cost are also in exponential relationship. Hash divides the cipher text into blocks and cost will increase when blocks increase – in linear relationship. MAC resembles asymmetric one-way Hash and is hence taken as symmetric calculation. That is, we consider both Hash and MAC symmetric calculation, while Mod asymmetric calculation.

The calculation cost indicates the total amount of calculations taken to negotiate and get one session key (the results in Table I are average numbers). In the encryption process, each encryption or decryption is counted as one calculation. The Smart Card is established only once, so its cost is negligible. Table I shows that each protocol reaches the mentioned security: The BR model contains forward security (FS) and all protocols reach FS and DA security as they can resist off-line DAs.

In Fig. 15, Feng's Protocol takes more transmissions than others because it needs to change DDH's parameters. Not using asymmetric calculation (which costs much more than symmetric calculation) enables Ding's Protocol to reach security (but not KCI and LEP security) at lower cost. Here, we focus on asymmetric calculation times. From both Table I and Fig. 15, we find the increase in cost and security is in linear relationship for all protocols but AidkeyC2C. This reveals the asymmetric calculations involved in AidkeyC2C are unreasonable. The linear relationship for our protocol, on the other hand, justifies our involving of additional cost to get the valuable security gain.

Table II lists complexity comparisons (the results are calculated and obtained based on [16]) for some frequently-used calculations, including asymmetric algorithm–RSA, symmetric algorithm–AES (advanced encryption standard), Hash–SHA1, MAC–MD5, Mod and the Diffie-Hellman algorithm. Cost comparisons are given in Table III in which

n = password length

d and p = exponent parameter length for exponent calculation (set as 16)

e = a random number greater than 105 (set as 105 here).

The complexity versus password lengths for various operations is depicted in Fig. 16. In Fig. 16, we can see that the complexity of asymmetric encryption and decryption rises while password length grows.

Fig. 17 shows that cost difference is apparently larger between protocols with or without asymmetric encryption than between protocols with asymmetric encryption. For protocols with asymmetric encryption, Jin's takes 4 times of Mod, Feng's and AidkeyC2C take 8 times of asymmetric encryption, while ours involves 8 times of asymmetric encryption plus 4 times of Mod (Mod is indeed a form of asymmetric calculation with less complexity). That explains why Feng's, AidkeyC2C and our protocol (all involve 8 times of asymmetric encryption) do not generate large cost difference. The result further justifies our choice of using reasonable extra cost to trade for desirable security gain – such as LEP security.

4.5 Practical Applications

(1) Handheld devices or diskless workstations

PAKE can be widely used in practical applications, especially in handheld devices and diskless workstations. This is because C2C PAKE does not ask a client to connect to all servers for all the time; instead it will establish a connection only when necessary – to prevent possible attacks and as a result achieve more desirable communication security. When new clients join the network, original clients do not need any extra information because the protocol will process security authentication. Besides, clients can directly communicate with the server and therefore handily exchange or update the Smart Card data.

(2) Kerberos workstations

Having similar structures as Kerberos, C2C PAKE can be used to upgrade the system security of a Kerberos workstation, which creates a password only at the beginning of a secure communication with an honest server – Smart Card data can be exchanged at this time. As the password will remain the same for a long period, it is necessary and desirable to establish a new authentication mechanism independent of the password – by the Smart Card function – to tolerate the risk of password leakage. That is, the Smart Card function in our new protocol can help servers secure better communications.

(3) Commercial or medical workstations

As our protocol reaches higher security, it fits better for workstations requiring intensive security, such as those involving commercial transactions or medical practices. For instance, when clients are engaging banking or medical activities, they can meanwhile create or update the Smart Card to ensure security passage.

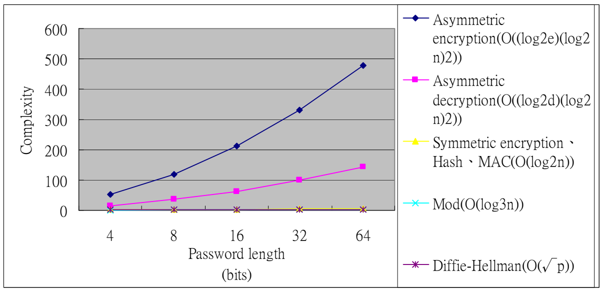


Fig. 16. Calculation cost comparison – complexity.

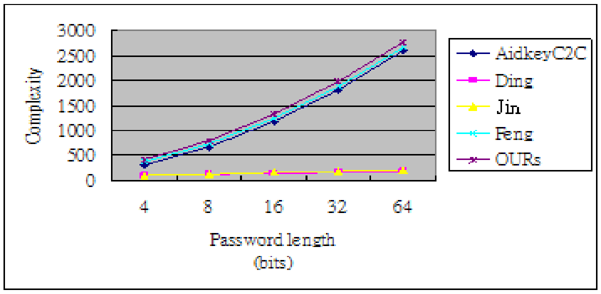


Fig.17. Complexity comparison among various protocols

V. CONCLUSIONS

To fight against the ever-growing malicious attacks in today's wireless environments, i.e., to attain more desirable communication security for wireless networks, this paper presents a new and advanced cross-realm C2C PAKE protocol – based on Jin's Protocol that allows clients to exchange Smart Card information and attain authentication by Mod Calculation. Aided by Mod Calculation, a set of Diffie-Hellman operations, Ding's parameter exchange approach and asymmetric encryption, our new protocol employs client passwords, Smart Card information and server private keys to form a strong security protection mechanism. The key advantage of our protocol is, when both client passwords and server private keys are snatched by adversaries, communication security can still be sustained. Security evaluation (by Yoneyama's Model) and cost comparison (in terms of calculation times and complexity) show that, at reasonable cost, our C2C PAKE protocol outperforms related protocols in security gain – being able to defend BR, KCI and even LEP attacks.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council, Taiwan, R. O. C., under Grant No. NSC 100-2221-E-032-063.

REFERENCES

[1]. M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," Proc. Public Key Cryptography'05, 2005, LNCS 3386, pp. 65-84.

[2]. M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Proc. Advances in Cryptology - EUROCRYPT 2000, May 2000, LNCS 1807, pp. 140-156.

[3]. M. Bellare, and P. Rogaway, "Entity authentication and key distribution," Proc. Advances in Cryptology - CRYPTO '93, 1994, LNCS 773, pp. 232-249.

[4]. M. Bellare, and P. Rogaway, "Provably secure session key distribution - the three party case," Proc. 28th Annual ACM Symp. on Theory of Computing, May 1996, pp.57-66.

[5]. D. Boneh, "The decision Diffie-Hellman problem," Proc. 3rd Algorithmic Number Theory Symposium, 1998, LNCS 1423, pp. 48-63.

[6]. J. W. Byun, I. R. Jeong, D. H. Lee, and C.-S. Park, "Password-authenticated key exchange between clients with different passwords," Proc. 4th Information and Communications Security, Dec. 2002, LNCS 2513, pp.134-146.

[7]. R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," Proc. 2001 Advances in Cryptology - EUROCRYPT 2001, 2001, pp 451-472.

[8]. T. Coffee, "Best kept secrets: elliptic curves and modern cryptosystems," MIT 18.704, Fall 2004.

[9]. J. Daemen and V. Rijmen, The Design of Rijndael: AES-The Advanced Encryption Standard, Springer-Verlag, 2002.

[10]. X. Ding and C. Ma, "Cryptoanalysis and improvements of cross-realm C2C-PAKE protocol," Proc. 2009 WASE Int'l Conf. on Information Engineering, 2009, pp. 193-196.

[11]. D.-G. Feng and J. Xu, "A new client-to-client password-authenticated key agreement protocol," Proc. 2009 Int'l Workshop on Coding and Cryptology, 2009, LNCS 5557, pp. 63–76.

[12]. W. Jin and J. Xu, "An efficient and provably secure cross-realm client-to-client password-authenticated key agreement protocol with smart cards," Proc. 2009 Int'l Conf. on Cryptology and Network Security, 2009, LNCS 5888, pp. 299-314.

[13]. H. Krawczyk, "HMQV: a high-performance secure Diffie-Hellman protocol," Proc. Advances in Cryptology - CRYPTO'05, 2005, LNCS 3621, pp. 546-566.

[14]. R. L. Rivest, "RFC 1321: The md5 message-digest algorithm," Technical Report, Internet Activities Board, April 1992.

[15]. X. Wang, Y. Yin, and H. Yu, "Finding collisions in the full SHA-1," Proc. Advances in Cryptology - CRYPTO'05, 2005, LNCS 3621, pp. 17-36.

[16]. W. P. Wardlow, "The RSA public key cryptosystem," Proc. 1991 Coding Theory and Cryptography, 1991, pp. 101-124.

[17]. K. Yoneyama, "Efficient and strongly secure password-based server aided key exchange," Proc. INDOCRYPT 2008, 2008, LNCS 5365, pp. 172–184.