A Novel Proxy Signcryption Scheme and its Elliptic Curve Variant

Roayat Ismail Abdelfatah
Faculty of Engineering
Tanta University
Egypt

ABSTRACT
Proxy signcryption scheme allows an original signer to delegate his signing power to a proxy such that the latter can signcrypt a message on behalf of the former. In this paper, a new proxy signcryption scheme is proposed based on Discrete Logarithm Problem (DLP) and Diffie-Hellman Problem (DHP) with a reduced computational cost compared to other schemes in literature. The proposed scheme achieves public ciphertext authentication as the signcrypted message before being accepted, the receiver first verifies the signature. This property is very useful as the receiver can filter some incorrect ciphertext before decrypting it which achieves more efficient unsigncryption. Also, a variant of the main scheme that works over elliptic curves will be considered, since it has proven to provide better security with shorter keys and hence less storage requirements which makes it more suitable for resource constrained devices such as pagers and mobile phones. Numerical examples have been given with Mathematica to emphasize the ease of its practical use.

Keywords
Proxy signcryption, Discrete Logarithm Problem (DLP), Elliptic Curve Cryptography.

1. INTRODUCTION
The public key cryptography was firstly proposed by Diffie and Hellman [1]. Since then, public key encryption and digital signature are important tools to communicate with each other in a secure and authenticated way over open and insecure channels. The proxy signature is a cryptographic primitive before that was first introduced by Mambo et al. [2] in 1996. The scheme allows an entity, called the original signer to designate another entity, called a proxy signer, to sign messages on its behalf. The proxy signature primitive has found numerous practical applications, particularly in distributed computing where delegation of rights is quite common, such as in e-cash systems, global distribution networks, grid computing, mobile agent applications, and mobile communications. Based on the delegation type, there are three types of proxy signature [3-5]: full delegation, partial delegation, and delegation by warrant. In full delegation schemes, a proxy signer is given the same secret key SK that the original signer has, so that both create the same signatures. Obviously, when the proxy signer deliberately signs a document unfavorable to the original signer, this misbehavior is not detected because the signature created by the proxy signer is indistinguishable from the signatures created by the principal signer. In partial delegation schemes, a new secret key SK’ is given securely to the proxy signer. The proxy signature is checked by the modified equation and not by the original equation. This implies that a signature created by the proxy signer is distinguishable from a signature created by the original signer. In such delegation schemes, only the public key of the original signer is required for the signature verification. Partial delegation is further classified as proxy-protected and proxy-protected according to protection of proxy signer. In a proxy-protected scheme, only the proxy signer can generate valid proxy signatures, while in a proxy-unprotected scheme, either the original signer or the proxy signer can produce proxy signatures because both have knowledge on the proxy private key. In many applications, proxy-protected signature schemes are required to avoid the potential dispute between the original signer and the proxy signer. Kim et. al. [6] gave a new type of delegation called partial delegation with warrant, which certifies that the proxy is exactly the signer to be entrusted. This type of delegation is more secure than the full delegation as the created proxy signatures are distinguishable from ordinary signatures. Confidentiality, integrity, non-repudiation, and authentication are the important requirement for many cryptographic applications. Traditional approach to achieve these requirements is to sign a message and then encrypt the signature. In 1997, Zheng [7] proposed a cryptographic primitive, called signcryption, to achieve the combined functionalities of digital signatures and encryption in an efficient manner. Since then, some research works on signcryption have been carried out [8-17]. In 1999, Gamage et al. [18] extended the proxy signature and introduced a proxy signcryption scheme by combining proxy signature and encryption technology. It allows an entity to delegate his authority of signcryption to a trusted agent. Gamage’s scheme is based on Discrete Logarithm Problem (DLP) in the traditional PKI based setting. The proxy signcryption scheme is a useful cryptographic tool. Let us consider a scenario that a boss in a company can delegate his or her capability of signing a message to an entity, when absence of time. If the message involves some sensitive information, the traditional proxy signature cannot satisfy this requirement. However, a proxy signcryption scheme just solves the problem. Many research works on proxy signcryption have been introduced [19-21].

In this paper, a new efficient proxy signcryption scheme based on delegation of signing rights using warrants is proposed. The security of the proposed scheme is based on the hardness of the DLP. The security aspects of the proposed scheme are discussed showing that proposed scheme satisfies several desirable requirements with a better efficiency compared to other schemes in literature. Moreover, a more efficient variant of the proposed proxy signcryption scheme based on elliptic curve DLP (ECDLP) and ECDHP is presented along with its related analysis. The organization of the reset of the paper is as follows. Section 2 discusses the computationally hard problems; both the DLP, the DHP and the related ECDLP and ECDHP. Section 3 introduces the structure and security properties of the proposed scheme; section 4 discusses the proposed scheme together with its proof of correctness, security analysis and performance analysis; section 5 introduces a
variant of the proposed proxy signcryption scheme based on the ECDLP and ECDHP. A long with its related analysis. Finally, a numerical example with Mathematica10 program is given in section 6 to demonstrate the ease of implementation of the proposed scheme.

2. COMPUTATIONALLY HARD PROBLEMS

2.1 Discrete Logarithm Problem (DLP)
Let \( p \) and \( q \) be two large primes satisfying \( q/p - 1 \), and \( g \) a generator of order \( q \) over \( GF(p) \). The discrete logarithm problem is, given an instance \((y, p, q, g)\), where \( y = g^x \mod p \) for some \( x \in \mathbb{Z}_q \), to derive \( x \) [22,23].

2.2 Discrete Logarithm (DL) Assumption
A probabilistic polynomial-time algorithm \( B \) is said to \((t, \ell)\) break the DLP if given an instance \((y, p, q, g)\) where \( y = g^x \mod p \) for some \( x \in \mathbb{Z}_q \), \( B \) can derive with probability \( \ell \) after running at most \( t \) steps. The probability is taken over the uniformly and independently chosen instance and over the random bits consumed by \( B \). The \((t, \ell)\) DL assumption holds if there is no probabilistic polynomial-time adversary that can \((t, \ell)\) break the DLP [22, 23].

2.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)
An elliptic curve group is described using multiplicative notation, then the elliptic curve discrete logarithm problem is: given points \( P \) and \( Q \) in the group \( \mathbb{Z}_p^* \), find a number such that \( kP = Q; k \) is called the discrete logarithm of \( Q \) to the base \( P \) [22, 24].

2.4 Diffie-Hellman Problem (DHP)
Let \( p \) and \( q \) be two large primes satisfying \( q/p - 1 \), and \( g \) a generator of order \( q \) over \( GF(p) \). The Diffie-Hellman Problem (DHP) is, given an element \( a^b \mod p \) and \( a^c \mod p \) and, find \( a^{bc} \mod p \).

2.5: The Elliptic Curve Diffie - Hellman Problem (ECDHP)
The elliptic curve Diffie-Hellman problem is: given three points \( P, R, Q: R = xP \) and \( Q = yP \) in the group \( \mathbb{Z}_q^* \), find \( xyP \).

3. STRUCTURE AND SECURITY PROPERTIES
The proxy signcryption scheme can be viewed as the combination of a general proxy signature and a signcryption scheme. A proxy signcryption is a cryptographic primitive involving three entities: an original signer/sender (O), a proxy signer (P), and a verifier/receiver (R). Each entity has a secret and a corresponding public key.

3.1 Structure
A proxy signcryption scheme consists of the following algorithms:

1. Setup: This algorithm generates the system parameters.
2. Proxy Designation Protocol: This is an interactive protocol between the two parties (the original signer and his proxy agent). It takes the system parameters and private key of the original signer and outputs a corresponding proxy signing which the proxy agent can use to produce proxy signatures on behalf of the original signer.
3. Proxy Signcryption: It is usually a randomized algorithm. It takes as input the system parameters, the private key of the proxy signer, a message \( m \), the public key of the intended receiver, and outputs a proxy signcrypted text.
4. Proxy Unsigncryption and Verification: It is a deterministic algorithm that takes as inputs the system parameters, the signcrypted text, the private key of the receiver and the public keys of the original and proxy signers. It validates the alleged signature of the proxy signer as well as decrypts the contents of the ciphertext part of the signcrypted text.

3.2 Security Properties
The scheme should achieve all the following proxy signcryption scheme properties:

1. Correctness: A properly formed signcrypted ciphertext by the signcryption algorithm must be accepted by the unsigncryption algorithm.
2. Confidenziality: Without the knowledge of the sender or the designated receiver's private key, it should be infeasible for an adaptive attacker to gain any partial information on the contents of the signcrypted ciphertext.
3. Distinguishability: The proxy signature must be distinguishable from the normal signature.
4. Unforgeability: A designated signer, called proxy signer, can create a valid proxy signature for the original signer. But the original signer and third parties who are not designated as a proxy signer cannot create a valid proxy signature
5. Identifiability: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.
6. Verifiability: Validity of a proxy signature as well as the original signer's delegation on signature signing on a message can be verified using public parameters.
7. Non-repudiation of proxy signing: It is difficult for a proxy signer to falsely deny having signed its proxy signatures.
8. Non-repudiation of signature delegation: It is difficult for an original signer to falsely deny that it has delegated the signing power to a proxy signer.
9. Prevention of proxy key misuse: It should be confident that proxy key pair should be used only for creating proxy signature, which conforms to delegation information. In case of any misuse of proxy key pair, the responsibility of proxy signer should be determined explicitly.

4. THE PROPOSED SCHEME
First, we assume that all the parties: the original signcryter, the proxy signcryption, and the intended recipient are members of some common public key infrastructure.

The original signer key pair is \((x_s, y_s)\), the proxy signer key pair is \((x_p, y_p)\), and the recipient key pair is \((x_r, y_r)\).

4.1 The Proposed Scheme Construction
4.1.1. Setup: On input a security parameter \( k \), the setup algorithm generates the two primes \( p, q \) such that \( 2^{k-1} \leq p \leq 2^k \) and \( q \) divides \( p - 1 \). Moreover, it outputs an element \( g \in \mathbb{Z}_p \).
$Z_p^*$ of order $q$ and a hash function $H: \{0,1\}^* \rightarrow Z_q$.

Furthermore, a secure symmetric key encryption algorithm, $(E, D)$, must be agreed upon. The system parameters = $(p, q, g, H)$ are then published.

4.1.2 Key Generation: The original user $U_i$ chooses his private key $x_i \in Z_q$ and computes the public key:

$y_i = g^{x_i} \mod p$. The proxy $U_p$ chooses his private key $x_p \in Z_q$ and computes the public key as $y_p = g^{x_p} \mod p$. The recipient $U_r$ chooses his private key $x_r \in Z_q$ and computes the public key as $y_r = g^{x_r} \mod p$. Both the proxy $U_p$ and recipient $U_r$ use Diffie-Hellman protocol [only one time (offline) before the transmission of any message] to exchange a shared secret key $y_{sh}$ as follows:

The proxy $U_p$ computes: $y_{sh} = (y_r)^{x_p} \mod p = g^{x_r x_p} \mod p$

The recipient $U_r$ computes: $y_{sh} = (y_p)^{x_r} \mod p = g^{x_r x_p} \mod p$

4.1.3 Proxy Designation Protocol

The original user $U_i$ delegates his signing power to a proxy signcryption $U_p$ as follows: $U_i$ first choose a random integer $d \in Z_q$ and compute $t = (g^d \mod p) \mod q$ and

$\sigma \equiv \left(\frac{y_i}{y_i^d} - H(m_w, t)\right) \mod q$, where $m_w$ is a warrant consisting of the identifiers of the original signer and proxy signers, the delegation duration and so on. $(\sigma, m_w, t)$ is then sent to $U_p$. Upon receiving $(\sigma, m_w, t)$, $U_p$ computes:

$(\gamma_i \sigma + H(m_w, t) \mod p) \mod q$ and performs check its validity as follows: $t \equiv (\gamma_i^t \sigma + H(m_w, t) \mod p) \mod q$. If $t$ is not equal to the right-hand side, the proxy requests a new $(\sigma, m_w, t)$ to be sent again. The verification of the above equation proceeds as follows:

$y_i^{\sigma + H(m_w, t) \mod p} \equiv g^{x_i (\frac{y_i}{y_i^d} - H(m_w, t) + H(m_w, t)) \mod p} \equiv g^\sigma \mod p \equiv t$

After the proxy authenticates the original signer, the proxy computes the secret key as follows: $x_{pr} \equiv (\sigma + x_p) \mod q$ as his proxy signature, secret key. Then he computes and publishes the corresponding proxy public key:

$y_{pr} = g^{x_{pr}} \mod p$

Note that: $y_{pr} = g^{x_{pr}} \mod p = g^{(\sigma + x_p) \mod q} \mod p = g^{\frac{y_i}{y_i^d} \mod q} \mod p$

4.1.4. Proxy Signcryption

The proxy will do the following steps to sign and encrypt the message $m$. The proxy chooses a random number $w \in Z_q$ and computes:

1. $k \equiv H((g^w \mod p) \mod q)$
2. $z \equiv H((y_{sh} \cdot k) \mod p) \mod q$  
3. $C = E_k(m)$
4. $V = H(C, k)$

$5. S \equiv \left(\frac{w}{xy_{pr}} - V\right) \mod q$

The proxy sends $\delta = (C, V, S)$ to the receiver.

4.1.5. Proxy Unsigncryption and Verification

Upon receiving $(C, V, S)$, the receiver decrypts the message and checks the signature validity as follows:

1. Recover the key $k$ by computing:

$k = H((y_{pr}^{(S+V)} \mod p) \mod q)$

2. Verify that the received signature is valid by computing:

$V = H((C, k') \mod q)$ and accepts if $V = V$

3. Compute $z = H((y_{sh} \cdot k') \mod p) \mod q$, where $y_{sh}$ is the shared secret key between the proxy and the receiver.

4. Decrypt the ciphertext $m = D_z(C)$.

The signature verification equation involves the ciphertext instead of the plaintext message. Moreover, the equation for recovering $k$ does not involve the private key of the recipient or any private information. Consequently, the signature part of the signcrypted text can be verified by any third party.

The proposed scheme achieves public signciphertext authentication: the signcrypted message $\delta = (C, V, S)$ before being accepted, the receiver first verifies the signature. This property is very useful as the receiver can filter some incorrect signciphertext before decrypt it which achieves more efficient unsigncryption.

4.2 Proof of Correctness

The following equations demonstrate the correctness of the proposed scheme

$k' = H((y_{pr}^{(S+V)} \mod p) \mod q)$

$(y_{pr}^{(S+V)} \mod p \mod q) = H(g^{x_{pr} \mod q} \mod p \mod q) = H((g^w \mod p) \mod q) = k$

4.3 Security Analysis

1. Confidentiality: The only way to decrypt $C$ and obtain the message $m$ is to have the shared-secret key $y_{sh}$. But it is difficult to obtain this key from the public keys $y_p$ and $y_r$ due to the intractability of DHP over the finite field $Z_p^*$. For a passive adversary, the information available is only $(C, V, S)$. From this data, he can only obtain $k = H((y_{pr}^{(S+V)} \mod p) \mod q) = H((g^w \mod p) \mod q) \mod q$ but he cannot guess the corresponding $m$. Also, it is difficult to obtain the secret value $w$ from $g^w \mod p$ due to the intractability of the DLP. If an intruder intends to reveal the secret parameters $x_{pr}$ and $w$ from $S \equiv \left(\frac{w}{xy_{pr}} - V\right) \mod q$, this will be difficult because there are two unknown variables ($x_{pr}, w$) in one equation. Then even if $x_{pr}$ is revealed, a person cannot compute $y_{sh}$ as we mentioned previously and so cannot decrypt $C$. Therefore, the proposed scheme provides forward secrecy property with respect to the proxy signer.

2. Verifiability: During the unsigncryption, the receiver can be convinced that the proxy sender has the original signer’s signature on the warrant by using $y_{pr}$ which includes the private key of both the original signer and the proxy signer besides the random value $d$ and the warrant $m_w$ which contains the identity information of the original sender and the
limit of delegated signcrying capacity. Therefore, the receiver can be convinced of the original sender’s agreement on the signcrypted message. Thus, the scheme satisfies the verifiability requirement.

3. **Unforgeability**: According to the Eqn. \( x_{pr} \equiv (\sigma + x_p) \mod q \) the proxy sender includes his private key \( x_p \) in the computation of his proxy signcryption key, so no one can forge this key. For anyone who want to forge the proxy signature \( S \equiv \left( \frac{w}{x_{pr}} - V \right) \mod q \), he must know \( w \) and \( x_{pr} \), which both protected by DLP. Thus, no one except the proxy sender can create a valid proxy signcryption. So, the proposed scheme achieves unforgeability.

4. **Distinguishability**: This is obvious, because there is a warrant \( m_w \) in a valid proxy signature, at the same time the proxy signature public key \( y_{pr} \) which includes the private key of both the original signer and the proxy signer. So, the proxy signature is easy to be distinguishable from the normal signature.

5. **Prevention of Misuse**: Using the warrant \( m_w \) in our scheme had determined the limit of the delegated signing capacity. So, the proxy signer cannot sign some messages that have not been authorized by the original signer and this prevent abuse of the proxy key.

6. **Nonrepudiation**: Since the original signer is the only one who can compute \( \sigma \) as his private key \( x_i \) is included in it, he cannot repudiate the signing capability delegation to the proxy signer and this can be verified publicly as we indicated previously. So, the proposed scheme provides non-repudiation of signature delegation. The original signer doesn’t obtain the proxy signer’s private \( x_p \) and the proxy signer doesn’t obtain the original signer’s private key \( x_i \), neither the original signer nor the proxy signer can sign in place of the other party.

7. **Public Verification**: If the proxy signer denies the signature of the message \( m \), the receiver can prove the dishonesty of the signer by passing \((C, V, S)\) to the any third party who can check the signature validity by computing \( k' = H((y_{pr})^{(S|V)} \mod p) \mod q) \) then verify that the received signature is valid by computing \( V' = H(C, k') \) and accepts if:
\[
V' = V.
\]
So, in the proposed scheme anyone can verify the signature without the need to any secret values. Thus, it achieves public verifiability.

**4.4 Performance Analysis**

The security of the proposed scheme is based on the DLP and DHP. Its performance is better than another DLP-based scheme introduced in [20]. This comparison is provided in Table 2 and Figure 1. Table 1 shows the abbreviations that will be used in the comparison.

### Table 1. Time abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_e )</td>
<td>Time for performing a modular exponentiation computation.</td>
</tr>
<tr>
<td>( T_{\text{mult}} )</td>
<td>Time required for executing a modular multiplication in a finite field.</td>
</tr>
<tr>
<td>( T_h )</td>
<td>Time required for executing one-way hash function.</td>
</tr>
</tbody>
</table>

### Table 2. The comparison of the proposed proxy signcryption with the scheme in [20]

<table>
<thead>
<tr>
<th>Phase</th>
<th>Scheme in [20]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy Designation</td>
<td>( 3T_e + 2T_h + 3T_{\text{mult}} )</td>
<td>( 3T_e + 2T_h + 1T_{\text{mult}} )</td>
</tr>
<tr>
<td>Signcryption</td>
<td>( 1T_e + 1T_{\text{h1}} + 1T_{\text{mult}} + 1T_{\text{encr}} )</td>
<td>( 1T_e + 1T_{\text{mult}} + 1T_{\text{decr}} + 1T_{\text{encr}} )</td>
</tr>
<tr>
<td>Unsigncryption</td>
<td>( 3T_e + 2T_h + 3T_{\text{mult}} + 1T_{\text{decr}} )</td>
<td>( 1T_e + 2T_h + 1T_{\text{decr}} + 1T_{\text{mult}} )</td>
</tr>
<tr>
<td>Total No. of Operations</td>
<td>( 7T_e + 5T_{\text{h1}} + 7T_{\text{mult}} + 1T_{\text{encr}} + 1T_{\text{decr}} = 21 )</td>
<td>( 5T_e + 7T_h + 4T_{\text{mult}} + 1T_{\text{encr}} + 1T_{\text{decr}} = 18 )</td>
</tr>
</tbody>
</table>

**Fig 1: The comparison of the proposed proxy signcryption with the scheme in [20]**

To simplify the estimation of computational costs, we consider only the major operation. For example, the computational cost of modular multiplication, hash function, symmetric key encryption and decryption and hash function is ignored as compared with the expensive costs of modular exponentiation. So, the proposed scheme needs 5 modular exponentiations compared to 7 modular exponentiations of the scheme in [20].

Thus, the computational cost saving \( \frac{7 - 5}{7} = 28.5\% \). Another advantage of the proposed scheme over the scheme in [20] is that the proposed scheme achieves public verifiability while the scheme in [20] does not. The equation for recovering \( k \) does not involve the private key of the recipient or any private information. Consequently, the signature part of the signcrypted text can be verified by any third party.

While in the scheme in [20], the signature verification requires the knowledge of the receiver’s private key, so it can’t provide public verifiability.
5. The Elliptic Curve Variant of the Proposed Scheme

Since 1985 that elliptic curves applied independently by Miller [25] and Koblitz [26] to introduce a new public key cryptography, many researchers tried to employ it on different data types and improve the efficiency by proposing various techniques. In fact, the most attractive advantage of ECC that motivated cryptographers to use it was the greater security and more computationally efficient performance with equivalent key size in comparison with another public key as the RSA, ElGamal, etc. Shorter keys result in less storage requirements. The ECDLP is more difficult than the DLP.

In this version of the proposed scheme, the secret keys are chosen as random elements, where \( x \in Z_q^* \). The system-wide parameters include the elliptic curve \( E \), a point \( G \) on the elliptic curve with a prime order \( q \). The corresponding public keys are computed as \( Y = x . G \), where: \( (x, Y) \) is the original signer key pair \((x_1, Y_1)\) is the proxy signer key pair and \((x_2, Y_2)\) is the recipient key pair. The EC variant of the proposed proxy signcryption scheme is as follows.

5.1 The Proposed Scheme Construction

5.1.1 Setup

The system authority (SA) selects two large primes \( p \) and \( q \) where \( p \equiv 1 \pmod{q} \). An elliptic curve \( E \) is chosen with \( G \) is a generator point on the elliptic curve.

5.1.2 Key Generation

The original user \( U_i \) chooses his private key \( x_i \in Z_q^* \) and computes the public key \( Y_i = x_i . G \). The proxy \( U_p \) chooses his private key \( x_p \in Z_q^* \) and computes the public key as \( Y_p = x_p . G \). The recipient \( U_r \) chooses his private key \( x_r \in Z_q^* \) and computes the public key as \( Y_r = x_r . G \). Both the proxy \( U_p \) and recipient \( U_r \) use Diffie-Hellman protocol [only one time offline before the transmission of any message] to exchange a shared secret key \( Y_{sh} \) as follows:

The proxy \( U_p \) computes: \( Y_{ah} = x_p . Y_p = x_p . x_i . G \)

The recipient \( U_r \) computes: \( Y_{rh} = x_r . Y_r = x_r . x_p . G \)

5.1.3 Proxy Designation Protocol

The original user \( U_i \) delegates his signing power to a proxy signcryption \( U_p \) as follows: \( U_i \) first chooses a random integer \( d \in Z_q^* \) and computes

\[
T = d . G = (\alpha, \beta) = (T_r, T_u)
\]

\[
\sigma \equiv \left( \frac{d}{x_i} - H(m_w, \alpha) \right) \mod q
\]

The original user \( U_i \) sends \((\sigma, m_w, \alpha)\) to \( U_p \). Upon receiving \((\sigma, m_w, \alpha)\), \( U_p \) computes:

\[
Y_i . (\sigma + H(m_w, \alpha) \text{ and performs check its validity as follows:} \]

\[
(T_r, T_u) = Y_i . (\sigma + H(m_w, \alpha)) \text{. Checks if } T_u = \alpha \text{ if not equal to the right-hand side, the proxy requests a new } (\sigma, m_w, \alpha) \text{ to be sent again. The verification of the above equation proceeds as follows:}
\]

\[
Y_i . (\sigma + H(m_w, \alpha) = x_i . G . \left( \frac{d}{x_i} - H(m_w, \alpha) + H(m_w, \alpha) \right) = G . d = T
\]

After the proxy authenticates the original signer, the proxy computes the secret key as follows: \( x_{pr} \equiv (\sigma + x_p) \mod q \) as his proxy signature, secret key. Then he computes and publishes the corresponding proxy public key:

\[
Y_{pr} = x_{pr} . G = \sigma + x_p . G = \left( \frac{d}{x_i} - H(m_w, \alpha) + x_p \right) . G
\]

5.1.4 Proxy Signcryption

The proxy will do the following steps to sign and encrypt the message \( m \). The proxy chooses a random number \( w \in Z_q \) and computes:

1. \( K = w . G = (x_t, y_t) \)
2. \( Z = Y_{ah} + K = (x_2, y_2) \)
3. \( C = E_{x_i} (m) \)
4. \( V = H(C, x_1) \)
5. \( S \equiv \left( \frac{w}{x_{pr}} - V \right) \mod q \)

The proxy sends \( \delta = (C, V, S) \) to the receiver.

5.1.5 Proxy Unsigncryption and Verification

The receiver decrypts the message and checks the signature validity as follows:

1. Recover the key \( K \) by computing:
\[
K = Y_{pr} \cdot (S + V) = (x_t, y_t)
\]
2. Verify that the received signature is valid by computing:
\[
V' = H(C, x_1) \text{ and accepts if: } V = V'
\]
3. Compute \( Z = Y_{ah} + K = (x_2, y_2) \)
   where \( Y_{sh} \) is the shared secret key between the proxy and the receiver.
4. Decrypt the ciphertext \( m = D_{x_2}(C) \).

5.2 Proof of Correctness

The following equation demonstrates the correctness of the proposed scheme:

\[
K = Y_{pr} \cdot (S + V) = x_{pr} . G \cdot \left( \frac{w}{x_{pr}} - V + V \right) = G . w = (x_t, y_t)
\]

5.3 Performance Analysis

The performance of the proposed proxy signcryption scheme based on the ECDLP is analyzed and compared to the scheme in [20]. It is found that the proposed scheme involves fewer computations than the scheme in [20]. Table 3 defines the notation that is used in comparison. Table 4 and Figure 2 shows the comparison of the proposed scheme based on the ECDLP to that of [20].

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{EC-mult} )</td>
<td>Time required for executing multiplication operation on elliptic curve ( E )</td>
</tr>
<tr>
<td>( T_{EC-add} )</td>
<td>Time required for executing addition operation on elliptic curve ( E )</td>
</tr>
<tr>
<td>( T_{mul} )</td>
<td>Time required for executing a modular multiplication in a finite field</td>
</tr>
<tr>
<td>( T_h )</td>
<td>Time required for executing one-way hash function</td>
</tr>
<tr>
<td>( T_{enc}/T_{dec} )</td>
<td>Time required for executing an encryption/decryption operation</td>
</tr>
</tbody>
</table>
Table 4. The proposed signcryption scheme compared with the scheme in [20]

<table>
<thead>
<tr>
<th>Phase</th>
<th>The Scheme in [20]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy Designation</td>
<td>$3T_{EC\text{-mult}} + 1T_{EC\text{-add}} + 2T_{h} + 2T_{mult}$</td>
<td>$3T_{EC\text{-mult}} + 2T_{h} + 2T_{mult}$</td>
</tr>
<tr>
<td>Signcryption</td>
<td>$1T_{EC\text{-mult}} + 1T_{EC\text{-add}} + 1T_{h} + 1T_{mult} + 1T_{encr}$</td>
<td>$1T_{EC\text{-mult}} + 1T_{EC\text{-add}} + 1T_{h} + 1T_{mult} + 1T_{encr}$</td>
</tr>
<tr>
<td>Unsingcryption</td>
<td>$3T_{EC\text{-mult}} + 3T_{EC\text{-add}} + 3T_{h} + 3T_{mult}$</td>
<td>$3T_{EC\text{-mult}} + 3T_{h} + 3T_{mult}$</td>
</tr>
<tr>
<td>Total No. of Operations</td>
<td>$7T_{EC\text{-mult}} + 4T_{EC\text{-add}} + 4T_{h} + 4T_{mult} + 4T_{encr} + 1T_{dec}$</td>
<td>$5T_{EC\text{-mult}} + 2T_{EC\text{-add}} + 2T_{h} + 2T_{mult} + 2T_{encr} + 1T_{dec}$</td>
</tr>
</tbody>
</table>

Fig 2: The comparison of the proposed EC variant with the scheme in [20]

To simplify the estimation of computational costs, we consider only the major operation which is the multiplication operation on elliptic curve $E$. The proposed scheme needs 5 modular exponentiations compared to 7 modular exponentiations of the scheme in [20]. Thus, the computational cost saving $7 - 5 = 28.5\%$.

6. NUMERICAL EXAMPLE

Here is a numerical example as a proof-of-concept, which has been implemented using Mathematica 10.2 program. In this example, the parameters used are among the 256-bit recommended domain parameters for elliptic curves.

- $p$ is the prime specifying the base field.
- $a$ and $b$ are coefficients of the equation; $y^2 = (x^3 + ax + b)$ mod $p$ defining the elliptic curve.
- $\sigma = (x, y)$ is the base point, i.e. a point in $E$ of prime order, with $x$ and $y$ being its $x$-and $y$-coordinate, respectively.
- $q$ is the prime order of the group generated by $G$.

6.1 Setup

$p = 7684956397045344220809746629001649093037950209943$
$a = 155203735601445031516197751$
$b = 5669818760532611004362722839617784607712061453947$
$b = 2141093868218876388413993$
$q = 175773249732138841075607789794520262950426058923$
$q = 08456704685230063325438902$
$q = 2592357855591906306533639079$

6.2 Key Generation

$x_i = (\text{original signer secret key}) = 8665959940333993592163116089274553897378463889$
$x_r = (\text{receiver secret key}) = 973410554442261607260695179907$
$m = (\text{the message}) is the word “cryptography”

6.3 Proxy Key Generation

$d = 206669979766610830382768754247885030508940$
$m = 0100000398081655008090327617096$

\[T = \{(116039191616090634322713514605716968075070369, 0), (116039191616090634322713514605716968075070369, 1)\}\]
Y_p = \{13041773698909967771619958039067633922074615
920127266461559756095977587635876, 7141585815
446878481945340818835758425154555996903261955
609906145144359426222\}

6.4 Signcryption Generation

w = 88990075432637789997765544324567887766654909
876;
K = w . G = \{(x_1, y_1)\}
64162219985773727294867760547057607657495075
68251454626793297696529785182744, 501676066442
448277394769285182928629584394033706869005929
37316930438700797917].
Z = Y_\delta + K = \{(x_2, y_2)\}
8683133498161910258312183094363452227251778288
52995973178495134662680987160220, 100578135546
667461611237361227386978065064495112101180542
126711362544280738846].
C = E_{x_1}(m) =
8683133498161910258312183094363452227251778288
530061254668248158686492808268453.
V = H(C, x_1) = 28666057124587368600392974107572652702.
S = \{w \over m_{pr} - V \} \mod q =
683197827426845195709681071797691902394651462
87528156492117057389313175962207.

6.5 Unsigncryption and Verification

Y_{pr} = (S + V) \cdot (x_1, y_1) =
\{64162219985737727294867760547057607657495075
68251454626793297696529785182744, 501676066442
448277394769285182928629584394033706869005929
37316930438700797917\}
V' = H(C, x_1) =
28666057124587368600392974107572652702 = V

The receiver accepts the received ciphertext.

Z = Y_\delta + K = \{(x_2, y_2)\} =
8683133498161910258312183094363452227251778288
52995973178495134662680987160220, 100578135546
667461611237361227386978065064495112101180542
126711362544280738846].
C = D_{x_2}(C) = \text{crypto}\text{graphy}

7. CONCLUSIONS

As people in modern societies are busier than any human era and computer network has profound impact on how people work and live through fast and convenient information exchange, people need more help from each other to accomplish more work via network connections in limited period of time. Therefore, privilege delegation mechanisms has become a necessary service in modern enterprises and organizations. Proxy signcryption scheme provides a secure privilege delegation mechanism for a person to delegate his privilege to his proxy agent to accomplish things. This paper introduces a proxy signcryption scheme in which the original signer delegates his signing rights to a proxy agent. This scheme has many applications such as in e-cash systems. The security properties of the proposed scheme are investigated revealing that it meets all security requirements. The use of a warrant facilities identifying both the original as well as

proxy agents. Additionally, warrants are used to specify the signing capacity of the proxy agent to prevent misuse of the delegation. Moreover, the proxy signature is distinguishable from the original signer to protect a malicious proxy agent. The proposed scheme as well as its elliptic-curve based variant is compared with the scheme in [20] and it is found that the proposed scheme reduces the computational burden so that it is more efficient to support the class of applications targeted by it such as e-commerce using mobile computing and communication devices that require large number of individual short messages for the completion of a transaction. These include banking services stock trading international roaming information for GSM etc.

8. REFERENCES


