Employing Secure and Efficient Password-Authenticated Key Exchange in Wireless Networks

Po-Jen Chuang* and Yi-Ping Liao
Department of Electrical and Computer Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.

Abstract

The password-authenticated key exchange (PAKE) is an important tool to secure wireless communications. To counter possible malicious attacks in wireless communications, this paper develops a stronger new cross-realm client-to-client (C2C) PAKE protocol based on the smart card framework agreement. Employing the client passwords, smart card information and server private keys, the new PAKE protocol works by the Mod calculation, Asymmetric encryption and Diffie-Hellman operations. It can practically enhance the security of wireless communications even when both client passwords and server private keys are snatched. To verify the performance of the new protocol, we bring in the Yoneyama’s security model which can verify very intrigue attacks (including key-compromise impersonation and leakage of ephemeral private attacks) to check the security levels of existing C2C PAKE protocols and our protocol. The collected cost comparison results show that, in contrast to other protocols, our new protocol yields notably better security gain at very reasonable cost.

Key Words: Wireless Networks, Client-to-Client Password-Authenticated Key Exchange (C2C PAKE), Cross-Realm, Smart Cards, Security Models, Performance Evaluation

1. Introduction

In wireless communications, an adversary can easily eavesdrop, tamper or intercept routing packets to launch possible attacks because packets are transmitted in an open environment [1]. To secure wireless communications, clients must establish safer session keys from the recorded less safe passwords before packet transmission begins, to avoid adversaries and maintain communication security. This is the act of password-authenticated key exchange, briefed as PAKE. PAKE has three basic structures: 2-party [2], 3-party [3] and C2C [4–7] (Figures 1–3). In 2-party PAKE, any two clients must use the previously recorded passwords to authenticate with each other before setting up communication keys. This may add a heavy burden to the system with numerous clients because each client must record lots of passwords. To improve it, 3-party PAKE lets two clients conduct mutual authentication via the server – who has all client passwords – to set up the communication keys. C2C PAKE advances one step further. It provides cross-realm communication, i.e., communication between clients of different servers. Specifically, it allows two clients in different realms to authenticate each other’s identity via the servers so as to build the communication keys. C2C-PAKE clearly outperforms 3-party PAKE since the latter lets close clients conduct authentication via the remote server.

To counter any possible malicious attacks on wireless networks, we need stronger security protection mechanisms. In a time when malicious attacks evolve ra-

*Corresponding author. E-mail: pjchuang@ee.tku.edu.tw

Figure 1. 2-party PAKE.
pidly, we especially need to advance security protection mechanisms so that they can protect network performance even when very critical information, such as passwords or server private keys, is snatched. To tackle the security problems, we first conduct a thorough analysis on possible adversary attacks and also on recent C2C PAKE protocols in this investigation. Based on the obtained analytical findings, we then develop a more advanced C2C PAKE protocol to upgrade the security of wireless communications. Involving the client passwords, smart card information and server private keys, the new C2C PAKE protocol functions by a number of operations, including the Mod calculation, a set of Diffie-Hellman operations, a parameter exchange approach and the asymmetric encryption. At reasonable cost, our protocol can significantly enhance communication security even when both client passwords and server private keys are grabbed by adversaries.

This paper is organized as follows. Section 2 gives possible adversary attacks, a formal problem statement and an analytical survey on major C2C PAKE protocols. The proposed new protocol and performance comparisons with related protocols are illustrated in Sections 3 and 4, respectively. Section 5 concludes the paper.

2. Background Study

To facilitate future discussion, we provide necessary background study, including possible adversary attacks, a formal problem statement as well as an analytical survey on major C2C PAKE protocols, in the following.

2.1 Possible Attacks

2.1.1 Dictionary Attacks (DA)

The attacker keeps guessing at the password and verifies the correct one by the return message. The following are different DA forms.

1. online DA (onDA): The attacker directly guesses the password via key authentication with the server and successfully interprets the return message when guessing right.

2. undetectable onDA (UDonDA): Similar to onDA but the server cannot detect the ongoing attack.

3. offline DA (offDA): The attacker collects the client authentication packet by eavesdropping and calculates the client’s password accordingly. (Note that both offDA and UDonDA can lead to the more serious impersonation attacks.)

2.1.2 Man-in-the-Middle Attacks

An attacker can join the communication between two parties by tampering the authentication information, without being detected.

2.1.3 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.

2.1.4 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 the information of other communications. A protocol able to resist such attacks is said to satisfy the forward security.

2.1.5 Denning-Sacco Attacks

When an (insider) attacker learns how to generate a communication key, he can use the key and a target client’s
information (such as ID) to produce the client’s communication key.

2.1.6 Replay Attacks
An attacker may intercept a client’s authentication packet and use it to re-authenticate with the server. If passing certification, the attacker can impersonate the client.

2.1.7 Denial-of-Service Attacks
An attacker paralyzes a server by sending it massive meaningless messages.

2.1.8 Impersonation Attacks
An attacker obtains a client’s password, using the fake identity to attack other clients or the server. Such an attack is hard to resist without advanced id checking mechanisms.

2.1.9 Password-Compromise Impersonation Attacks
An attacker obtains the password of a client (say A), uses it to masquerade as other clients and communicates with A – without being detected. This happens to clients of general protocols who communicate with each other based only on passwords. (Both password-compromise impersonation attacks and impersonation attacks are key-compromise impersonation (KCI) attacks).

2.2 The Problem Statement
Figure 3 depicts the basic operation of a cross-realm C2C PAKE protocol. To initiate the communication, a client in the left realm (say client A) first sends a communication request (Step 1) to its server who will send back a Ticket packet after verifying his identity (Step 2). Client A then passes the Ticket to his target client in the right realm (say client B) at this communication attempt (Step 3). Client B in turn sends his authentication information along with the Ticket to its server (Step 4) who will verify the identity and then send client A the negotiation information to help him negotiate a communication key with client B (Steps 5 and 6). The main goal of our investigation is to develop a desirable new cross-realm C2C-PAKE protocol for such a practice. To reach the goal, we need to consider both communication security and computational overhead. That is, our new C2C-PAKE protocol must improve the message exchange to fix previous security leaks on one hand and meanwhile keep computational cost under good control on the other hand.

2.3 Existing C2C PAKE Protocols
Some C2C PAKE protocols use smart cards to perform and support authentication (i.e., to verify identity) because they can modify data via physical contact or induction, store clients’ authentication information and perform encryption/decryption operations. The following is a brief introduction on major C2C PAKE protocols, given to facilitate our later discussions.

2.3.1 The Byun Protocol [4]
The Byun protocol, the first C2C PAKE protocol, has two major problems (Figure 4):
(1) As many of its packets contain password information, an adversary can grab Epwa (g^x), Epwa (g^y), Epwb (g^x') and Epwb (g^y') from communication (Figure 4(A)) and use the information to conduct offDA.
(2) An adversary can use Ticket B which contains g^{pw\cdot r} (Figure 4(B)) and subsequent negotiation information to conduct offDA.

2.3.2 The Feng Protocol [5,7]
To fortify the Byun protocol, the Feng protocol (Figure 5) lets packets carry less password information and adds asymmetric encryption to the Tickets and certification (involving both private and public keys, Figure 5 (A)). It uses the password only once for authentication between the server and the client. As the packet comes from the server to the client (Figure 5 (B)), an adversary can not guess the password by it, largely reducing the probability of offDA. This protocol nevertheless has its own problems:
(1) Using asymmetric encryption increases the cost of operations.
(2) To pass the packet containing the password from the server to the client, the client must first uplink a request to the server – taking two extra transmission runs.
(3) The two extra transmission runs, conducted to initiate negotiation only, are not certified and hence prone to the denial-of-service attacks.

2.3.3 The Jin Protocol [6,7]
This protocol (Figures 6–7) uses smart cards, not asymmetric encryption, to satisfy the KCI security. It exchanges the smart card information in the registration phase and uses it to set up a communication key in the login-and-authentication phase, yielding the following advantages.
(1) Effectively avoiding offDA because authentication involves no packets containing passwords (Figure 7(A) contains no PW).

(2) Effectively avoiding KCI attacks because authentication is not performed by passwords – an adversary holding only the password but no smart card information cannot forge the server-client authentication.

(3) Able to secure the password even when the server is under attack – because the server authenticates a
client by storing parameter $x$, not the password (Figure 7(B): the server uses only $x$ to calculate $R_A$).

The Jin protocol also faces problems.
1. Before it is put into work, each client must employ secure approaches to set up smart card information.
2. An adversary can still launch KCI attacks by using argument $x$, instead of client passwords (Figure 7(B): $x$ alone can be used to calculate $R_A$).

2.3.4 The Ding Protocol [8]

This protocol involves less cost to protect client passwords. As Figure 8 shows, the server broadcasts $g^a$ and $g^b$ for authentication. It can effectively avoid offDA because the server-client authentication uses the password only once (Figure 8(A)) and the packet contains a random number to keep an adversary from guessing at the password. The problems for this protocol will be (1) the server needs to broadcast $g^a$ and $g^b$ constantly to inform the clients, and (2) in the authentication phase from clients to the server, the server can not authenticate clients and is therefore vulnerable to denial-of-service attacks.

3. The New Protocol

Our new C2C PAKE protocol, an advanced smart card protocol based on the Jin protocol, has the following features.

3.1 Involving Mod Calculation

Figure 9 exhibits that our protocol can resist offDA due to the following two facts:
1. $R_A' = R_A'' = h_1(ID_A)x \mod p$ (note that the value of mod calculation remains equal after addition, multiplication and exponentiation).
$R_A' = (R_A - h1(PW_A))^a \mod p = ((h(ID_A)^x + h(PW_A)) \mod p) - h1(PW_A))^a \mod p$
$R_A'' = W_Ax \mod p = (h1(ID_A)^a \mod p)^x \mod p$
$R_A' = R_A'' = h1(ID_A)^a \mod p$

2. The transmission packets containing $R_A'$ and $W_A$ include random numbers as well and use no passwords after this phase.

3.2 Employing the Diffie-Hellman Operations

As Figure 9 shows, our protocol uses a set of Diffie-Hellman operations to enhance the transmission security because it will

1. keep an adversary from launching attacks without the password: to intercept the packet, he must first use the password to get the result of the Diffie-Hellman operations,
2. not increase the packets with passwords, thus reduc-
ing the risk of dictionary attacks. (In Figure 9(E), (A) adds a Diffie-Hellman operation and uses G to encrypt the packets. Please note that the original smart card authentication packet [6] is given in Figure 10 for clearer cross-reference.)

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

3.3 How to Exchange Parameters

We exchange parameters like the Ding protocol. As seen in Figure 9(D), the broadcast parameters in the Ding protocol are put in our initial smart cards, to save two extra transmissions in contrast to the Feng protocol (Figure 11).

3.4 Using Asymmetric Encryption

Our protocol adds asymmetric encryption to both the authentication packet (Figure 9(B)) and the Ticket (Figure 9(C)), to ensure that an adversary cannot crack the security without the private key. In the Ticket, PubSB makes sure only server B can decrypt the packet, while PriSA guarantees that the packet is sent from server A and that an adversary cannot decrypt the Ticket even if he has compromised the shared key between the servers. That is, our protocol can prevent an adversary with no private keys from launching attacks – at the cost of 4 asymmetric encryptions (or 2 pairs of encryption and decryption), and meanwhile makes it hard for attackers to disguise as servers – at the cost of another 4 asymmetric encryptions.

3.5 Key Advantages of Our Protocol

The above feature designs help establish a practically stronger C2C PAKE protocol, capable of enhancing wireless communication security on the following two grounds:

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

(2) An adversary must first obtain the private key and the shared key between servers to decrypt the Ticket (Figure 9(C)).

4. Performance Evaluation

Extensive performance evaluation is conducted to check and compare the performance of our new protocol and existing cross-realm C2C PAKE protocols, on two critical grounds: communication security and computational overhead. That is, we will evaluate and compare the performance of the protocols in terms of security gain and corresponding cost.

4.1 About Security Models

The performance of security protocols is subject to a number of variables (e.g., the length of the cipher text, encryption designs or password complexity) and it is usual that different evaluation mechanisms will yield different results. Thus, to prove its own value, each security protocol tends to take advantage from assuming an ideal calculation method or security model to simulate the attacker acts. To attain unbiased performance comparisons between different protocols, we need appropriate security models which can verify not only known attacks but also unknown ones. After a close survey, we decide to carry out our performance evaluation by the Yoneyama security model in [9] because it can verify attacks - such as KCI, LEP (Leakage of Ephemeral Private), UDonDA, and offDA - which the frequently used BR (Bellare-Rogaway) [10–13] and CK (Canetti-Krawczyk) [14,15] models are unable to verify.

To facilitate later discussions, a brief introduction on the attacker capability defined by the Yoneyama model is given below.

Execute: An attacker disguises as clients A and B to create a communication connection with the server, and gets information from the collected records to launch a DA.

SendClient: An attacker sends a forged packet to a client who will yield the calculated results following the requirements in the packet.

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

(Execute, SendClient and SendServer together can verify the BR security, i.e., verify if an attacker is capable of striking a general attack.)

StaticKeyReveal: An attacker can obtain the target static key information, such as the password between a client and the server - to reach a KCI attack or the server’s private key - to strike an LEP attack.

SessionKeyReveal: An attacker can obtain a client’s session key after the session completes.

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, taking full control over the client. A client not attacked by this oracle is called an honest client.

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

Test: To test if an attacker can get a client’s session key by guessing – the model randomly selects 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

The Yoneyama security model is used to evaluate our C2C PAKE protocol, under the assumption that if an attacker obtains the private key, he will not obtain K, pw or x; if unable to get the private key, he will get x, pw and K. The following are some experiments (Exp), in which the word probability is shortened as prob. for brevity).

Exp 0: Let Succ0 be the case that an attacker has correctly guessed the authentication bit.

$$\text{Adv}_{\text{pake},\text{ID}}(A) = 2\text{Pr}[\text{Succ0}] - 1$$

Exp 1: Authenticate h1, h2, h3 and the ideal encryption and decryption e and D. The prob. that the attacker uses Send, Reveal and Execute to find the random numbers is $$\frac{q_E^2 + q_h^2 + q_n^2 + q_s^2}{2(q - 1)}$$.

$$\text{Pr}[\text{Succ0}] - \text{Pr}[\text{Succ1}] \leq \frac{q_E^2 + q_h^2 + q_n^2 + q_s^2}{2(q - 1)}$$

Exp 2: Replace W by a random number. The prob.
that an attacker can distinguish $W_A$ from the random number equals the prob. that he can verify $W_A$ by obtaining $x$, which will be $Adv_{ddh}^{dl}(T_{ddh})$. Then, there are two cases.

(Case 1) The attacker obtains the private key: The prob. of successfully verifying $W_A = \text{the prob. of using the password to get } g^x$ and crack DDH $= q_{Static\ Key} \cdot Adv_{ddh}^{dl}(T_{ddh}) \cdot Adv_{ddh}^{dl}(T_{ddh})$.

(Case 2) The attacker obtains the password: The prob. of successfully verifying $W_A = \text{the prob. of cracking DDH and asymmetric encryption} = q_{Static\ Key} \cdot Adv_{ddh}^{dl}(T_{ddh}) \cdot Adv_{cc}^{eua}(T_{cc2})$.

By Formulas (1) and (2), we have

$$|Pr[Succ1] - Pr[Succ2]| \leq q_{Static\ Key} \cdot Adv_{ddh}^{dl}(T_{ddh}) \cdot (Adv_{ddh}^{dl}(T_{ddh}) + Adv_{cc}^{eua}(T_{cc2}))$$

Note that Exp 2 is conducted mainly to verify the prob. that an attacker can crack the authentication packet $\{(g^2)_{pw}, (ID_A, ID_B, T_1, C_A, W_A)_{cr}, Pub_A\}$. In Case 1, the attacker gets the private key by $q_{Static\ Key}$, breaks symmetric encryption by $Adv_{ddh}^{dl}(T_{ddh})$, and cracks DDH by $Adv_{ddh}^{dl}(T_{ddh})$. In Case 2, he gets the password by $q_{Static\ Key}$ and cracks asymmetric encryption by $Adv_{cc}^{eua}(T_{cc2})$.

**Exp 3:** We use a random value to replace $k$ in the Ticket and get 2 cases.

(Case 1) The attacker gets the private key: the prob. of successfully verifying $k = \text{the prob. of breaking the encryption and decryption} q_{Static\ Key} \cdot Adv_{cc}^{eua}(T_{cc2}, q_e, q_d)$.

(Case 2) The attacker gets $K$: the prob. of successfully verifying $k = \text{the prob. of breaking the asymmetric encryption} q_{Static\ Key} \cdot Adv_{cc}^{eua}(T_{cc2})$.

We use the Ticket 3 times and get

$$|Pr[Succ2] - Pr[Succ3]| \leq 3q_{Static\ Key} \cdot Adv_{cc}^{eua}(T_{cc2}, q_e, q_d) + Adv_{cc}^{eua}(T_{cc2})$$

Note that Exp 3 verifies the prob. that an attacker can crack the Ticket. In Case 1, the attacker gets the private key by $q_{Static\ Key}$ and cracks the symmetric encryption of $K$ by $Adv_{cc}^{eua}(T_{cc2}, q_e, q_d)$; in Case 2, he gets $K$ by $q_{Static\ Key}$ and cracks on asymmetric encryption $Adv_{cc}^{eua}(T_{cc2})$.

**Exp 4:** To verify the prob. that an attacker can counterfeit MAC, which will be $Adv_{MAC}^{ma}(T_{mac}, q_m, q_e)$. As MAC is used 2 times (Et and Eb), we have

$$|Pr[Succ3] - Pr[Succ4]| \leq 2Adv_{MAC}^{ma}(T_{mac}, q_m, q_e)$$

**Exp 5:** Replace DDH by $U = g^x$, $V = g^z$ and $Z = g^y$.

The prob. that an attacker can distinguish DDH from $U$, $V$ and $Z$ the prob. that he can crack DDH $= Adv_{ddh}^{dl}(T_{ddh})$. Thus

$$|Pr[Succ4] - Pr[Succ5]| \leq Adv_{ddh}^{dl}(T_{ddh})$$

The prob. that Exp 5 succeeds equals the prob. that the attacker guesses $sk$ by none of the above but other ways (including the prob. of using Corrupt ($C_1, 2$) and onDA $q_{send}(2, D)$). Thus,

$$Pr[Succ5] \leq (q_{sendClient} + q_{sendServer})/2|D| + 1/2$$

From (1)–(7), we attain

$$Adv_{ddh}^{dl}(T, R) \leq (q_e^2 + q_h^2 + q_k^2 + q_{\lambda}^2)(q - 1) + q_{Static\ Key} \cdot Adv_{ddh}^{dl}(T_{ddh}) \cdot Adv_{ddh}^{dl}(T_{ddh}) + Adv_{cc}^{eua}(T_{cc2}) + 3q_{Static\ Key} \cdot (Adv_{cc}^{eua}(T_{cc2}, q_e, q_d) + Adv_{cc}^{eua}(T_{cc2})) + 2Adv_{MAC}^{ma}(T_{mac}, q_m, q_e) + Adv_{ddh}^{dl}(T_{ddh}) + (q_{sendClient} + q_{sendServer})/2|D| + 1/2$$

In ideal conditions, it is unlikely that an attacker can break down all encryption mechanisms. This is because both the dictionary/password length and the number of oracles the attacker uses are as large as infinite. When $q$ and $D$ are close to infinite, and $Adv_{ddh}^{dl}(T_{ddh}), Adv_{ddh}^{dl}(T_{ddh}), Adv_{cc}^{eua}(T_{cc2}), Adv_{cc}^{eua}(T_{cc2}, q_e, q_d)$ and $Adv_{MAC}^{ma}(T_{mac}, q_m, q_e)$ are approximately zero, $Adv_{ddh}^{dl}(A) = 2 \cdot 1/2 - 1$ – a near zero value – indicating an attacker has almost no chance to break this protocol.

According to this security model, an attacker can eavesdrop, send packets and obtain sk to launch a general attack or break the BR security, or he can obtain a static key (containing 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 the BR + KCI + LEP security, not just the general BR security.

### 4.3 The Security of Other Protocols

Brief security analysis on other protocols is provided here to assist subsequent comparisons. **AidkeyC2C** can attain the BR but not KCI security because possible password leakage may cause attacks (in Figure 12, an attacker can counterfeit X” by the password). **The Ding protocol** also attains the BR but not KCI security because an attacker can use the obtained password to decrypt $E_X$ and $E_R$, as Figure 13 shows. **The Jin protocol** can reach the KCI security if its smart card is secure. If not secure (e.g., when the password leaks), it is vulner-
able to KCI attacks too. As for the Feng protocol, a secure private key will ensure the KCI security, but a fetched private key will lead to possible LEP attacks (in Figure 14, the Ticket is encrypted by both the private and public keys).

4.4 Security and Cost Comparisons

Table 1 lists the security and cost of AidkeyC2C, Ding, Jin, Feng and our new protocol (OURs). When considering cost, we take both calculation times and complexity into account. The cipher text length and cost for symmetric/asymmetric encryptions are respectively 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 – cost will increase when blocks increase (in linear relationship). MAC resembles asymmetric one-way Hash and is hence taken as symmetric calculation. In short, 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 (note that the results in Table 1 are average numbers). In the encryption process, each encryption or decryption is counted as one calculation. The cost of the smart card is negligible as it is established only once.

Based on Table 1, Figure 15 further illustrates the cost comparison. As it shows, the Feng protocol takes more transmission runs than other protocols, resulting from the need to change the DDH parameters. Ding takes less cost to reach security (but not the KCI or LEP security) because it involves no asymmetric calculation which costs much more than symmetric calculation and is the focus of our discussion here. We see from both Table 1 and Figure 15 the increase in cost and security is in linear relationship for all protocols – except AidkeyC2C, indicating the asymmetric calculations in AidkeyC2C are unreasonable. The linear relationship between the cost increase and security gain in our protocol tells a different story: it strongly justifies our choice of pursuing valuable security gain at the cost of reasonable extra cost.

Table 2 gives complexity comparisons for frequently-used calculations, including the asymmetric algorithm – RSA, symmetric algorithm – AES (advanced encryption standard), Hash – SHA1, MAC – MD5, Mod and the Diffie-Hellman algorithm (note that the listed complexity is calculated and obtained according to [16]). Table 3 lists complexity comparisons among protocols. In both tables,

\[ n = \text{password length} \]

![Figure 12. AidkeyC2C: the weakness.](image)

![Figure 13. Ding: the weakness.](image)

![Figure 14. Feng: the weakness.](image)

**Table 1. Calculation times and security comparisons**

<table>
<thead>
<tr>
<th></th>
<th>Transmission times</th>
<th>Symmetric calculation</th>
<th>Asymmetric calculation</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>AidkeyC2C</td>
<td>6</td>
<td>18</td>
<td>8</td>
<td>DA + BR + FS</td>
</tr>
<tr>
<td>Ding</td>
<td>6</td>
<td>36</td>
<td>0</td>
<td>DA + BR + FS</td>
</tr>
<tr>
<td>Jin</td>
<td>6</td>
<td>30</td>
<td>4</td>
<td>DA + BR + FS + KCI (Incomplete)</td>
</tr>
<tr>
<td>Feng</td>
<td>9</td>
<td>34</td>
<td>8</td>
<td>DA + BR + FS + KCI</td>
</tr>
<tr>
<td>OURs</td>
<td>6</td>
<td>44</td>
<td>12</td>
<td>DA + BR + FS + KCI + LEP</td>
</tr>
</tbody>
</table>
d and \( p = \) exponent parameter length for exponent calculation (set as 16)

\( e = \) a random number greater than or equal to \( 10^5 \) (set as \( 10^5 \) here)

Figure 16, which gives complexity versus password lengths for the operations, shows that the complexity of asymmetric encryption and decryption rises with the password length. Figure 17 exhibits large cost difference between protocols with and without asymmetric encryption, and the difference is apparently larger than that between protocols with asymmetric encryption. Among protocols with asymmetric encryption, Jin takes 4 Mod operations, Feng and AidkeyC2C take 8 asymmetric encryptions, and ours takes 8 asymmetric encryptions plus 4 Mod operations. Mod is in fact a form of asymmetric calculation with less complexity. It explains why Feng, AidkeyC2C and our protocol generate not much cost difference, and also justifies our choice of paying reasonable extra cost for desirable security gain – such as the LEP security.

### 4.5 Practical Applications

#### 4.5.1 Handheld Devices or Diskless Workstations

C2C PAKE can be widely used in practical applications, especially in handheld devices or diskless workstations because C2C PAKE does not ask a client to connect continually with all servers. Instead, it sets up a con-
nection only when necessary to prevent possible attacks and achieve better communication security. When new clients join the network, the original clients need no extra information because the protocol will handle security authentication. Also, clients can directly communicate with the server, to handily exchange or update the smart card data.

4.5.2 Kerberos Workstations

Having similar structures as Kerberos, C2C PAKE can be used to upgrade the 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 remains the same for a long period of time, it is desirable to build 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 in a Kerberos workstation secure better communications.

4.5.3 Commercial or Medical Workstations

With its high security strength, our protocol can be put to work in workstations requiring intensive security, such as those involving commercial transactions, medical practices or military purposes. (For instance, when clients are engaged in banking or medical activities, they can meanwhile create or update the smart card to ensure security passage.)

4.6 Other Discussions

Recall that the main goal of our investigation is to develop a stronger new C2C PAKE protocol, and we have achieved the goal by coming up with a new protocol which can advance the security of wireless communications, at very reasonable cost, even when both client passwords and server private keys are snatched. This investigation indeed handles a very popular research topic which has brought up a number of recent results in the literature. For instance, to enhance the security of PAKE, we can see the solutions for two-server PAKE in which the client can establish different cryptographic keys respectively with two servers [22]. The advantage lies in that when one server is compromised, an attacker cannot pretend to be a client with the information from the compromised server. In fact, existing C2C PAKE protocols (including ours) usually store the passwords needed for client authentication in a single server in each realm, and when a server is compromised (due to hacking or insider attacks), the stored passwords will all be disclosed. We believe that if the two-server concept is brought into our new protocol, we can furthermore enhance the security of PAKE.

5. Conclusions

We introduce a new and advanced cross-realm C2C PAKE protocol in this paper to counter the massively increasing malicious attacks in the ever-growing wireless communications, with the main purpose of building safer wireless environments. Involving the client passwords, smart card information and server private keys, the new PAKE protocol establishes a strong security protection mechanism by a number of operations, including the Mod calculation, a set of Diffie-Hellman operations, a parameter exchange approach and the asymmetric encryption. It is desirable in being able to sustain communication security even when both client passwords and server private keys are snatched by adversaries. Performance evaluation on the security and cost of various PAKE protocols show that, in contrast to other protocols, our new protocol attains very significant security gain due to its ability to defend more complex attacks, such as the BR, KCI and even LEP attacks, at reasonable extra cost.

References


Manuscript Received: Aug. 28, 2014
Accepted: Oct. 22, 2015