MPKC-based Threshold Proxy Signcryption Scheme
Li Huixian¹, Gao Jin¹, Wang Lingyun¹, and Pang Liaojun²
¹School of Computer Science and Engineering, Northwestern Polytechnical University, China
²State Key Laboratory of Integrated Services Networks, Xidian University, China

Abstract: The threshold proxy signcryption can implement signature and encryption simultaneously in one logical step, and can be used to realize the decentralized protection of the group signature key, so it is an efficient technology for network security. Currently, most of the existing threshold proxy signcryption schemes are designed based on the traditional public key cryptosystems, and their security mainly depends on the difficulty of the large integer decomposition and the discrete logarithm. However, the traditional public key cryptosystems cannot resist the quantum computer attack, which makes the existing threshold proxy signcryption schemes based on traditional public key cryptosystems insecure against quantum attacks. Motivated by these concerns, we proposed a threshold proxy signcryption scheme based on Multivariate Public Key Cryptosystem (MPKC) which is one of the quantum attack-resistant public key algorithms. Under the premise of satisfying the threshold signcryption requirements of the threshold proxy, our scheme can not only realize the flexible participation of the proxy signcipers but also resist the quantum computing attack. Finally, based on the assumption of Multivariate Quadratic (MQ) problem and Isomorphism Polynomial (IP) problem, the proof of the confidentiality and the unforgeability of the proposed scheme under the random oracle model is given.

Keywords: Multivariate public key cryptosystem, signcryption, threshold proxy signcryption, quantum attack.

Received December 5, 2017; accepted May 29, 2019
https://doi.org/10.34028/iajit/17/2/7

1. Introduction
Signcryption, originally proposed by Zheng [20], can achieve digital signature and encryption in a single logical step. It’s more efficient than that of sign-then-encrypt approaches. Lots of signcryption schemes have been designed since the concept of signcryption was proposed. But these schemes involve expensive bilinear pairing operations, which are a bottleneck for resource-constrained devices. In order to solve this problem, Gamage and Leiwo [2] proposed the proxy signcryption by combining the proxy signature with the signcryption. However, for most of the existing proxy signcryption schemes, there is only one signcyper, which results in the right abuse of the proxy signcyper. To prevent this problem, Chan and Wei [1] adopted the idea of threshold and proposed a threshold proxy signcryption scheme. But most of the existing threshold proxy signcryption schemes are designed based on traditional public key cryptosystems whose security mainly depends on large integer factoring problem and discrete logarithm problem. The security of these hard problems will be broken by quantum computers and quantum algorithm in the Quantum age [12], so it is urgent to propose new threshold proxy signcryption schemes that can resist quantum attack.

1.1. Literature Review
The signcryption technology has broad application prospects, so it has been studied extensively [4, 6, 7, 11, 14]. Since Gamage and Leiwo [2] suggested transferring the high-cost computation of cryptographic from the cryptosystem to servers with great computational abilities, lots of proxy signcryption schemes have been proposed subsequently. In 2010, Lin et al. [8] proposed a proxy signcryption scheme based on bilinear pairings under the security of chosen ciphertext attack and chosen message attack. However, Pan et al. [10] proved that Lin et al. [8] scheme does not satisfy the indistinguishability under adaptively chosen ciphertext attack and unforgeability under chosen message attack. In 2012, Swapna et al. [13] proposed an identity-based proxy signcryption scheme with forward security and public verifiability. Unfortunately, Yeh [19] found that Swapna’s scheme is vulnerable to the proxy certificate forgery attack and further gave an improved scheme against this attack in 2014. In 2015, Xue et al. [17] combined the location protocol with the proxy signcryption scheme and proposed a proxy signcryption model based on locations, which is applicable for mobile network and extends the application scenarios of proxy signcryption.

In the above proxy signcryption schemes, there is only one signcyper, which results in the possible right abuse of the proxy signcyper. To solve this problem, Chan and Wei [1] proposed the threshold proxy signcryption scheme. Threshold proxy signcryption schemes can achieve decentralized protection of group signcryption keys and decentralize the right of members to prevent the abuse of right. Wang and Liu [15]
indicated that threshold proxy signature schemes cannot provide confidentiality, so they extended the threshold proxy signature scheme and constructed an identity-based threshold proxy signcryption scheme in 2005. Yang and Yu [18] noticed that Wang et al.’s scheme cannot resist collusion attacks, and gave a new identity-based threshold proxy signcryption scheme. Li et al. [5] proposed a new identity-based threshold proxy signcryption scheme from bilinear pairings. This scheme is forward secure and can prevent public key replacement attack and Key Generation Centre (KGC) attack effectively. By applying the concepts of double replacement attack and Key Generation Centre, Zhou and Yu [21] proposed a new double-threshold proxy signcryption scheme from bilinear pairings which is suitable for many applications in practice.

1.2. Our Contributions

Despite many innovations, the above proxy signcryption schemes are designed based on the traditional public key cryptosystems, and their security mainly depends on the difficulty of large integer decomposition and the discrete logarithm. It is known that the traditional public key (PK) cryptosystems cannot resist the quantum computer attack [3, 12] which makes the existing proxy signcryption schemes based on traditional public key cryptosystems insecure against quantum attacks.

Aimed at this problem, we proposed a threshold proxy signcryption scheme based on Multivariate Public Key Cryptosystem (MPKC) [9] which is a good candidate of public key cryptosystem which can resist quantum attack. Our scheme not only can satisfy requirements of threshold proxy signcryption, but also make the proxy signcryption join flexibly. The most important thing is that our scheme can resist the attack of quantum computer. For convenience, we use MPKC-based Threshold Proxy Signcryption Scheme (MTTPSC) for the short name of the proposed scheme.

1.3. Organizations

The rest of the paper is organized as follows. Section 2 contains preliminaries about some hard problems and general form of MPKC. In section 3, we give framework and security model of our scheme. In section 4 we describe the MTTPSC scheme in detail. In section 5 we give detailed correctness analysis and security proof of our scheme. Section 6 is the performance analysis of our scheme and comparison with some previous works. We conclude our paper with some suggestions for future work in section 7.

2. Preliminaries

Suppose \( GF(p) \) is a finite field with prime order \( p \), \( n \) is a positive integer. \( x_1, \ldots, x_n \in GF(p) \) are \( n \) variables over a finite field. The multivariate quadratic polynomial equation consisting of these \( n \) variables is:

\[
 f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i + c
\]

where \( a_{ij}, b_i, c \in GF(p) \). On the finite field \( GF(p) \), the system of equations consisting of \( m \) equations for the variable tuple \( x=(x_1, \ldots, x_n) \) has the following form:

\[
 \begin{align*}
 f^{(1)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^{(1)} x_i x_j + \sum_{i=1}^{n} b_i^{(1)} x_i + c^{(1)} \\
 f^{(2)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^{(2)} x_i x_j + \sum_{i=1}^{n} b_i^{(2)} x_i + c^{(2)} \\
 f^{(m)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^{(m)} x_i x_j + \sum_{i=1}^{n} b_i^{(m)} x_i + c^{(m)}
 \end{align*}
\]

2.1. Multivariate Quadratic (MQ) Problem

Given a group of equations over a finite field, as shown above, it is required to find a set of variables \( \tilde{x} \in GF(p)^n \), \( \tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \), such that operation results of all the equations are all 0, that is, \( f^{(1)}(\tilde{x}) = \cdots = f^{(m)}(\tilde{x}) = 0 \). Finding \( \tilde{x} \) that meets the above requirements is called an MQ problem.

2.2. Isomorphism Polynomial (IP) Problem

Given \( F \) and \( \bar{F} \) be two public sets of \( n \) quadratic with \( n \) variables over \( (p) \), if \( F \) and \( \bar{F} \) are isomorphic, then \( \bar{F} = T \circ F \circ S \) ( \( \circ \) denotes composition of mappings), where \( T \) and \( S \) are two reversible affine transformations \( \text{on} GF(p)^n \rightarrow GF(p)^n \). Finding \( (T, S) \) from \( F \) to \( \bar{F} \) such that \( F = T \circ F \circ S \) is called the IP problem.

2.3. The General form of MPKC

In a primitive multivariate public key cryptosystem, one user’s public key is \( p = T \circ F \circ S \) and the corresponding secret key is \( (T, F, S) \). For a given message \( x=(x_1, \ldots, x_n) \), if the sender wants to send it to the receiver confidentially, he should calculate the ciphertext \( y=(y_1, \ldots, y_n) \) by using the receiver’s public key \( P \), that is, \( y=P(x_1, \ldots, x_n) \). In order to decrypt the ciphertext \( y \) by using his secret key, the receiver can calculate \( (z_1, \ldots, z_n)=T^{-1}(y_1, \ldots, y_n) \), \( (d_1, \ldots, d_n)=F^{-1}(z_1, \ldots, z_n) \), and \( (x_1, \ldots, x_n)=S^{-1}(d_1, \ldots, d_n) \). At last, the receiver gets the plaintext \( x=(x_1, \ldots, x_n) \).

3. Threshold Proxy Signcryption Model Based on MPKC

3.1. Framework of MTTPSC

In a threshold proxy signcryption scheme, there is an original signcryption denoted by \( id_s \). A group of proxy signcrypters that are appointed by the original signcryption is denoted by \( \text{pr} = \{ id_1, id_2, \ldots, id_m \} \) (\( m \) is the number of proxy signcrypters). The receiver is denoted by \( id_r \). Our scheme consists of the following five algorithms:
Setup (1^3): system initialization algorithm, which is executed by the KGC. Given a system security parameter 1^4, KGC selects the necessary parameter params for the system.

Extract (id, params, ε): this algorithm includes user’s partial public/private key generation and user’s final public/private key generation.

Proxy Key Generation and Authorization: this algorithm is executed by the original signcrypter id, and mainly generates proxy signcryption keys for proxy signcryption and authentication information which is used to show identity of the original signcrypter and proxy signcryption and state message types of proxy signcryption and necessary authorization.

Threshold Proxy Signcrypt: inputting related params and a message m, the proxy signcryption will generate the signcryption ciphertext as legal representatives of the original signcrypter.

De-signcrypt: this algorithm is executed by the receiver id, the recipient can get the message m if the verification is passed.

3.2. Security Model of MTPSC

Wang et al. [16] first proposed the security model of multi-proxy signature schemes. Based on this model, we will give the security model of MTPSC in this section, which mainly consists of message confidentiality security and unforgeability security. During the game of confidentiality and unforgeability, there are several random oracles to be queried by the attacker listed below:

• Partial Key Extract Query: enter the user identity, the random oracle calls the Extract algorithm to generate the corresponding user private key PPS and returns.

• Secret Key Extract Query: a private key extraction query is performed on the user identity, and the random oracle returns the full private key PS of the user.

• Public Key Extract Query: entering the user identity, and the random oracle returns the public key PK associated with the user’s id.

• Replace Public Key Query: the user identity id and a valid PK are entered, and the random oracle replaces the user's with PK'.

• Proxy Authorization Extract Query: entering the original signcrypter’s identity id and the proxy signcryption’s identities Pr={id_1, id_2, ... , id_num}, The random oracle will run the Proxy Key Generation and Authorization algorithm to get the corresponding proxy key and authorization certificate m_

• Threshold Proxy Signcrypt Query: entering the message m, the original signer identity id, and the proxy signcryption Pr={id_1, id_2, ... , id_num}, the random oracle runs the Threshold Proxy Signcrypt algorithm and generates a signcryption ciphertext.

• ThresholdDe-signcrypt Query: Entering the signcryption ciphertext σ, the original signcrypter identity id and Pr={id_1, id_2, ... , id_num} and the recipient identity id, the random oracle will run the De-signcrypt algorithm and returns the plaintext.

Here we give the detail definition of message confidentiality and message unforgeability of our scheme as follows.

• Definition 1. IND-MTPSC-CCA

(Indistinguishability of ciphertexts under adaptive chosen-ciphertext attack of Threshold Proxy Signcryption scheme based on Multivariate Public Key Cryptosystem).

The message confidentiality of our scheme satisfies the indistinguishability of ciphertexts under adaptive chosen-ciphertext attack. Suppose that A is an attacker and II denotes our MTPSC scheme. The following interactive games are defined between A and a challenger C.

• Setup: C executes this algorithm to generate system master key and system parameter params. System master key will be saved secretly and params will be sent to attacker A. Attacker A generates target identity id^*.

• Phase 1: A issues the following queries to C and C executes random oracles as listed above.

• Secret Key Extract Query: When C receives secret key extract query about identity id (id_i ≠ id^*), it runs key extraction algorithm and obtains the corresponding private key S_{id_i}.

• Threshold Proxy Signcrypt Query: when C receives threshold proxy signcryption query (m, id, ido, Pr), it calculates proxy signcryption ciphertext σ = Threshold_proxy_signcrypt (m, id, ido, Pr) and sends it to A.

• Threshold De-signcrypt Query: when C receives de-signcryption query (σ, id), it will calculate the m=De-signcryption (m, σ) and will be returned to A, otherwise C will output ⊥.

• Challenge: A randomly selects two different messages (m_0, m_1) with the same bit length and the original signcryption’s identity id, a list of proxy signcryption Pr and the receiver’s identity id^*. C randomly selects a message m_b (b ∈ {0,1}) calculates corresponding private keys of these users, and threshold proxy signcryption ciphertext σ = Threshold_proxy_signcrypt (m_b, id^*, ido, Pr). Then C sends σ to A.

• Phase 2: just as the query process in Phase 1, attacker A performs a lot of queries to random oracle. It should be noticed that the attacker cannot issue queries to target identity id^*'s private key in this phase and cannot query σ either.
• **Guess:** the attacker A outputs a guess value $b' \in \{0,1\}$. If $b' = b$, attacker A will win the game.

The advantage of attacker A in this game is

$Adv_{\Pi}^{IND-MTPSC-CCA}(A) = \Pr[b' = b] - \frac{1}{2}$.

If the probability of guessing $b'$ correctly in this game is less than $\varepsilon$ for any IND-MTPSC-CCA attacker A in the polynomial time $t'$, we call this scheme is $(t', \varepsilon)$-IND-MTPSC-CCA secure.

• **Definition 2.** UF-MTPSC-CMA-CWA (Existential unforgeability against chosen message attacks and chosen warrant attacks of Threshold Proxy Signcryption scheme based on Multivariate Public Key Cryptosystem)

The unforgeability of our scheme MTPSC is against chosen-message and chosen-authorization attacks. Let $\Pi$ be MTPSC. Assume the following games are played between a forger $F$ and a challenger $C$.

• **Setup:** $C$ runs this algorithm to generate system parameter $params$ and master key. It will send $params$ to forger $F$ and $F$ will output a target identity $id_t$.

• **Attack:** attacker can access random oracle listed above. It should be noticed that the forger cannot carry out key extraction query and replace public key query for the target identity.

• **Forgery:** $F$ at last achieves the following forgery goal.

Attacker outputs assigncryption ciphertext $\sigma$ of message $m$ such that the ciphertext can be successfully deciphered by the receiver. For the generation of signcryption ciphertext, the target user $id_t$ is one of proxy signcryption key. It is also required that $id_t$ is not authorized by the original signcryption and $id_t$ will not issue threshold proxy signcryption queries to the message $m$. This type of attack is called chosen-message attack.

Attacker output assigncryption ciphertext $\sigma$ about message $m$ such that the ciphertext can be successfully deciphered by the receiver. For the generation of signcryption ciphertext, the target user $id_t$ is one of proxy signcryption key and proxy signcryption key is not authorized by the original signcryption. This type of attack is called chosen-authorization attack.

The advantage of forger $F$ is defined as

$Adv_{\Pi}^{UF-MTPSC-CMA-CWA}$ which is the probability of winning the above game. If the advantage of attacker’s winning the above game is negligible, we call the scheme unforgeable.

### 4.1. Setup Algorithm

Taking as input a secure parameter $1^k$, KGC selects some appropriate parameters according to the secure parameter and generate other system parameters. Firstly, KGC selects a finite field $GF(p^k)$ whose generator is $p$ and the order is $q (q = p^k, k$ is a positive integer). Select secure hash functions $H_0: \{0,1\}^{n+1} \times \{0,1\}^* \rightarrow GF(p)^m$ where $m$ is the length of the identity. $H_1: \{0,1\}^* \rightarrow GF(p)^m, H_2: \{0,1\}^* \rightarrow GF(p)^m$, where $m$ is the bit-length of the message. Select an invertible multivariate quadratic polynomials $F: GF(p) \rightarrow GF(p)$ and two invertible affine transformations $T: GF(p) \rightarrow GF(p), S: GF(p) \rightarrow GF(p)$, then compute $F = T \circ F \circ S$.

The system public key of KGC is $(F, F)$ and system private key is $(T, S)$. KGC publishes $params =< G, p, q, H_0, H_1, H_2, F >$ and secretly keeps system private key $(T, S)$.

### 4.2. Extract Algorithm

Key extraction consists of the following two algorithms. Given user’s identity id, KGC firstly generates user’s partial private key which the user later employs to calculate his final public key and private key.

#### 4.2.1. Generation of User’s Partial Public Key and Private Key

Given a user’s identity id, KGC selects two invertible affine transformations $T_{id_p}'$ and $S_{id_p}'$ for this user and calculates $F_{id_p} = T_{id_p}' \circ F \circ S_{id_p}'$. The partial public key is $F_{id_p}$ and the partial private key is $(T_{id_p} = T_{id_p}' \circ T', S_{id_p} = S \circ S_{id_p}')$. KGC sends the user’s partial private key to the user via a secure channel.

#### 4.2.2. Generation of User’s Final Public Key and Private Key

After receiving the partial private key from KGC, the user $id$ randomly selects two invertible affine transformations $(T_{id}' S_{id}')$ in finite field and computes $F_{id} = T_{id}' \circ F_{id} \circ S_{id}'$. The public key of the user id is $F_{id}$ and the private key is $(T_{id} = T_{id}' \circ T_{id}' \circ T, S_{id} = S \circ S_{id}' \circ S_{id}')$. Publish the public key while keeping the private key secret. In this case, KGC does not know the user’s private key. The user sends the public key to the KGC who will publish it.

### 4.3. Proxy Key Generation and Authorization Algorithm

In order to realize proxy signcryption, Alice needs to authorize a group of $num$ proxy signcryption, denoted as $Pr=\{id_1, id_2, ..., id_{num}\}$ to signcrypt on behalf of her,
so Alice executes the following steps to complete proxy signcryption authorization, and generate authorization certificate \( m_w \) and proxy signcryption key.

### 4.3.1. Generation of Proxy Signcryption Key

The original signcrypted Alice generates its proxy signcryption key and shares it with \( num \) proxy signcrypteders. Alice’s public key is \( F_{\text{Alice}} \) and his private key is \((T_{\text{Alice}}, S_{\text{Alice}})\. Alice selects two invertible affine transformations \((T'_p, S'_p)\) infinite field, and computes \( T_p = T'_p \circ T_{\text{Alice}} \) and \( S_p = S'_p \circ S_{\text{Alice}} \). Then the private key of the proxy signcryption is \((T_p, S_p)\). The corresponding public key is

\[
F_p = T'_p \circ F_{\text{Alice}} \circ S'_p \circ S_{\text{Alice}} \circ S_p = T_p \circ F_{\text{Alice}} \circ S_p
\]

The inverse operation of the public key of proxy signcryption is \( F_p^{-1} = S_p^{-1} \circ F_{\text{Alice}}^{-1} \circ T_p^{-1} \). Then Alice will delegate the proxy key to the \( num \) proxy signcrypteders. Both \( T_p \) and \( S_p \) are invertible affine transformations over \((GF(p))^n \rightarrow GF(p)^n\), and the corresponding inverse \( T_p^{-1} \) and \( S_p^{-1} \) are also invertible affine transformations over \((GF(p))^n \rightarrow GF(p)^n\). Here \( T_p^{-1} = ((T_p^{-1})^{(1)})^T, ((T_p^{-1})^{(2)})^T, ..., ((T_p^{-1})^{(n)})^T, \) Where \( (T_p^{-1})^{(i)} \) is the \( i \)th element of \( T_p^{-1} \).

### 4.3.2. Secret Sharing

Alice selects \( num \) different numbers \( \beta_i \in GF(p)^n \) in finite field \((GF(p))^n\) and uses them to construct a Vandermonde Matrix:

\[
A = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
\beta_1 & \beta_2 & \cdots & \beta_num \\
\vdots & \vdots & \ddots & \vdots \\
\beta_i^{-1} & \beta_{i+1}^{-1} & \cdots & \beta_{num}^{-1}
\end{pmatrix}
\]

Then we calculate as follows.

\[
\begin{align*}
B_1 &= T_p^{-1} \times A \\
&= \begin{pmatrix}
(T_p^{-1})^{(1)} \\
(T_p^{-1})^{(2)} \\
\vdots \\
(T_p^{-1})^{(n)}
\end{pmatrix}
\begin{pmatrix}
1 & 1 & \cdots & 1 \\
\beta_1 & \beta_2 & \cdots & \beta_num \\
\vdots & \vdots & \ddots & \vdots \\
\beta_i^{-1} & \beta_{i+1}^{-1} & \cdots & \beta_{num}^{-1}
\end{pmatrix} \\
&= \begin{pmatrix}
b_{1,1} & b_{1,2} & \cdots & b_{1,\text{num}} \\
b_{2,1} & b_{2,2} & \cdots & b_{2,\text{num}} \\
\vdots & \vdots & \ddots & \vdots \\
b_{\text{num},1} & b_{\text{num},2} & \cdots & b_{\text{num},\text{num}}
\end{pmatrix} \\
&= (B_1[1], B_1[2], ..., B_1[\text{num}])
\end{align*}
\]

\[
B_2 = S_p^{-1} \times A = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
\beta_1 & \beta_2 & \cdots & \beta_num \\
\vdots & \vdots & \ddots & \vdots \\
\beta_i^{-1} & \beta_{i+1}^{-1} & \cdots & \beta_{num}^{-1}
\end{pmatrix} \times \begin{pmatrix}
b_{1,1} & b_{1,2} & \cdots & b_{1,\text{num}} \\
b_{2,1} & b_{2,2} & \cdots & b_{2,\text{num}} \\
\vdots & \vdots & \ddots & \vdots \\
b_{\text{num},1} & b_{\text{num},2} & \cdots & b_{\text{num},\text{num}}
\end{pmatrix}
\]

\[
= (B_2[1], B_2[2], ..., B_2[\text{num}])
\]

In the above formulas \((B_1[i], B_2[i])_{i=1,2,...,\text{num}}\) represents the \( i \)th column vector. Thus, the original signcrypted Alice has split the proxy key into \( num \) parts to share. Alice makes \( B_{id_i} = (B_1[i], B_2[i]) \) as the private key of proxy signcrypted \( id_i \) and sends it to the proxy signcrypted via a secure channel.

### 4.3.3. Proxy Signcryption Authorization

Alice sends \((B_{id_i}, \beta_i)\) to each proxy signcrypted \( id_i \) where \((i=1,2,...,\text{num})\) by a secure secret channel. At the same time Alice generates an authorization \( m_w \) which consists of necessary information about the authorization which includes the identity of the original signcrypted \( id_{\text{Alice}}, \) members of proxy signcryption group \( Pr=\{id_1, id_2, ..., id_{\text{num}}\}, \) appointed legal proxy signcryption expiry date \( t, \) public keys of proxy signcryption and some necessary information, that is \( m_w =\{id_{\text{Alice}}, id_1, ..., id_{\text{num}}, F_p\} \). Then Alice signs the authorization by using a hash function \( H_0 \) and computes \( \text{War} \) under the inverse of Alice’s public key \( F_{\text{Alice}}^{-1}, \text{War} = F_{\text{Alice}}^{-1}(H_0(m_w)) \), Alice publishes the authorization \((m_w, \text{War})\).

### 4.4. Threshold Proxy Signcryption Algorithm

Without loss of generality, we assume that there are at least \( n \) proxy signcrypteders who cooperate to signcrypt the message \( m \) for Alice in the threshold proxy signcryption scheme.

#### 4.4.1. Identity Authentication of Proxy Signcrypteders

First of all, each proxy broadcasts its identity to all the other \( n-1 \) proxy signcrypteders. Through a query of public authorization \( m_w \), each proxy signcrypted makes sure whether other proxy signcrypteders’ identities are valid in the same group. Then, each proxy signcrypted verifies the equation \( F_{wuc}(\text{War}) = H_0(m_w) \). If this equation does not hold, the generation of proxy signcryption ciphertext is rejected, otherwise, continue to do the next step. If the identities of all other proxies are legal, the next step will be done, otherwise, abort the algorithm. Through the above operations, each member can get a list of identity \( \text{list} = \{id_{j_1}, id_{j_2}, ..., id_{j_n}\} \), where \( 1 \leq j_1 < j_2 < \cdots \leq j_n \leq \text{num} \).

#### 4.4.2. Secret Sharing and The Proxy Signcryption Ciphertext Generation

According to the order in the identity list, the first proxy signcrypted of \( n \) proxy signcrypteders randomly selects a value \( r = GF(p)^n \), then computes \( R = F(r) \) and \( Y = H_1(m \parallel R \parallel id_{\text{Alice}} \parallel \text{list}) \).

1. Each proxy signcrypted’s private key is \( B_{id_{j_i}} = (B_1[j_i], B_2[j_i])_{i=1,2,...,\text{num}} \). According to the order in the identity list, each proxy signcrypted \( id_{j_i} \) sends the
\( k \)th element \((B1[j], B2[j], \ldots, B1[j])\) of its private key and \( \beta_{ji} \) to the user \( id_{ji} \). Then, each proxy signcyrpeter \( id_{ji} \) performs the following computations:

\[
(B1[j], B1[j], \ldots, B1[j]) \times \begin{pmatrix}
1 & 1 & \cdots & 1 \\
\beta_{ji} & \beta_{ji} & \cdots & \beta_{ji} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{ji}^{n-1} & \beta_{ji}^{n-1} & \cdots & \beta_{ji}^{n-1}
\end{pmatrix}^{-1}
\]

\[
= (PV1_{ji}, PV1_{ji}, \ldots, PV1_{ji}) = PV1_{ji}
\]

\[
(B2[j], B2[j], \ldots, B2[j]) \times \begin{pmatrix}
1 & 1 & \cdots & 1 \\
\beta_{ji} & \beta_{ji} & \cdots & \beta_{ji} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{ji}^{n-1} & \beta_{ji}^{n-1} & \cdots & \beta_{ji}^{n-1}
\end{pmatrix}^{-1}
\]

\[
= (PV2_{ji}, PV2_{ji}, \ldots, PV2_{ji}) = PV2_{ji}
\]

2. Each proxy signcyrpeter \( id_{ji} \) computes \( y = PV1_{ji}(Y) \) and sends \( y \) to a trusted third party. Receiving \( y \) from \( n \) proxy signcyrpertes, \( KGC \) calculates \( Y' = (y_1, \ldots, y_n) \) and \( Y'' = F_{Alice}(Y') \) by using the partial private key of the original signcyrpeter. Then, \( KGC \) broadcasts \( Y'^{\prime} = n \) proxy signcyrpertes.

3. After receiving \( Y'' \), each proxy signcyrpeter \( id_{ji} \) calculates \( l = PV2_{ji}(Y'') \) and \( si_{gi} = F_{id_{ji}}(Y) \) and broadcasts \((l_i, si_{gi})\) to other proxy signcyrpertes.

4. Each proxy signcyrpeter \( id_{ji} \) collects information of \( L = (l_1, \ldots, l_n) \) and \( \text{Sing}(S_{gi_1}, \ldots, S_{gi_n}) \) then computes \( W_{ji} = F_{receiver}(L \parallel S_{gi} \parallel R) \) and \( Z_{ji} = H_2(L, S_{gi}, R) \oplus m \).

5. Proxy signcyrpertes send \((W = W_{ji}, Z = Z_{ji}, list)\) to the trusted third party which verifies whether \((W, Z, list)\) generated by each proxy signcyrpeter is equal in value. If yes, the final signcyrpition ciphertext is \( \sigma = (id_{Alice}, list, L, Z, W) \). Otherwise, the trusted third party ignores the tuple \((W, Z, list)\). At last, the trusted third party sends the ciphertext to the receiver \( F_{receiver} \).

### 4.5. De-Signcyrption Algorithm

When the receiver receives the threshold proxy signcyrption ciphertext \( \sigma = (id_{Alice}, list, L, Z, W) \), he will perform de-signcyrption algorithm. First, he checks whether the identities of proxy signcyrpertes in the list are legal by querying public authorization information by \( F_{Alice}(War) = H_a(m_a) \), the \( n \) queries whether threshold proxy signcyrpertes exercise the power of proxy within the effective period. If the results of the above checks are correct, the receiver performs the following steps.

1. By using his own private key, the receiver can compute \( F_{receiver}^{-1}(W) = L' \parallel S_{gi}' \parallel R' \) and \( S_{gi}' = (si_{g1}', \ldots, si_{gn}') \).

2. Judge \( L' \). If this equation does not hold, the receiver rejects the signcyrpition ciphertext. If so, the receiver continues to compute \( m' = Z \mathbin{\oplus} H_2(L' \parallel S_{gi}' \parallel R') \) and checks whether the type of the message is authorized. If not, the algorithm aborts and returns \( \bot \).

3. The receiver checks whether the following equation holds or not.

\[
F_p(L) = H_1(m' \parallel R' \parallel id_{Alice} \parallel list) = F_{id_{ji}}(si_{g1}') = \cdots = F_{id_{jn}}(si_{gn}')
\]

If the above equation holds, the receiver accepts the ciphertext and the plaintext is \( m' \). Otherwise, rejects the ciphertext and returns \( \bot \).

### 5. Correctness and Security Analysis

#### 5.1. Correctness Analysis

- **Theorem 1**: the verification process in the de-signcyrption is correct.

- **Proof**: After receiving the ciphertext \( \sigma = (id_{Alice}, list, L, Z, W) \), the receiver will perform the de-signcyrption algorithm. The receiver makes use of his own private key, and computes \( F_{receiver}^{-1}(W) = L' \parallel S_{gi}' \parallel R' = F^{-1}_{receiver}(F_{receiver}(L \parallel S_{gi} \parallel R)) \) = \( L' \parallel S_{gi} \parallel R \). \( L' \) holds due to the above equations. Thus \( m' = Z \mathbin{\oplus} H_2(L' \parallel S_{gi}' \parallel R') = Z \mathbin{\oplus} H_2(L \parallel S_{gi} \parallel R) \). The receiver calculates the message by using his private key.

\[
F_p(L) = H_1(m' \parallel R' \parallel id_{Alice} \parallel list) = H_1(m \parallel R \parallel id_{Alice} \parallel list) = F_{id_{ji}}(si_{g1}') = \cdots = F_{id_{jn}}(si_{gn}')
\]

So the computation process of the de-signcyrption progress is correct.

#### 5.2. Security Proof

MPKC is mainly based on the hardness of Multivariate Quadratic (MQ) problem and IP problem. We give proofs in the random oracle model of message confidentiality and ciphertext indistinguishability as follows.

- **Theorem 2**: for our scheme, if there is an probability polynomial time (PPT) that attacker A can win the game defined in definition1 with non-negligible advantage \( \epsilon \) in section 3.2, where attacker A can performs at most \( q_{si}i \) (i=0,1,2) query to hash functions \( H_i \) (i=0, 1, 2). The number of threshold signcyrption query is \( q_{sc} \), public key extract query is \( q_{jake} \), secret key extract query is \( q_{sake} \) and de-signcyrption query is \( q_{dake} \), then there exists an algorithm \( C \) that can transform the ability of attacker A to the advantage of solving MQ problem within PPT time. Its advantage \( \epsilon' \) satisfies \( \epsilon' \geq \frac{\epsilon}{q_{sc} + q_{jake}} - \frac{2^{q_{dake}}}{n^{200}} \).

- **Proof**: algorithm \( C \) wants to solve an instance of MQ problem \( F(x_0), y_0 = F(x_0) \), and \( C \) already knows \( y_0 \). In the following challenge game, \( C \) will make use of \( A \)'s ability to solve the MQ problem.

- **Setup**: \( C \) executes this algorithm and sets system
parameters. Randomly select two invertible affine transformations \((T, S)\) in finite field to be system private key and set system public key as \(\tilde{F} = T \circ F \circ S\). C selects invertible affine transformations \(T_0; GF(p)^n \rightarrow GF(p)^n\) and \(S_0; GF(p)^n \rightarrow GF(p)^n\). System partial private key is \((T_0 \circ T, S \circ S_0)\). C sends \(\text{params} = (G, p, q, H_0, H_1, H_2, F)\) to A. Attacker A receives system public key and outputs target identity \(i^*\) and its corresponding system key is \(i^*\). In order to handle queries to random oracle \(H_i\) \((i = 0, 1, 2)\) from A, C saves each query result into the corresponding \(H_i\)-list \((i = 0, 1, 2)\).

- **Phase 1**: A can launch queries to random oracle through C, and C gives a respond.

- **\(H_0\) query**: The tuple saved in \(H_0\)-list is \((id, list, t, F_p, r_0, T_0, S_0, F)\), where \((T_0, S_0)\) is part of \(id\)'s private key and its corresponding public key is \(F = T_0 \circ T \circ F \circ S_0 \circ S_1\). Then A inputs tuple \((id, list, t, F_o)\) as a query and check whether \((id, list, t, F_p, r_0, T_0, S_0, F)\) in \(H_0\)-list. If not, C randomly selects \(T_1 \in GF(p)^n\), computes \(h_0 = T_1 \circ T_0 \circ F \circ S_0 \circ S_1\) and returns it to A. And C saves \((id, list, t, F_p, r_0, T_1, S_1, F)\) into \(H_0\)-list if there exists the tuple, C will return the corresponding value \(h_0\) to A.

- **\(H_1\) query**: A inputs \((m, R, \text{id}_1, \text{list})\) and queries to \(H_1\), C searches whether there is a corresponding item in \(H_1\)-list. If not, C randomly selects \(h_1 \in GF(p)^n\), and puts \((m, R, \text{id}_1, \text{list}, h_1, T'_0, S'_0)\) into \(H_1\)-list, where \((T'_0, S'_0)\) is a part of proxy signcryption key generated by \(id_1\), then returns \(h_1\). If there exists a corresponding item, C directly returns \(h_1\) as respond.

- **\(H_2\) query**: A inputs \((L, \text{Sig}, R, \text{id}_1)\), if there exists corresponding \(z\) in \(H_2\)-list, then return \(z\) to A, otherwise, randomly select \(z \in \{0, 1\}^n\) and put the tuple \((L, \text{Sig}, R, \text{id}_1, z, \ldots)\) into \(H_2\)-list.

- **Secret Key Extract Query**: A issues secret key extract query to \(id\). C first checks whether \(id_1 = id^*\). If so, aborts this query. Otherwise C searches in \(H_0\)-list and extracts the record \((id, list, t, F_p, r_0, T_0, S_0, S_1)\). If there exists the record, C restores private key \((T_0 \circ T \circ F \circ S_0 \circ S_1)\) and return it to A. Otherwise, C selects \(T'_1; GF(p)^n \rightarrow GF(p)^n\) and \(S'_1: GF(p)^n \rightarrow GF(p)^n\), then saves them into \(H_0\)-list.

- **Public Key Extract Query**: on the input of identity \(id_1\), C searches whether there is a corresponding \(id_1\) in \(H_0\)-list. If exists, computes \(F_1 = T'_1 \circ T_0 \circ F \circ S_0 \circ S_1\) according to contents of the tuple and returns it. Otherwise, C gets corresponding private key through secret key extract query, computes the public key and returns it.

- **Proxy Authorization Extract Query**: take \((id, list)\) as input and execute proxy authorization query where \(id\) is the original signcrypter. If there exists corresponding item in \(H_0\)-list, C returns \(F_p\). Otherwise, C gets the private key of the original signcryption through secret key extract query and executes proxy key generation and authorization algorithm to generate corresponding proxy key. Then update the record of \(H_0\)-list and \(H_1\)-list. If \(id_1 = id^*\), C first selects invertible affine transformations \(T'_p; GF(p)^n \rightarrow GF(p)^n\) and \(S'_p: GF(p)^n \rightarrow GF(p)^n\). computes \(F_p = T'_1 \circ T_0 \circ F \circ S_0 \circ S'_p\). Next, performs \(H_0\) query, then puts \((id_1, list, t, F_p, r_0, h_0, \ldots)\) into \(H_0\)-list and puts \((\ldots, \text{id}_1, list, \ldots, T'_p, S'_p)\) into \(H_1\)-list.

- **Threshold Proxy Signcryption Query**: inputting \((m, \text{id}_1, list, \text{id}_1')\). If \(id_1 = id_1\) or \(id_1' = id^*\), then aborts. If both the original signcryption and proxy signcrypters' list don’t include the user to be attacked, and the receiver is not the user to be attacked, C performs secret key extract query and proxy authorization extract query to get corresponding private keys. If \(id_1 = id^*\), C executes proxy authorization extract query and puts \((id_1, list, t, F_p, r_0, h_0, \ldots)\) into \(H_0\)-list. Then randomly selects \(r \in GF(p)^n\) and computes \(R = \tilde{F}(r)\), gets \(h_1\) through \(H_1\) query and computes \(L = F_p^{-1}(h_1)\). C puts \((m, R, \text{id}_1, list, h_1, T'_p, S'_p)\) into \(H_2\)-list and executes secret key extract query for members in the list and computes \(\text{Sig}\). Then C runs public key extract query for receiver \(id_1\), and computes \(W = F_1(L \parallel \text{Sig} \parallel R)\). Cissues \(H_2\) query with \((L, \text{Sig}, R, \text{id}_1)\), gets \(z_1\) and computes \(Z = z_1 \oplus m\). Finally C puts \((L, \text{Sig}, R, \text{id}_1, z_1, W, Z)\) into \(H_2\)-list. C sends \(\sigma = (id, list, L, W, Z)\) to attacker A. If \(id^* \in list\), C performs secret key extract query for \(id_1\) and gets the corresponding private key. Then C runs proxy authorization extract query to calculate the proxy signcryption secret key \(F_p\). C only knows the public key of \(id^*\).Generate proxy signcryption ciphertext by the following steps: C selects \(\text{sign}^* \in GF(p)^n\) and \(r \in GF(p)^n\), computes \(R = \tilde{F}(r)\) and \(h_1 = F_1(id^*(\text{sign}^*))\). Then C puts the record \((m, R, \text{id}_1, list, h_1, T'_p, S'_p)\) into \(H_1\)-list, and computes \(L = F_p^{-1}(h_1)\), gets other members' private key in list by secret key extract query and calculation of threshold proxy signcryption algorithm. Finally, gets receiver's public key by public key extract query and computes \(W = F_1(L \parallel \text{Sig} \parallel R)\). At last, C performs \(H_2\) query with tuple \((L, \text{Sig}, R, id)\) to get \(z_1\) and computes \(Z = z_1 \oplus m\), puts \((L, \text{Sig}, R, id, z_1, W, Z)\) into \(H_2\)-list. At last C sends \(\sigma = (id, list, L, W, Z)\) to A.

- **Threshold De-signcryption query**: attacker A wants to obtain plaintext corresponding to threshold proxy signcryption ciphertext \(\sigma = (id, list, L, W, Z)\) and the receiver is \(id\). Suppose \(id_1 \neq id^*\), then C can get receiver’s private key by secret key extract query, uses de-signcrypted algorithm to obtain \(m\) and returns
it to $A$. If $id_r = id^*$ and there exists $(L, \Sigma, R, id, z, W, Z)$ in $H_2\text{-}list$, $G$ gets and returns it. If not, returns invalid ciphertext. Otherwise keeps on searching $H_1\text{-}list$ to check whether it includes $(m, R, id_1, list, h_1, T^*_p \circ s_p)$. If exists, $C$ calculates $m' = Z \oplus z_1$ and if $F_p(L) = h_1$, $C$ returns $m'$. If the ciphertext $\sigma$ cannot be decrypted correctly in the above two cases, return illegal ciphertext. In this phase, the failure probability of taking legal ciphertext as illegal ciphertext is $\frac{q_{dsc}}{n2^{[G]}}$ during the de-signcryption query. The probability of not finding the corresponding tuple in $H_2\text{-}list$, is less than $\frac{1}{n2^{[G]}}$, and the probability of not finding the corresponding tuple in $H_0\text{-}list$ is less than $\frac{1}{n2^{[G]}}$. Due to there are $q_{dsc}$ de-signcryption queries, so the probability of rejecting valid ciphertext is less than $\frac{q_{dsc}}{n2^{[G]}}$.

- **Challenge phase:** $A$ randomly selects two different messages $\{m_0, m_1\}$ with the same length, the original signcryption’s identity is $id_c$. $C$ randomly selects a bit $b$ and generates the threshold proxy signcryption challenge ciphertext. $C$ randomly selects $L' \in GF(p)^n$, $S \in GF(p)^n$, $R' \in GF(p)^n$ and $z' \in GF(p)^n$, sets $X_0 = L' \circ S \circ R$, and calculates $Y_0 = F^*(X_0)$ and $Z' = z' \oplus m_b$. At last $C$ sends the ciphertext $\sigma = (id_s, list, L', z', W')$ to attacker $A$.

- **Phase 2:** this is just as the same as Phase 1. $A$ issues queries to random oracle and gets the responds. But $A$ cannot issue queries to target identity $id^*$’s private key and also cannot query $\sigma'$ in de-signcryption query.

- **Guess:** $A$ outputs a guess value $b' \in \{0, 1\}$ in this phase. Through the above game, it can be concluded that the process effectively simulates situations to attack the scheme in reality. If $b' = b$, attacker $A$ will win the game. If $A$ wins, it should accept $H_2$ and gets $F^*(L \parallel S \parallel R) = Y_0$. $C$ randomly selects a record $(L, \Sigma, R, id_r, z_1, W, Z)$, and it will choose a record containing the right element which make $F^*(L' \parallel S \parallel R) = Y_0$ hold with probability $\frac{1}{q_{h_2} + q_{s_c}}$. At last, $C$ outputs the solution $X_0$ of MQ problem.

If the attacker breaks the confidentiality of the scheme, then $C$ can make use of it to solve MQ problem. Then we analyze the advantage of $C$. Let $E$ denotes the event of $A$ correctly outputting the right guess $b' = b$. Event $E_1$ issues query to target identity when executing secret key extract query. Event $E_2$ denotes signcryption failure, because the receiver is the target identity in some query to signcryption. Event $E_3$ denotes de-signcryption failure and $C$ rejects a valid ciphertext.

According to above discussions, $Pr(E) = \varepsilon$ is known. When $E$ happens, $E_1$ and $E_2$ don’t occur, that is $\neg E_1 \land \neg E_2$. The probability that $E_3$ occurs is less than $\frac{q_{dsc}}{n2^{[G]}}$. We use $E_4$ to denote the probability of $C$ choosing the right value from $H_2\text{-}list$ in guess phase and thus $Pr(E_4) \leq \frac{1}{q_{h_2} + q_{s_c}}$. So the advantage of $C$ is $\varepsilon' = Pr(E \land \neg E_2 \land \neg E_3 \land E_4) = \frac{1}{q_{h_2} + q_{s_c}} \left(1 - \frac{q_{dsc}}{n2^{[G]}}\right)$.

- **Theorem 3:** For our scheme, if there is an PPT forger $A$ that can win the game with non-negligible advantage $\varepsilon$, where $A$ can perform at most $q_{h_2}(i = 0, 1, 2)$ query to hash functions $H_i(i = 0, 1, 2)$, and the number of threshold proxy signcryption query is $q_{s_c}$, public key extract query is $q_{pke}$, secret key extract query is $q_{skc}$, verification query is $q_{skc}$, then there exists an algorithm $C$ that can transform the ability of forger $A$ to the advantage of solving IP problem within PPT time. Its advantage $\varepsilon'$ satisfies $\varepsilon' \geq \frac{1}{q_{h_2} + q_{s_c}} \left(1 - \frac{q_{dsc}}{n2^{[G]}}\right)$.

- **Proof:** algorithm $C$ wants to solve an instance of IP problem $(F^* = \bar{T} \circ T_0 \circ \bar{F} \circ S_0 \circ S', T_0 \circ \bar{F} \circ S_0)$. In the following challenge game, $C$ will make use of $A$’s ability to solve IP problem.

- **Setup:** $C$ executes this algorithm and sets system parameters. $G$ is a finite field whose characteristic is Randomly select two invertible affine transformations $(T, S)$ as system private key. Set system public key as $\bar{F} = T \circ T_0 \circ S$. Randomly selects $T_0: GF(p)^n \rightarrow GF(p)^n$ and $S_0: GF(p)^n \rightarrow GF(p)^n$, and system partial private key is $(T_0 \circ T, S \circ S_0)$. $C$ sends params $= (G, p, q, H_0, H_2, H_2, \bar{F})$ to $A$. Forger $A$ receives system parameters and outputs target identity $id^*$ and its corresponding public key is $F^*$. In order to handle query to random oracle $H_i(i = 0, 1, 2)$ from $A$, $C$ saves each query result into corresponding $H_i\text{-}list(i = 0, 1, 2)$.

- **Phase 1:** $A$ can launch query to random oracle defined in section 3.2. The processes of $H_0$ query, $H_1$ query, $H_2$ query, secret key extract query, public key extract query, and proxy authorization extract query, threshold proxy signcryption query are all the same as in Theorem 2. Here we need to add Verification query for proof.

- **Verification Query:** on the input of a receiver’s identity $id_r$ and ciphertext $\sigma = (id_s, list, L, W, Z)$. If $id_r \neq id^*$, $C$ gets receiver’s private key by secret key extract query, obtains plaintext $m$ according to the de-signcryption algorithm and returns it to $A$. Otherwise, $C$ checks $H_2\text{-}list$ to find whether there exists $(L, \Sigma, R, id, z, W, Z)$. If so, $C$ keeps on searching $H_1\text{-}list$ to check whether it includes $(m, R, id_1, list, h_1, T^*_p \circ s_p)$. If exists, $C$ calculates $m' = Z \oplus z_1$ and if $F_p(L) = h_1$, and returns $m'$. If the corresponding plaintext $m$ cannot be solved in the above two cases, return ciphertext illegal. In this
phase, the failure probability of taking legal ciphertext as illegal ciphertext resulting in de-signcryption query is $\frac{q_{\text{ver}}}{n^2}$, in fact, the probability of not finding the corresponding tuple in $H_2$-list is less than $\frac{1}{n^2}$ during de-signcryption query, the probability of not finding the corresponding tuple in $H_1$-list is less than $\frac{1}{n^2}$. Due to there are $q_{\text{ver}}$ de-signcryption queries, so the probability of rejecting valid ciphertext is less than $\frac{q_{\text{ver}}}{n^2}$.

- **Forger phase**: after performing the above polynomial-bounded query to random oracle, $A$ outputs a forgery signcryption ciphertext $\sigma = (id_s, list, L', Z', W')$ about $m$ and the signcryption ciphertext cannot be obtained by signcryption queries. In the ciphertext, $id_s = id^*$, that is, the original signcryption is $id^*$. In this situation, it is required that $id_s$ cannot be queried by secret key extract query in the above queries and $(id_s, list)$ cannot be queried by proxy authorization query. If $id^* \in list$, it is required that before outputting forgery ciphertext, $id_s$ cannot be queried by secret key extract query and $(id_s, list)$ cannot be queried by proxy authorization query in previous random oracle queries.

Through the above game, it can be concluded that the process effectively simulates situations to attack the scheme in reality. If $A$ successfully forges a threshold proxy signcryption ciphertext in the above game, then it has to get $(T^*, S^*)$ by $H_0$ query. $C$ chooses a record $(id_i, list, t, F_p, T'_i, S'_i, F_i)$ from $H_0$-list and lets $(T^*, S^*)$ as the solution of IP problem. The probability of $C$’s choosing $(T^*, S^*)$ correctly is $\frac{1}{q_{H_0}+q_{\text{sc}}}$. If the attacker breaks the unforgeability of the scheme, then $C$ can make use of it to solve IP problem. We will analysis the advantage of $C$. Let $E$ denotes the event of $A$ outputing the forgery ciphertext successfully. We use Event $E_1$, $E_2$ and $E_3$ defined in the theorem 2. According to the above discussions, When $E$ happens, $E_1$ and $E_2$ don’t happen, that is $\neg E_1 \land \neg E_2$. The probability that $E_3$ occurs is less than $\frac{q_{\text{ver}}}{n^2}$. We use $E_4$ to denote the probability of $C$ choosing the right value from $H_0$-list in guess phase and $Pr(E_4) \leq \frac{1}{q_{H_0}+q_{\text{sc}}}$. The advantage of $C$ is $\epsilon' = Pr(\neg E_1 \land \neg E_2 \land \neg E_3 \land E_4)$. where $\epsilon' \geq \frac{1}{q_{H_0}+q_{\text{sc}}} (1 - \frac{q_{\text{ver}}}{n^2})$.

6. Security Property Analysis and Scheme Comparison

6.1. Security Analysis

In this part, we list several important security properties and give a detailed analysis of our scheme.

- **Public verification**: the signcryption ciphertext $\sigma = (id_{\text{Alice}}, list, L, W, Z)$. contains the identity of the original signcryption and the proxy signcryption. Everyone can check whether the identity information in the signcryption ciphertext is legal and the expiry date is valid by querying the public license information ($m_0$, $\text{War}$) and verifying the equation $F_{\text{Alice}}(\text{War}) = H_0(m_0)$.

- **Non-repudiation**: the signcryption ciphertext $\sigma$ contains the signature generated by each proxy signcryption. Due to the difficulty of the IP problem of MPKC, the attacker cannot obtain the private keys of the proxy signcryption. Therefore, the participant who generates the threshold proxy signcryption ciphertext cannot deny the signcryption it generates.

- **Prevention of misuse**: the original signcryption generates a proxy signcryption authorization $m_w$, which indicates the type of agent the identities of the original signcryption and proxy signcrypters, and legal proxy signcryption expiry date $t$ and so on.

Thus the receiver can easily judge whether the threshold signcryption information is consistent with the authorization information.

- **Revocation**: if the original signcryption wants to revoke proxy authority, or update the proxy signcryption key, the original signcryption can revoke the license ($m_w$, $\text{War}$), so that the recipient doesn’t query the corresponding authorization information when performing verification, and rejects the ciphertext $\sigma$.

6.2. Scheme Comparisons

In this section, we compare our proposed scheme with several previous works about signcryption. Table 1 summarizes the comparison in terms of security properties. As can be seen from this table, these signcryption schemes all support confidentiality and unforgeability, which are the basic security properties of signcryption. Lin’s et al. [8] scheme cannot support threshold proxy signcryption, which may lead to abuse of power. Yang’s and Yu [18] scheme and Zhou and Yu [21] scheme realize the threshold proxy signcryption, but these two schemes are based on the traditional public key cryptosystem and cannot resist quantum attacks. Li’s et al. [7] scheme cleverly uses the multivariate public key cryptosystem to design a certificateless multi-recipient signcryption scheme, which can resist quantum computing attacks, however it does not support threshold proxy signcryption. Combining the advantages of these schemes, our scheme not only meets all the security features of threshold proxy signcryption, but also can resist quantum computing attacks.
7. Conclusions

We propose a threshold proxy signcryption scheme based on MPKC, which can resist quantum attacks. The confidentiality and unforgeability proofs of the scheme under the random oracle model are given based on the assumption of the MQ problem and the IP problem. Besides, the proposed scheme also satisfies properties of verifiability, non-repudiation and so on. Compared with the existing schemes, our scheme is suitable for the quantum computing environment. It provides theoretical and technical support for the application of signcryption technology on smart devices in the Internet of Things era. Nevertheless, our scheme may not be suitable for some special application scenarios. The future work is to design an anonymous threshold proxy signcryption scheme.

Acknowledgements

This work was supported in part by the National Key Technologies R and D Program of China under Grant No. 2018YFB1105303, the Natural Science Basic Research Plan in Shaanxi Province of China under Grant Nos. 2018JM6064 and 2019JM-129, and the National Cryptography Development Fund under Grant No. MMJJ20170208.

References

[1] Chan W. and Wei V., “A Threshold Proxy Signcryption,” in Proceedings of International Conference on Security and Management, Las Vegas, pp. 249-254, 2002.
[2] Gamage C. and Leiwj J., “An Efficient Scheme for Secure Message Transmission using Proxy-signcryption,” in Proceeding of the 20th Australasian Computer Science Conference, Australia, pp.18-21, 1998.
[3] Gao W., Hu Y., Wang B., and Xie J., “Improved Identification Protocol in the Quantum Random Oracle Model” The International Arab Journal of Information Technology, vol. 14, no. 3, pp. 339-345, 2017.
[4] Lai J., Mu Y., and Guo F., “Efficient Identity-based Online/offline Encryption and Signcryption with Short Ciphertext,” International Journal of Information Security, vol. 16, no. 3, pp. 299-311, 2017.
[5] Li F., Xin X., and Hu Y., “ID-based Threshold Proxy Signcryption Scheme from Bilinear Pairings,” International Journal of Security and Networks, vol. 3, no. 3, pp. 206-215, 2008.
[6] Li H. and Pang L., “Cryptanalysis of Wang et al.’s Improved Anonymous Multi-receiver Identity-based Encryption Scheme,” IET Information Security, vol. 8, no. 1, pp. 8-11, 2014.
[7] Li H., Chen X., Pang L., and Shi W., “Quantum Attack-resistant Certificateless Multi-receiver Signcryption Scheme,” PLoS One, vol. 8, no. 6, 2013.
[8] Lin H., Wu T., Huang S., and Yeh Y., “Efficient Proxy Signcryption Scheme with Provable CCA and CMA Security,” Computers and Mathematics with Applications, vol. 60, no. 7, pp. 1850-1858, 2010.
[9] Lu G., Xue L., Nie X., and Qin Z., “Cryptanalysis of Novel Extended Multivariate Public Key Cryptosystem with Invertible Cycle,” International Journal of Network Security, vol. 20, no. 3, pp. 509-514, 2018.
[10] Pan C., Li S., Zhu Q., Wang C., and Zhang M., “Notes on Proxy Signcryption and Multi-proxy Signature Schemes,” International Journal of Network Security, vol. 17, no. 1, pp. 29-33, 2015.
[11] Pang L., Hu Y., Liu Y., and Xu K., “Efficient and Secure Certificateless Signature Scheme in the Standard Model,” International Journal of Communication Systems, vol. 30, no. 5, pp. 1-14, 2017.
[12] Shor P., “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” SIAM Review, vol. 41, no. 2, pp. 303-332, 1999.
[13] Swapna G., Gopal P., Gowri T., and Reddy V., “An Efficient ID-based Proxy Signcryption Scheme,” International Journal of Information and Network Security, vol. 1, no. 3, pp. 200-206, 2012.
[14] Tanwar S. and Kumar A., “Extended Identity based Multi-signcryption Scheme with Public Verifiability,” Journal of Information and Optimization Sciences, vol. 39, no. 2, pp. 503-517, 2018.
[15] Wang M. and Liu Z., “Identity based Threshold Proxy Signcryption Scheme,” in Proceedings of the 5th International Conference on Computer and Information Technology, Shanghai, pp. 695-699, 2005.
[16] Wang Q., Cao Z., and Wang S., “Formalized Security Model of Multi-proxy Signature Schemes,” in Proceedings of the 5th International
Conference on Computer and Information Technology, Shanghai, pp. 668-672, 2005.

[17] Xue Q., Li F., Ge G., Shen J., and Cao Z., “Position-based Proxy Signcryption,” in Proceedings of IEEE/CIC International Conference on Communications in China, Shenzhen, pp. 1-6, 2015.

[18] Yang J. and Yu Z., “New Identity-based Threshold Proxy Signcryption Scheme,” Journal of Computer Applications, vol. 30, no. 1, pp. 121-124, 2010.

[19] Yeh J., “The Insecurity of Two Proxy Signcryption Schemes: Proxy Credential Forgery Attack and How to Prevent It,” The Journal of Supercomputing, vol. 70, no. 3, pp. 1100-1119, 2014.

[20] Zheng Y., “Digital Signcryption or How to Achieve Cost (Signature and Encryption) \ll \text{Cost} (\text{Signature})+\text{Cost} (\text{Encryption}),” in Proceeding of Annual International Cryptology Conference, Santa Barbara, pp. 165-179, 1997.

[21] Zhou M. and Yu Z., “New Double-threshold Proxy Signcryption Scheme from Bilinear Pairings,” Computer Engineering and Applications, vol. 47, no. 32, pp. 98-100, 2011.

Li Huixian is an Associate Professor with School of Computer Science and Engineering, Northwestern Polytechnical University, Xi’an, China. Her main research interests include information security, cryptography and secure protocols.

Gao Jin is currently pursuing the Master degree in School of Computer Science and Engineering, Northwestern Polytechnical University, Xi’an, China. His research interest focuses on attribute-based encryption.

Wang Lingyun is currently pursuing the Master degree in School of Computer Science and Engineering, Northwestern Polytechnical University, Xi’an, China. His research interest focuses on multivariate public key cryptosystem.

Pang Liaojun is a full professor with State Key Laboratory of Integrated Services Networks of Xidian University, and at the same time he was a visiting scholar at the Department of Computer Science of Wayne State University of USA. His research interests include Internet security, cryptography, secure mobile agent system and e-commerce security technology. He became a Member (M) of IEEE in 2009.