A Secure and Efficient
Key Exchange Protocol for Mobile Communications

Fuw-Yi Yang and Jinn-Ke Jan*

Department of Applied Mathematics, National Chung Hsing University,
Taichung 402, Taiwan, R.O.C., Email: yangfy@ms7.hinet.net

*Department of Computer Science, National Chung Hsing University,
Taichung 402, Taiwan, R.O.C., Email: jkjan@cs.nchu.edu.tw

Abstract

This paper proposes a key exchange protocol with mutual authentication, which requires only 0.1 modular multiplications for online computations. This online computation is ten times faster than that of conventional protocols. The message size of the proposed protocol is about half (50%–66%) that of the previous protocols. In addition to its efficiency in online computation and bandwidth, the paper provides a formal proof to guarantee the security of the proposed protocol. Possessing of both secure and efficient properties makes the proposed protocol suitable for the low power mobile communications.

1. Introduction

The L-MAKEP (linear mutual authentication key exchange protocol) proposed in [1] is a key exchange protocol specially designed for the wireless communications between a low power mobile device (client) and a powerful base station (server). The goal of the L-MAKEP is to make the computational complexity as efficient as possible at the client end without decreasing the strength of security. Consequently, the server sides do most of the computations. It is a good idea in the environment of the wireless communications, especially in the case of a lower power client end.

However, there are flaws in L-MAKEP. The scheme in [2] successfully mounted an unknown key-share attack [3-6] on this protocol and proposes an improved protocol.
A scenario of unknown key-share attack proposed in [3-4] is as follows. Assume that an account holder $U$ (user $U$) initiates a session communication with a bank server $S$, and an adversary $E$ mounts an unknown key share attack on this session. As a result of the attack, user $U$ believes that a session key is shared with the server $S$ and server $S$ believes that the session key is shared with the adversary $E$. Although adversary $E$ does not obtain the session key, this attack could cause problem. For example, if user $U$ initiates the protocol to deposit an electronic fund in his account, the deposit will eventually be made to the adversary $E$’s account. The protocol L-MAKEP and its improvement in [2] are also insecure under the attack of middle-man as shown by the scheme in [7]. The work in [7] also proposes an improved protocol to resist the attacks of middle-man and unknown key-share; we call the protocol I-MAKEP.

This paper proposes an efficient and secure key exchange protocol for mobile communications. Although both participants are authenticated through the public key cryptography, the online computational cost for client side is only 0.1 modular multiplications. This computational complexity is ten times faster than that of the protocols L-MAKEP and I-MAKEP. The message size for the proposed protocol is 3712 bits while the message size for the protocol L-MAKEP is 7328 bits and for the protocol I-MAKEP is 5600 bits (Section 3 will show the details.). Moreover, the proposed protocol consists of only three messages while L-MAKEP and I-MAKEP consist of four messages (two round trips). The savings in computations, bandwidth, and messages is valuable for key exchange protocols, especially in the environment of wireless communications with low power mobile devices. In addition, the paper provides a formal proof to show that breaks the security of the proposed protocol is to solve the difficulty of the factoring problem. Thus the proposed protocol not only resists to the attacks mentioned above but also resists to other potential attacks.

The organization of this paper is as follows. Section 2 presents the new protocol and describes the notation used in the paper. Section 3 compares the computational cost and message size between the proposed protocol, L-MAPEP, and I-MAKEP. In Section 4, the security of the new protocol is investigated and proven. Finally, Section 5 concludes the paper.
2. Protocol proposed

This section presents an efficient and secure key exchange protocol with mutual authentication (ES-MAKEP). The protocol is inspired by the concept of chameleon hash function in [8]. We first describe the cryptographic settings for the protocol.

Let $\varepsilon_{PK}()$ denote an asymmetric encryption function and $\delta_{SK}()$ be the corresponding decryption function. Similarly, $E_K()$ represents a symmetric encryption function and $D_K()$ is the decryption function so that $m = D_K(E_K(m))$. Both the encryption functions $\varepsilon_{PK}(.)$ and $E_K(.)$ are assumed to be secure in the model of adaptive chosen ciphertext attack [9-11]. In this attack model, an adversary is given a decryption oracle to decrypt the ciphertexts adaptively chosen by the adversary except the one that he is trying to decrypt. The identification of a client entity $U$ (user $U$) is denoted by $ID_U$ and the identification of a server $S$ is represented by $ID_S$. Server $S$ has a private key $SK_S$ and the corresponding public key $PK_S$. The public key and secret key of user $U$ are selected in the following way. User $U$ randomly chooses four large prime numbers $p, q, p', q'$ such that $p = 2p' + 1$ and $q = 2q' + 1$. Then, he or she randomly selects an element $g$ of order $\lambda(n)$ from the multiplicative group $\mathbb{Z}_n^*$, where $n = pq$ and $\lambda(n) = \text{lcm}(p - 1, q - 1)$. Thus user $U$ has the private key $(p, q)$ and the public key $(g, n)$. In addition, $x \| y$ denotes that string $x$ concatenates string $y$, $|n|$ represents bit length of $n$, $r \in_R G$ denotes that $r$ is a random number selected from the set $G$, and $l$ indicates the length of session keys, e.g. 160 bits for a typical value suggested in [12].

The following steps describe the details of the protocol ES-MAKEP.

1. User $U$ randomly selects three numbers $r_{UK}, r_{UR}$ and $r_{UF}$, i.e. $r_{UK}, r_{UF} \in_R \{0, 1\}^l$ and $r_{UR} \in_R \mathbb{Z}_{\lambda(n)}$. Compute the quantities $C_{r_{UK}} = \varepsilon_{PK_S}(r_{UK})$ and $CMT = g^{r_{UF} \| r_{UR}} \mod n$, i.e. encrypt the random number $r_{UK}$ and calculate the commitment of $r_{UF}$ and $r_{UR}$. Then, user $U$ sends server $S$ message $M1 = \{ C_{r_{UK}}, CMT, ID_U \}$ to ask for initiating a new session.

2. On receiving the message $M1$, server $S$ decrypts the ciphertext $C_{r_{UK}}$ to obtain $r_{UK}$. $S$ also selects a random numbers $r_{SK} \in_R \{0, 1\}^l$, calculates the session key $\sigma_{SU} = r_{SK}$.
and encrypts the random number \( r_{UK} \), i.e. \( C2_{r_{UK}} = E_{\sigma_{SU}}(r_{UK}) \). Then, server \( S \) sends \( M2 = \{ r_{SK}, C2_{r_{UK}} \} \) to challenge user \( U \).

3. Upon receiving the challenge message \( M2 \), \( U \) calculates the session key \( \sigma_{US} = r_{UK} \) and decrypts the ciphertext \( C2_{r_{UK}} \) to obtain a random number \( r'_{UK} \), i.e. \( r'_{UK} = D_{\sigma_{US}}(C2_{r_{UK}}) = D_{\sigma_{US}}(E_{\sigma_{SU}}(r_{UK})) \). Note that \( \sigma_{SU} \) is equal to \( \sigma_{US} \) if messages \( M1 \) and \( M2 \) are successfully transmitted. User \( U \) authenticates server \( S \) by checking \( r_{UK} = r'_{UK} \). After authenticating server \( S \), \( U \) constructs a response message as follows.

3.1. Compute the quantities \( S_F = h(r_{UK}, r_{SK}, ID_U, ID_S) \) and \( C3 = E_{\sigma_{SU}}(ID_U) \).

3.2. Solve for \( S_R \) in equation (1).

\[
2^{[n]} r_{UF} + r_{UR} = 2^{[n]} S_F + S_R \mod \lambda(n) \\
S_R = 2^{[n]} (r_{UF} - S_F) + r_{UR} \mod \lambda(n) \tag{1}
\]

\( U \) sends the response message \( M3 = \{ C3, S_R \} \) to server \( S \).

4. \( S \) computes the quantities \( S_F = h(r_{UK}, r_{SK}, ID_U, ID_S) \) and \( CMT' = g^{S_F \parallel S_R} \mod n \). \( S \) authenticates user \( U \) by verifying \( CMT = CMT' \).

For easy reading, the message flows are listed below.

- \( U \to S \): \( M1 = \{ C1_{r_{UK}}, CMT, ID_U \} \), where \( C1_{r_{UK}} = \varepsilon_{PK_S}(r_{UK}) \) and \( CMT = g^{r_{UF} \parallel r_{UR}} \).
- \( S \to U \): \( M2 = \{ r_{SK}, C2_{r_{UK}} \} \), where \( C2_{r_{UK}} = E_{\sigma_{SU}}(r_{UK}) \) and \( \sigma_{SU} = r_{UK} \oplus r_{SK} \).
- \( U \to S \): \( M3 = \{ C3, S_R \} \), where \( C3 = E_{\sigma_{SU}}(ID_U) \), \( 2^{[n]} r_{UF} + r_{UR} = 2^{[n]} S_F + S_R \) and \( S_F = h(r_{UK}, r_{SK}, ID_U, ID_S) \).

### 3. Performance

For each step in the protocols ES-MAKEP, L-MAKEP, and I-MAKEP, the computations and message sizes are displayed in Table 1, 2 and 3. The cost of computations is further divided into two parts: online and offline computations. To simplify the estimation of message sizes and computations, we have assumed that the asymmetric encryption is a 1024-bit RSA encryption with a low exponent public key \( e = 3 \), the symmetric encryption outputs a stream of 160-bit and \( ID_U \) is a 160-bit
encoding. Also, the cost of additions, hash operations, and symmetric encryption and decryption are not included. For a practical cryptographic settings [12], \(|n|=1024\) bits and \(|r_U| = |S_r| = 160\) bits, the computational cost (software implementation) for solving \(S_r\) is estimated to be only about 0.1 modular multiplications of two 1024-bit numbers modulo a 1024-bit modulus [8]. Note that computing \(S_r\) involves only a reduction of a 1184-bit number to a 1024-bit number while computing a modular multiplication of two 1024-bit numbers requires a conventional multiplication and a reduction of a 2048-bit number to a 1024-bit number. For easy comparison, the message flows of protocols L-MAKEP and I-MAKEP are listed below.

**The message flows of L-MAKEP:** The server \(S\) has published a public key \(PK_S\), large prime \(p\) and a primitive element \(g \in Z_p^*\).

\[U \rightarrow S: M1 = \{ID_U, g^{a_{2i-1}}, g^{a_{2i}}, \text{Sig}_{TA}(ID_U, g^{a_{2i-1}}, g^{a_{2i}})\}, \] where \(a_{2i-1}, a_{2i} \in R Z_{p-1}\), and \(\text{Sig}_{TA}(m)\) is a signature of a trusted authority (TA) on the message \(m\). Server \(S\) should verify the signature \(\text{Sig}_{TA}(ID_U, g^{a_{2i-1}}, g^{a_{2i}})\).

\[S \rightarrow U: M2 = \{r_S\}, \text{where } r_S \in R Z_{p-1}.\]

\[U \rightarrow S: M3 = \{x, y\}, \text{where } x = e_{PK_S}(r_U), y = (a_{2i-1}(x \oplus r_S) + a_{2i}) \mod (p - 1), \text{ and } r_U \in R \{0, 1\}^{160}. \text{User } U \text{ computes the response message } M3 \text{ and server } S \text{ verifies it by checking } g^y = g^{a_{2i-1}(x \oplus r_S)} g^{a_{2i}} \mod p.\]

\[S \rightarrow U: M4 = \{E_d(x)\}, \text{where } \sigma = r_U \oplus y.\]

**The message flows of I-MAKEP:** The server \(S\) has private key \((d, p, q)\) and public key \((e, g, N)\), where \(p, q\) are two large primes, \(N = p \times q\), \(g\) has maximum order in the group \(Z_N^*\), \(\phi(N) = (p - 1)(q - 1), \gcd(e, \phi(N)) = 1, \text{ and } e \times d = 1 \mod \phi(N)\). The user \(U\) uses the pair \((ID_U, v)\) to register at server \(S\) and receives a certificate \(y = (v - ID_U)^d \mod N\), where \(v = g^x \mod N\) and \(x \in R Z_N\) is the secret key of \(U\).

\[U \rightarrow S: M1 = \{ID_U, y\}. \text{Server } S \text{ verifies } M1 \text{ by checking } v = y^e + ID_U \mod N.\]

\[S \rightarrow U: M2 = \{r_S\}, \text{where } r_S \in R Z_N.\]

\[U \rightarrow S: M3 = \{u, t, s\}, \text{where } u = g^w \mod N, t = e_{PK_S}(k), s = w + x h(r_S, t, u), \text{ and } w \text{ and } k \in R Z_N. \text{User } U \text{ computes the response message } M3 \text{ and server } S \text{ verifies it} \]
by checking $u = v^{b(r_S,t,u)} g^s \mod N$.

$S \rightarrow U: M4 = \{h(k)\}$.

As shown in Table 1, the client’s online computations for ES-MAKEP, L-MAKEP, and I-MAKEP are 0.1, 1, and 1 MMs, respectively. Although L-MAKEP requires only two MMs for offline computation, this protocol requires a trusted authority to issue several commitments, which are the sources of the middle-man attack described in [7]. Table 2 and Table 3 show that among the three protocols, ES-MAKEP has the most efficient in computations and the least requirement in bandwidth and messages.

Table 1: Client’s computations in protocols ES-MAKEP, L-MAKEP, I-MAKEP

<table>
<thead>
<tr>
<th></th>
<th>ES-MAKEP</th>
<th>L-MAKEP</th>
<th>I-MAKEP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Online</td>
<td>Offline</td>
<td>Online</td>
</tr>
<tr>
<td>Message $M1$</td>
<td>0</td>
<td>1778 MMs$^a$</td>
<td>0</td>
</tr>
<tr>
<td>Message $M2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Message $M3$</td>
<td>0.1 MMs</td>
<td>0</td>
<td>1 MM</td>
</tr>
<tr>
<td>Message $M4$</td>
<td>$\text{None}$</td>
<td>$\text{None}$</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>0.1 MMs</td>
<td>1778 MMs</td>
<td>1 MM</td>
</tr>
</tbody>
</table>

a. Computing $C_1^r_{UK} = e^{PK_S}_S (r_{UK})$ requires 2 MMs and computing $CMT = g^{r_{UR}} r_{UR}$ requires $1.5 \times (160 + 1024) = 1776$ MMs, where MM denotes modular multiplications of two 1024-bit numbers modulo 1024-bit modulus.

b. Encryption requires 2 MMs (assume RSA encryption and public key $PK_S = 3$).

c. Computations of commitment requires $1.5 \times 1024 = 1536$ MMs and encryption requires 2 MMs (public key $e = 3$).
Table 2: Server’s computations in protocols ES-MAKEP, L-MAKEP, I-MAKEP

<table>
<thead>
<tr>
<th></th>
<th>ES-MAKEP</th>
<th>L-MAKEP</th>
<th>I-MAKEP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Online</td>
<td>Offline</td>
<td>Online</td>
</tr>
<tr>
<td>Message M1</td>
<td>1536 MMs</td>
<td>0</td>
<td>2 MMs$^d$</td>
</tr>
<tr>
<td>Message M2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Message M3</td>
<td>1776 MMs$^f$</td>
<td>0</td>
<td>3456 MMs$^g$</td>
</tr>
<tr>
<td>Message M4</td>
<td>None</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>3312 MMs</td>
<td>0</td>
<td>3458 MMs</td>
</tr>
</tbody>
</table>

- d. Verification requires 2 MMs (assume TA uses RSA cryptosystem and verification key $e = 3$).
- e. Computational cost of encryption requires 2 MMs (public key $e = 3$).
- f. Computing $CMT' = g^{S_E} || S_R$ requires $1.5 \times (160 + 1024) = 1776$ MMs.
- g. Decryption requires $1.5 \times 1024 = 1536$ MMs and verification requires $1.25 \times 1.5 \times 1024 = 1920$ MMs, by the technique of simultaneous multiple exponentiations [13].
- h. Decryption requires $1.5 \times 1024 = 1536$ MMs and verification requires $1.17 \times 1.5 \times 1024 = 1797$ MMs.

Table 3: The message sizes of protocol ES-MAKEP, L-MAKEP, I-MAKEP

<table>
<thead>
<tr>
<th></th>
<th>ES-MAKEP</th>
<th>L-MAKEP</th>
<th>I-MAKEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message M1</td>
<td>2208 bits</td>
<td>3232 bits</td>
<td>1184 bits</td>
</tr>
<tr>
<td>Message M2</td>
<td>320 bits</td>
<td>1024 bits</td>
<td>1024 bits</td>
</tr>
<tr>
<td>Message M3</td>
<td>1184 bits</td>
<td>2048 bits</td>
<td>3232 bits</td>
</tr>
<tr>
<td>Message M4</td>
<td>none</td>
<td>1024 bits</td>
<td>160 bits</td>
</tr>
<tr>
<td>Total</td>
<td>3712 bits</td>
<td>7328 bits</td>
<td>5600 bits</td>
</tr>
</tbody>
</table>
4. Security analysis

In the followings, subsection 4.1 proves that the proposed protocol ES-MAKEP is a mutual authentication protocol (Theorem 1 and 2) and has resistance to those attacks proposed in [2, 7], subsection 4.2 proves that the protocol is a secure protocol (Lemma 3, 4, and 5).

4.1 ES-MAKEP is a mutual authentication protocol

Theorem 1. Protocol ES-MAKEP correctly authenticates user U.

Proof. If user U knows the secret key \((p, q)\), then U can compute the quantity \(\lambda(n) = \text{lcm}(p - 1, q - 1)\), solve for \(S_R\) in (1) and construct the response message \(M_3 = \{C_3, S_R\}\). Since \(2^{[n]} r_{UF} + r_{UR} = 2^{[n]} S_F + S_R \mod \lambda(n)\), the equation \(g^{r_{UF} \| r_{UR}} = g^{S_F \| S_R \mod n}\) is obtained. Thus server S successfully authenticates user U.

Assume that an adversary E does not now the private key \((p, q)\) of user U and successfully impersonates the user U, then E has generated the quantities \(S_F\) and \(S_R\) such that \(g^{r_{UF} \| r_{UR}} = g^{S_F \| S_R \mod n}\). Thus equation (2) is obtained.

\[
g^{r_{UF} \| r_{UR}} - S_F \| S_R = g^w = 1 \mod n
\]  

Equation (2) implies that \(w\) is a multiple of \(\lambda(n)\). Since \(\lambda(n) = \text{lcm}(p - 1, q - 1)\) and \(\phi(n) = (p - 1)(q - 1)\), then \(\phi(n)\) divides \(2w\). The algorithm in [14] shows that the factorization of \(n\) can be computed efficiently, if any multiple of \(\phi(n)\) is known. The public key \(n\) thus can be factored by the adversary E using the algorithm in [14]. This conclusion contradicts the intractable assumption of factoring problem. Therefore, if server S successfully authenticates user U, then U knows the private key \((p, q)\). \[\square\]

Theorem 2. Protocol ES-MAKEP correctly authenticates server S.

Proof. If server S knows the secret key \(SK_S\), then S can decrypt the ciphertext \(C_{1_{UK}}\) to obtain \(r_{UK}\), calculate the session key \(\sigma_{SU} = r_{UK} \oplus r_{SK}\), and generate the ciphertext \(C_{2_{UK}} = E_{\sigma_{SU}}(r_{UK})\). On receiving \(r_{SK}\), U calculates the session key \(\sigma_{US} = r_{SK} \oplus r_{UK}\), using the stored \(r_{UK}\). Thus, the session keys \(\sigma_{SU}\) and \(\sigma_{US}\) have the same quantity.
Clearly, the quantity recovered by decrypting the ciphertext $C_{2_{UK}}$ will be equal to the quantity of stored $r_{UK}$.

With overwhelming probability, $S$ knows the secret key $SK_S$, if user $U$ authenticates server $S$ as legal. Namely, only $S$ can decrypt the ciphertext $C_{1_{UK}}$ to obtain the random number $r_{UK}$. This result is derived from the security of the encryption functions $e_{UK}()$ and $E_{K}()$, which is assumed to be secure against the adaptive chosen ciphertext attack. Therefore, server $S$ is successfully authenticated by $U$ if and only if $S$ knows the secret key $SK_S$. □

The attacks described in [2, 7] are performed by modifying messages intercepted from previous sessions. Thus by Theorem 1 and 2, the protocol ES-MAKEP is resistant to them.

4.2 ES-MAKEP is a secure protocol

This subsection investigates the security of the protocol ES-MAKEP. We adopt the security measure and those attack model used in [15-16]. Assume that an adversary with total control over the communication channels can mount parallel attacks, and is told the previous session keys. A key exchange protocol is secure if the following requirements are satisfied.

1. If both participants honestly execute the protocol, then the session key is $\sigma = \sigma_{US} = \sigma_{SU}$.
2. No one can calculate the session key $\sigma$ except participants $U$ and $S$.
3. The session key is indistinguishable from a truly random number.


Proof. After mutually authenticating, both participants have agreed on the random numbers $r_{UK}$ and $r_{SK}$ by Theorem 1 and Theorem 2. Therefore, $\sigma = \sigma_{US} = r_{UK} \oplus r_{SK} = \sigma_{SU}$. □

Proof. User $U$ generates the random number $r_{UK}$ and constructs the ciphertext $C_{1r_{UK}} = e_{PK}(r_{UK})$. The encryption functions $e_{PK}()$ and $E_K()$ are secure. Thus, only $U$ and $S$ know the quantity of $r_{UK}$. Therefore, only participants $U$ and $S$ can calculate the session key $\sigma = \sigma_{US} = \sigma_{SU} = r_{UK} \oplus r_{SK}$.


Proof. The proof is straightforward, since both the numbers $r_{UK}$ and $r_{SK}$ are randomly selected from the set $\{0, 1\}^l$.

5. Conclusions

The paper has proposed a secure and efficient key exchange protocol. The proposed protocol outperforms the previous protocols L-MAKEP and I-MAKEP in the message round, message size, server’s computations and client’s online computation. These advantages make it most suitable for the environment of low power mobile communications. However, all the protocols ES-MAKEP, L-MAKEP, and I-MAKEP do not have the property of perfect forward secrecy. This strength of security requires that even the long term secret key is compromised, the session keys should remain to be safe. As can be seen from the first step in Section 2, the random number $r_{UK}$ and session key $\sigma_{SU}$ is computed if server’s secret key is revealed. This suggests further research in providing a key exchange protocol with the perfect forward secrecy and all advantages of ES-MAKEP.

References


