An Efficient Self Proxy Signature Scheme Based on Elliptic Curve Discrete Logarithm Problems

Nedal Tahat, K. A. Alzu'bi and I. Abu-Falahah

Department of Mathematics, Faculty of Sciences.
The Hashemite University, Zarqa 13133, Jordan
nedal@hu.edu.jo, khaledaa@hu.edu.jo, iabufalahah@hu.edu.jo

Abstract

Self proxy signature is a type of proxy signature wherein, the original signer delegates the signing rights to himself, thereby generating temporary public and private key pairs for himself. The aim of self proxy is to protect the signer's permanent secret key. Most of proposed self proxy signature scheme have been based on discrete logarithms which required for a protocol to generate and verify a self proxy signature. Therefore, to guarantee the quality of the growing popular communication services, it is urgent to construct low-computation self proxy signature scheme. For this reason, in this paper we propose an efficient self proxy digital signature scheme based on elliptic curve discrete logarithm problem (ECDLP). The scheme based on elliptic curve cryptosystem (ECC) are more efficient than those associated with other cryptosystems, like the RSA and the DSA security solutions. Furthermore, The scheme require less number of operations than Mashhadi scheme [10] and so is more efficient than Mashhadi scheme. Next we show that the proposed scheme satisfies the undeniability, unforgeability and distinguishability properties.

Keywords: Self proxy signatures, Elliptic curve discrete logarithm, Distinguishability, Unforgeability, Undeniability

1 Introduction

The notion of proxy signature scheme dates back to 1996, proposed by Mambo, Usuda and Okamoto in their seminal paper [3]. In proxy signature scheme, a user
Alice, called the original signer delegates her signing rights to another user Bob, called the proxy signer. A verifier can distinguish between a normal signature and proxy signature but then is convinced that the message is authenticated by Alice. Since then, many proxy signature schemes have been proposed [1, 2, 8, 9]. Recently, the concept of self proxy signature scheme was introduced by Kim et al. [12]. In self proxy signature, a user delegates his signing rights to himself, i.e. the user can generate multiple pairs of temporary public and private keys. The lifetime of the temporary keys can be controlled by creating proper message warrants, depending on the application. In 2012, Mashhad [10] proposed a novel secure self proxy signature scheme based on DLP. Furthermore, Mashhad in his paper showed that some security leaks inherent in Kim et al. scheme and showed that an Adversary can forge a valid self proxy signature scheme for any message by using different ways. The security of Mashhadi self proxy signature scheme based on the difficulty of solving the discrete logarithm problem. Such a self proxy signature, require more number of modular multiplications in the signature generation and signature verification.

To optimize the trade-off between performance and security, the proposed scheme is based on the elliptic curve cryptosystem. The ECC was initiated by Koblitz [5] and Miller [11], where the security was established on the discrete logarithm problem over the points on an elliptic curve, called ECDLP. The basic operation are the executing of integer points on the elliptic curve over finite fields, including addition and multiplication. The operations associated with ECC are more efficient than those associated with other cryptosystems, like the RSA and the DSA security solutions. Owing to the fact that the ECC has a smaller key size and faster computation, therefore it is being gradually given more important by the academic and industrial circles.

2 Some Notations

The following parameters and notations will be used throughout this paper unless otherwise specified

- A field size \( q \), where either \( q = p \) in case that \( p \) is an odd prime (the common practice), or \( q = 2^m \) in case that \( q \) is a prime power.
- Two parameter \( a, b \in F_q \) to define the elliptic curve equation \( E \) over

\[
F_q : y^2 = x^3 + ax + b \quad (mod \ q)
\]

in case that \( q > 3 \), where \( 4a^3 + 27b^2 \neq 0 \ (mod \ q) \). \( E \) should be divisible by a large prime number with regard to the security issue raised by Pohlig and Hellman [13].
- An elliptic point \( G \) whose order is a large prime number in \( E(F_q) \), where \( G \neq O \) (\( O \) denotes infinity) such that the order of \( G \) is \( n \).
- \( h(.) \) A public cryptographically strong hash function.
• \( m_w \): the warrant which specifies the delegation period for the kind of message \( m \) is delegated, the identities of the signer, etc.

A signer Alice selects a private key \( d_a \in \mathbb{Z}_n^* \), and obtains a public key \( e_a = d_a G \pmod{n} \)

3 The Proposed Scheme

We now present the an efficient self proxy based on ECDLP. Our scheme is based on the normal proxy signature scheme in which a signer Alice delegates her capability to here self recursively. Our scheme can be divided into four phases: the setup, key generation, signature generation and verification phases.

3.1 Setup Phase

Alice generates the pair \((d_a, e_a)\) of private key \(d_a\) and public key \(e_a\) as follows:

She randomly pick an integer \( d_a \in \mathbb{Z}_n^* \), computes \( e_a = d_a G \pmod{n} \) and sends \( e_a \) to the certificate authority (CA). Then CA randomly chooses \( d_0 \in \mathbb{Z}_n^* \), computes \( C = d_0 G (\pmod{n}) \) and returns \( C \) to Alice. Alice computes \( C_a = d_a C (\pmod{n}) \) and sends \( C_a \) to CA. Then CA checks the equality \( d_0 e_a = C_a \); if it holds, CA accepts their certification, otherwise he refuses it.

3.2 Self Proxy Key Generation Phase

The signer Alice generates the temporary self proxy private - public key pair by using her original signing key pair \((d_a, e_a)\) as follows:

The signer Alice chooses random numbers \( k, d_t \in \mathbb{Z}_n^* \) and computes

\[ y = kG \pmod{n} = (x_1, x_2), \text{ where } u \equiv x_1 \pmod{n} \]
\[ e_t = d_t G \pmod{n} \]

Then she computes self proxy private key as

\[ d_p = k + (d_a + d_t) h(m_w) u \pmod{n} \]

and obtain its corresponding self proxy public key \( e_p = d_p G \pmod{n} \). Finally she publish \( e_t \).

3.3 Self Proxy Signature Generation Phase

Suppose that Alice want to generate a self proxy signature of a message \( m \). She performs the following operations:

The signer Alice chooses \( k' \in \mathbb{Z}_n^* \) randomly, and computes \( y' \) and \( \mu' \) as follows:

\[ y' = k' G (\pmod{n}) = (x'_1, x'_2), \text{ where } u' \equiv x'_1 \pmod{n} \]
\[ \mu' = (k' + d_p h(m, m_w) u') \pmod{n} \]

Then send \((m, (y', \mu'), y, m_w)\) to verifier Bob.
3.4 Self Proxy Signature Verification Phase

First of all, the verifier Bob checks the signer’s identity and the delegation lifetime of the warrant $m_w$. If all validations hold, the verifier Bob follows the next operations:

The verifier Bob recover the self proxy public key as follows:

$$e_p = (\gamma + h(m_w) u (e_a + e_t) ) (mod\ n) = (e_1, e_2)$$

The verifier check the validity of the next equality

$$\mu G = (\gamma' + h(m, m_w) u' e_p) (mod\ n)$$

If the equality holds, he accepted $(m, (\gamma', \mu'), \gamma, m_w)$ as the valid self proxy signature.

The correctness of the scheme is as follows:

**Theorem 1**: If $(m, (\gamma', \mu'), \gamma, m_w)$ is a signature of the message $m$ produced by a proposed self proxy signature scheme, then

$$\mu G = (\gamma' + h(m, m_w) u' e_p) (mod\ n)$$

**Proof**: The equation in verifying algorithm is true for valid self proxy signature since,

$$\mu G = (\gamma' + d_p h(m, m_w) u' ) G$$

$$= (\kappa G + d_p h(m, m_w) u G )$$

$$= (\gamma' + e_p h(m, m_w) u' ) (mod\ n)$$

Which means that $(m, (\gamma', \mu'), \gamma, m_w)$ is a valid signature of $m$. So, our proposed scheme provides a self proxy signature scheme.

4 Security Analysis

In this section, we discuss some security properties of our self proxy signature scheme. A secure self proxy signature scheme should satisfy the following requirement and we show that our proposed scheme satisfied the requirements.

a) Unforgeability: Only the valid signer can create the self proxy signature
b) Undeniability: The signer cannot deny his/her signatures to anyone.
c) Distinguishability: The self proxy signature must be distinguishable from the normal signature.
4.1 Unforgeability Property

Based on ECDLP, it is virtually impossible to obtain Alice's self proxy private key \( d_p \) from the corresponding public key \( e_p \). Also, it is very difficult for anyone to obtain \( d_p \) from the self proxy signature \( \mu \). Hence, \( d_p \) can be kept secretly and be reused. Therefore, Adversary (Adv) has to forge the valid signature \((m, (\gamma', \mu'), \gamma, m_w)\) on message \( m \) without the private key \( d_p \) by the following attacks:

**Attack 1**: We will show that after intercepting a valid self proxy signature \((m, (\gamma', \mu'), \gamma, m_w)\), Cindy cannot forge a valid self proxy signature for a message \( m_1 \). Suppose Cindy wants to forge a self proxy signature \((m_1, (\gamma_1, \mu_1), \gamma, m_w)\) for a message \( m_1 \), and claim dishonestly that has been generated by Alice. For this purpose, he chooses random integers \( \delta \in \mathbb{Z}_n^* \). Now, he should calculate \( \gamma_1 = (u_1, u_2) \) and \( \mu_1 \) as follows:

\[
\gamma_1 = (\delta G - h(m_1, m_w)u_1 e_p) \pmod{n}, \quad \text{where } u_1 \text{ is the first component of } \gamma_1 \\
\mu_1 \equiv \delta \pmod{n}
\]

However, he should solve ECDLP \( e_p = d_p G \pmod{n} \) in order to compute the above \( \gamma_1 \). Therefore, no Adversary can forge a valid self proxy signature by this attack.

**Attack 2** Suppose Cindy tries to forge a valid self proxy signature \((m_1, (\gamma_1, \mu_1), \gamma_1, m_w)\). For this purpose, he can generate the desired warrant \( m_w \) and choose random integers \( \delta, \sigma \in \mathbb{Z}_n^* \) and then compute \( \gamma_1 = \sigma G \pmod{n} = (u_1, u_2) \). Now, Cindy should calculate

\[
\gamma_1 = \delta G - h(m_1, m_w)u_1 (e_a + e_t) \pmod{n} \\
\mu_1 = (\sigma + \delta h(m_1, m_w)u_1) \pmod{n}
\]

However, he should solve ECDLP \( (e_a + e_t) = (d_p + d_t)G \pmod{n} \) in order to compute the above \( \gamma_1 \). Therefore, no Adversary can forge a valid self proxy signature by this attack. Consequently, because the two attacks are not possible, it is computationally difficult for the attacker Cindy to forge the self proxy signature. Therefore the proposed scheme satisfies the unforgeability property.

4.2 Undeniability Property

In the proposed scheme, any valid self proxy signature \((m, (\gamma', \mu'), \gamma, m_w)\) for a message \( m \) should be generated by Alice. This is because only Alice has the self private key \( d_p \). Moreover, the warrant \( m_w \) and temporary self proxy public key \( e_p \) are created by Alice and no Adversary can change them. When the self proxy signature \((m, (\gamma', \mu'), \gamma, m_w)\) is verified the warrant \( m_w \) is checked, and the public key of signer, \( e_a \), the temporary self proxy public key, \( e_p \), and the public information \( e_t \) are used in the verification phase. Thus, Alice cannot deny signing
the self proxy signature. Therefore the proposed scheme satisfies the undeniability property.

4.3 Distinguishability Property
In the proposed scheme, when the self proxy signature \((m, (\gamma, \mu), \gamma, m_\mu)\) is verified, Alice's public key and her identity are used in the verification phase; therefore, we can consider it as a self proxy signature and not a normal signature. Thus, anyone can distinguish the self proxy signature from normal signatures. Thus the proposed scheme satisfies the distinguishability property.

5 Performance
In this section, in terms of computational complexity, we compare our scheme with Mashhadi scheme and summarize the result in Table 1. We use the following notations to analyze the computational complexity.

- \(T_{\text{mul}}\) is a time complexity for executing the modular multiplication,
- \(T_{\text{exp}}\) is a time complexity for executing the modular exponentiation,
- \(T_{\text{ec-add}}\) is a time complexity for executing the addition of two elliptic curve points,
- \(T_{\text{ec-mul}}\) is a time complexity for executing the multiplication on elliptic curve points,
- \(T_h\) is a time complexity for performing a one-way hash function.

To describe the efficiency performance in terms of \(T_{\text{mul}}\), we use the following conversion \([6, 7]\). This conversion convert various operations units to the time complexity for executing the modular multiplication.

\[
T_{\text{exp}} \approx 240T_{\text{mul}}; \ T_{\text{ec-mul}} \approx 29T_{\text{mul}}; \ T_{\text{ec-add}} \approx 0.12T_{\text{mul}}
\]

Table 1: Comparison of computational complexity

<table>
<thead>
<tr>
<th>Items</th>
<th>Scheme by Mashhadi</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time complexity</td>
<td>Complexity in (T_{\text{mul}})</td>
</tr>
<tr>
<td>Signature generation</td>
<td>(T_{\text{exp}} + 2T_{\text{mul}} + T_h)</td>
<td>(242T_{\text{mul}} + T_h)</td>
</tr>
<tr>
<td>Verification</td>
<td>(3T_{\text{exp}} + 5T_{\text{mul}} + 2T_h)</td>
<td>(725T_{\text{mul}} + 2T_h)</td>
</tr>
</tbody>
</table>
Table 1 summarize difference between these three schemes. From the statistics in Table 1, it can be seen that be it the signature generation phase or the signature verification phase, the number of modular multiplications required by our scheme is less than that required by Mashhadi scheme. Therefore, our scheme can substantially raise the efficiency of signature generation and signature verification.

5. Conclusions

The security of Mashhadi self proxy signature scheme [10] is constructed on the discrete logarithm problem while the security of the proposed scheme is based on the difficulty of solving the elliptic curve discrete logarithm problem. According to [4], the elliptic curve discrete logarithm is significantly more difficult than DLP. For the most part, the well-known DSA system must use 1024 bit keys, only then can it attain computationally reasonable security; the ECC needs only 160 bit keys. So, at the same level of security, the speed of ECC is several times faster than DSA system; it can also saves on key storage space. Clearly, whether it is in terms of security or in performance, the proposed scheme is superior to Mashhadi self proxy signature scheme.

References


Received: May 27, 2013