Authenticated Key Agreement Without Using One-way Hash Functions Based on The Elliptic Curve Discrete Logarithm Problem

Li-Chin Huang and Min-Shiang Hwang

1 Department of Computer Science and Engineering, National Chung Hsing University, 250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.
2 Department of Computer Science and Information Engineering, Asia University, Taiwan, R.O.C.
* Corresponding author: mshwang@asia.edu.tw

Abstract. In this article, we propose a protocol to generate \( n^2 \) keys in one session under the assumption of the intractability of the elliptic curve discrete logarithm problem and MQV protocol. Our protocol has the advantage of requiring less computing time compared with other protocols.

Key words: Cryptography, Elliptic curve discrete logarithm, key agreement, key authentication

1 Introduction

In 1976, Diffie and Hellman [5] proposed the first practical solution to key agreement problem for two parties to establish a session key, which can be applied to provide security or data integrity for later communications between the two parties. Unfortunately, the original Diffie-Hellman protocol suffers from the “man-in-the-middle” attack because of lack of authentication between two communication parties. Over the years, many solutions [4, 8–11, 17] have been developed to solve this problem.


In order to enable two communication parties to establish multiple common secret keys between two parties efficiently, in 1998 Harn and Lin [6] developed an authenticated key agreement protocol based on the MQV protocol without using any one-way hash function. In Harn-Lin protocol, two parties can authenticate each other and establish \( n^2 \) common session keys. The Harn-Lin protocol sets up the limit that only \( (n^2 - 1) \) common session keys can be used to avoid the known key attack [16].

Later, Yen and Joye [21] pointed out a security problem and a solution in the Harn-Lin protocol. This security problem let an attacker impersonates one
party to generate common session secret keys with another party by forging a signature message modified by the previous one, called a forgery attack. However, according to Wu et al. [19], the Yen-Joye protocol cannot withstand the same attack from which Harn-Lin protocol suffers. Afterward, we proposed an improved protocol [8] to improve the Yen-Joye protocol.

In 2001, Harn and Lin [7] then modified the signature in which the equation is signed [6] to prevent the forgery attack. But still only \((n^2-1)\) common session keys are allowed to be used in their protocol. In 2002, Tseng [18] proposed a robust protocol that even two parties use all the \(n^2\) common session keys, the known-key attack can be avoided. Until now, there are many schemes had been proposed [2, 3, 12, 13, 20].

In this article, we propose a protocol based on the elliptic curve discrete logarithm problem to generate \(n^2\) common session-keys in one session, and all the keys can be used to withstand the known-key attack.

2 Review Tseng’s Protocol

In this section, let us briefly review Tseng’s protocol that can establish \(n^2\) common session keys between two parties. The protocol is divided into two phases: the initiation phase and the multiple-key agreement phase. To give a simple example, let’s assume that Bob and Alice want to establish four common session keys by using 2 short-term secret keys. They are required to go through the following processes:

**The initiation phase:** In the Diffie-Hellman scheme, the system publishes a large prime number \(p\). Bob and Alice select their random numbers \(x_A\) and \(x_B\) and compute the corresponding long-term public keys \(y_A = g^{x_A} \mod p\) and \(y_B = g^{x_B} \mod p\), respectively.

**The multiple-key agreement phase:**

1. Alice selects two random short-term secret keys \(k_{A1}\) and \(k_{A2}\), where \(k_A = k_{A1} + k_{A2} \mod q\), and then calculates the corresponding short-term public keys \(r_A = g^{k_A} \mod p\), \(r_{A1} = (y_B)^{k_{A1}} \mod p\) and \(r_{A2} = (y_B)^{k_{A2}} \mod p\). After obtaining the signature \(s_A\) based on the equation \(s_A r_A = x_A - r_{A1} k_A \mod q\), Alice sends \(\{r_{A1}, r_{A2}, s_A, \text{Cert}(y_A)\}\) to Bob, where \(\text{Cert}(y_A)\) is a certificate for the public key signed by a trustworthy party.

2. In the same way as Alice does, Bob also generates \(k_{B1}, k_{B2}, r_{B1}, r_{B2}\) and \(s_B\) and sends \(\{r_{B1}, r_{B2}, s_B, \text{Cert}(y_B)\}\) to Alice.

3. Alice verifies the messages \(\{r_{B1}, r_{B2}, s_B, \text{Cert}(y_B)\}\) from Bob, and furthermore checks the following equation:

\[
y_B = (r_B)^{r_{B1}} g^{s_Br_B} \mod p, \tag{1}
\]

where \(r_B = r_{b1} r_{b2} \mod p\), \(r_{b1} = (r_{B1})^{x_A^{-1}} \mod p\) and \(r_{b2} = (r_{B2})^{x_A^{-1}} \mod p\). If the above equation is correct, Alice will compute four common
session keys as follows:

\[
K_1 = r_{b1}^{k_{a1}} \mod p, \\
K_2 = r_{b1}^{k_{a2}} \mod p, \\
K_3 = r_{b2}^{k_{a1}} \mod p, \\
\text{and} \\
K_4 = r_{b2}^{k_{a2}} \mod p.
\]

Just like Alice, Bob also verifies the authenticated messages and generates four common secret keys: \(K_1 = r_{b1}^{k_{a1}} \mod p, K_2 = r_{b2}^{k_{a1}} \mod p, K_3 = r_{b1}^{k_{a2}} \mod p \) and \(K_4 = r_{b2}^{k_{a2}} \mod p\).

## 3 The Proposed Protocol

In this section, we shall propose a more efficient protocol to establish \(n^2\) common session keys between two parties based on the elliptic curve discrete logarithm problem. The protocol is composed of two phases: the initiation phase and the multiple-key agreement phase. Let’s assume that Bob and Alice want to establish four common session keys by using 2 short-term keys. The following two phases enable Alice and Bob to authenticate each other and to generate multiple common secret keys.

### The initiation phase:

The system publicly chooses an elliptic curve \(E \) over a finite field \(GF(q)\) and a base point \(G\) with order \(p\) \([15]\). Bob and Alice choose their secret key \(k_A \in [1, p - 1]\) and \(k_B \in [1, p - 1]\), and compute the corresponding long-term public keys \(Q_A = k_A G\) and \(Q_B = k_B G\), respectively.

### The multiple-key agreement phase:

1. Alice chooses two short-term secret keys \(k_{a1} \) and \(k_{a2}\), and then computes the corresponding short-term public keys \(R_{A1} = Q_{B} k_{a1} = k_{A1} G\) and \(R_{A2} = Q_{B} k_{a2} = k_{A2} G\). After getting the signature \(s_A\) based on the equation \(s_A = (R_{A2})^{-1} (Q_{A} - k_{A1} R_{A1}) \mod q\), Alice sends \(\{R_{A1}, R_{A2}, s_A, \text{Cert}(Q_A)\}\) to Bob, where \(\text{Cert}(Q_A)\) is a certificate for the public key signed by a trustworthy party.

2. Just as Alice does, Bob also generates \(k_{B1}, k_{B2}, R_{B1}, R_{B2}\) and \(s_B\) and sends \(\{R_{B1}, R_{B2}, s_B, \text{Cert}(Q_B)\}\) to Alice.

3. Alice verifies the authenticated messages \(\{R_{B1}, R_{B2}, s_B, \text{Cert}(Q_B)\}\) from Bob, and then checks the following equation:

\[
(R_{B2})_x (R_{B1})_x Q_B + S_B R_{B1} = (R_{B2})_x (Q_B)_x G.
\]  

(2)

4. If the equation is correct, Alice computes the following equations:

\[
R_{b1} = R_{B1} k_A^{-1} = k_{b1} G \mod p, \\
R_{b2} = R_{B2} k_A^{-1} = k_{b2} G \mod p.
\]
5. Alice generates four common session keys as follows:

\[ K_1 = R_{b1}k_{A1} \mod p, \]
\[ K_2 = R_{b1}k_{A2} \mod p, \]
\[ K_3 = R_{b2}k_{A1} \mod p, \]
and
\[ K_4 = R_{b2}k_{A2} \mod p. \]

Like Alice, Bob also verifies the authenticated messages and generates four common secret keys: \( K_1 = R_{a1}k_{B1} \mod p, K_2 = R_{a2}k_{B1} \mod p, K_3 = R_{a1}k_{B2} \mod p, \) and \( K_4 = R_{a2}k_{B2} \mod p. \)

4 Conclusions

In this paper, we apply the elliptic curve discrete logarithm problem (ECDLP) to establish \( n^2 \) common session keys between two parties in one session. This protocol is more efficient than [18]. The proposed protocol is also secure to against the known-key attack, replay attack, and forgery attack.

References