MPKC-based Threshold Proxy Signcryption Scheme

Li Huixian[1], Gao Jin[1], Wang Lingyun[1], and Pang Liaojun[2]
[1]School of Computer Science and Engineering, Northwestern Polytechnical University, China
[2]State 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-resistent 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 signcrypters 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 signcypter, which results in the right abuse of the proxy signcypter. 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 signcypter, which results in the possible right abuse of the proxy signcypter. To solve this problem, Chan and Wei [1] proposed the threshold proxy signcryption scheme. Threshold proxy signcryption schemes can achieve decentralized protection of group signcrption 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 threshold technology to signcryption theory, 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 signcrypter join flexibly. The most important thing is that our scheme can resist the attack of quantum computing. For convenience, we use MPKC-based Threshold Proxy Signcryption Scheme (MTPSC) 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 MTPSC 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:

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

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 $x=(x_1, ..., x_n)$ has the following form:

$$\begin{cases} f^{(1)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij}^{(1)} x_i x_j + \sum_{i=1}^{n} b_i^{(1)} x_i + c^{(1)} \\ f^{(2)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij}^{(2)} x_i x_j + \sum_{i=1}^{n} b_i^{(2)} x_i + c^{(2)} \\ \vdots \\ f^{(m)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij}^{(m)} x_i x_j + \sum_{i=1}^{n} b_i^{(m)} x_i + c^{(m)} \end{cases} \quad (2)$$

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 $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ 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 isomorphism, then $\bar{F} = T \circ F \circ S$ ($\circ$ denotes composition of mappings), where *T* and *S* are two reversible affine transformations on $GF(p)^n \to GF(p)^n$. Finding (*T, S*) from *F* to $\bar{F}$ such that $\bar{F} = T \circ F \circ S$ 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, $y=p(x_1, \ldots, x_n)=(y_1, \ldots, y_n)$ 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 MTPSC

In a threshold proxy signcryption scheme, there is an original signcrypter denoted by id_o. A group of proxy signcrypters that are appointed by the original signcrypter is denoted by pr= $\{id_1, id_2 \ldots, id_{sum}\}$ (*sum* is the number of proxy signcrypters). The receiver is denoted by id_r .Our scheme consists of the following five algorithms:

- *Setup* (1^{λ}): system initialization algorithm, which is executed by the KGC. Given a system security parameter 1^{λ}, KGC selects the necessary parameter *params* for the system.
- *Extract* (*id, params, s*): 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_o and mainly generates proxy signcryption keys for proxy signcrypters and authentication information which is used to show identity of the original signcrypter and proxy signcrypters and state message types of proxy signcryption and necessary authorization.
- *Threshold Proxy Signcrypt*: inputting related params and a message *m*, the proxy signcrypters will generate the signcryption ciphertext as legal representatives of the original signcrypter.
- *De-signcrypt*: this algorithm is executed by the receiver id_r, 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 partial 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_i 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_o and the proxy signcrypters' identities $Pr=\{id_1, id_2, \ldots, id_{num}\}$, The random oracle will run the Proxy Key Generation and Authorization algorithm to get the corresponding proxy key and authorization certificate m_w.
- *Threshold Proxy Signcrypt Query*: entering the message *m*, the original signer identity id_o and the proxy signcrypters $Pr=\{id_1, id_2, \ldots, id_{num}\}$, 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_o and $Pr=\{id_1, id_2, \ldots, id_{num}\}$ and the recipient identity id_r, 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 Π 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 identityid^*.
- *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$id_i (id_i \neq id^*)$, it runs key extraction algorithm and obtains the corresponding private keyPS_{id_i}.
- *Threshold Proxy Signcrypt Query*: when *C* receives threshold proxy signcrypt query (m, id_r, id_o, Pr), it calculates proxy signcryption ciphertext σ =*Threshold_proxy_signcrypt* (m, id_r, id_0, Pr) and sends it to *A*.
- *Threshold De-signcrypt Query*: when *C* receives de-signcryption query (σ, id_r), it will calculate the *m=De-singcrypt* (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 signcrypter's identity id_o, a list of proxy signcrypters Pr and the receiver's identity id^*. *C* randomly selects a message $m_b (b \in \{0,1\}$. calculates corresponding private keys of these users, and threshold proxy signcryption ciphertext σ^* = *Threshold_proxy_signcrypt* (m_b, id^*, id_0, 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 $Adv_{\Pi}^{IND-MTPSC-CCA}(A) = \left|Pr[b' = b] - \frac{1}{2}\right|$ .If the probability of guessing b' correctly in this game is less than ε for any IND-MTPSC-CCA attacker *A* in the polynomial time t', we call this scheme is (t', ε)-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 *Π* 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 asigncryption ciphertext σ of message *m* such that the ciphertext can be successfully de-signcrypted by the receiver. For the generation of signcryption ciphertext, the target user id_t is one of proxy signcrypters. It is also required that id_t is not authorized by the original signcrypter, 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 asigncryption ciphertext σ about message *m* such that the ciphertext can be successfully de-signcrypted by the receiver. For the generation of signcryption ciphertext, the target user id_t is one of proxy signcrypters and proxy signcrypters are not authorized by the original signcrypter. This type of attack is called chosen-authorization attack.

The advantage of forger *F* is defined as $Adv_{\Pi}^{\text{UF-MTPSC-CMA-CWA}}$ which is the probability of wining the above game. If the advantage of attacker's wining the above game is negligible, we call the scheme unforgeable.

4. Description of MTPSC Scheme

In this section we will give detailed design steps for our scheme. The MTPSC scheme is composed of the following five algorithms.

4.1. Setup Algorithm

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

The system public key of *KGC* is $(\bar{F}, F)$ and system private key is (*T*, *S*). KGC publishes $params = < G, p, q, H_0, H_1, H_2, \bar{F} >$ 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'_{idp} and S'_{idp} for this user and calculates $F_{idp} = T'_{idp} \circ \bar{F} \circ S'_{idp}$. The partial public key is F_{idp} and the partial private key is ($T_{idp} = T'_{idp} \circ T$, $S_{idp} = S \circ S'_{idp}$). *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 $F_{id} = T'_{id} \circ F_{idp} \circ S'_{id}$. The public key of the user id is F_{id} and the private key is ($T_{id} = T'_{id} \circ T'_{idp} \circ T$,$S_{id} = S \circ S'_{idp} \circ S'_{id}$). 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 signcrypters, 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 signcrypter Alice generates its proxy signcryption key and shares it with num proxy signcrypters. Alice's public key is F_{Alice} and his private key is (T_{Alice}, S_{Alice}). Alice selects two invertible affine transformations (T'_p, S'_p) infinite field, and computes $T_p = T'_p \circ T'_{Alice}$ and $S_p = S'_{Alice} \circ S'_p$ .Then the private key of the proxy signcryption is (T_p, S_p).The corresponding public key is

$$F_p = T'_p \circ T'_{Alice} \circ F_{Alicep} \circ S'_{Alice} \circ S'_p = T_p \circ F_{Alicep} \circ S_p \quad (3)$$

The inverse operation of the public key of proxy signcryption is$F_P^{-1} = S_P^{-1} \circ F_{Alicep}^{-1} \circ T_P^{-1}$. Then Alice will delegate the proxy key to the num proxy signcrypters. Both T_p and S_p are invertible affine transformations over $GF(p)^n \to GF(p)^n$, and the corresponding inverse T_P^{-1} and S_P^{-1} are also invertible affine transformations over $GF(p)^n \to GF(p)^n$. Here $T_P^{-1} = (((T_P^{-1})^{(1)})^T, ((T_P^{-1})^{(2)})^T, \dots, ((T_P^{-1})^{(n)})^T)^T$, Where $(T_P^{-1})^{(i)}{}_{(i=1,\dots,n)}$ is the *ith* element of T_P^{-1}.

4.3.2. Secret Sharing

Alice selects *num* different numbers $\beta_i \in GF(p)^n{}_{(i=1,2,\dots,num)}$ in finite field $GF(p)^n$ and uses them to construct a Vandermonde Matrix:

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_{num} \\ \vdots & \vdots & & \vdots \\ \beta_1^{n-1} & \beta_{num}^{n-1} & \cdots & \beta_{num}^{n-1} \end{pmatrix} \quad (4)$$

Then we calculate as follows.

$$\begin{aligned} B1 &= T_P^{-1} \times A \\ &= \begin{pmatrix} (T_P^{-1})^{(1)} \\ (T_P^{-1})^{(2)} \\ \vdots \\ (T_P^{-1})^{(n)} \end{pmatrix} \times \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_{num} \\ \vdots & \vdots & & \vdots \\ \beta_1^{n-1} & \beta_{num}^{n-1} & \cdots & \beta_{num}^{n-1} \end{pmatrix} \\ &= \begin{pmatrix} b1_{1,1} & b1_{1,2} & \cdots & b1_{1,num} \\ b1_{2,1} & b1_{2,2} & \cdots & b1_{2,num} \\ \vdots & \vdots & & \vdots \\ b1_{n,1} & b1_{n,2} & \cdots & b1_{n,num} \end{pmatrix} \\ &= (B1[1], B1[2], \dots, B1[num]) \end{aligned} \quad (5)$$

$$\begin{aligned} B2 &= S_P^{-1} \times A \\ &= \begin{pmatrix} (S_P^{-1})^{(1)} \\ (S_P^{-1})^{(2)} \\ \vdots \\ (S_P^{-1})^{(n)} \end{pmatrix} \times \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_{num} \\ \vdots & \vdots & & \vdots \\ \beta_1^{n-1} & \beta_{num}^{n-1} & \cdots & \beta_{num}^{n-1} \end{pmatrix} \\ &= \begin{pmatrix} b2_{1,1} & b2_{1,2} & \cdots & b2_{1,num} \\ b2_{2,1} & b2_{2,2} & \cdots & b2_{2,num} \\ \vdots & \vdots & & \vdots \\ b2_{n,1} & b2_{n,2} & \cdots & b2_{n,num} \end{pmatrix} \\ &= (B2[1], B2[2], \dots, B2[nu \end{aligned} \quad (6)$$

In the above formulas $(B1[i],B2[i])_{(i=1,2,\dots,num)}$ represents the i^{th} column vector. Thus, the original signcrypter Alice has split the proxy key into num parts to share. Alice makes $B_{id_i} = (B1[i], B2[i])$ as the private key of proxy signcrypter id_i and sends it to the proxy signcrypter via a secure channel.

4.3.3. Proxy Signcryption Authorization

Alice sends (B_{id_i}, β_i) to each proxy signcrypter id_i where (*i=1, 2,...,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 signcrypter id_{Alice}, members of proxy signcryption group Pr={id_1,id_2...,id_{num}}, appointed legal proxy signcryption expiry date *t*, public keys of proxy signcryption and some necessary information, that is m_w=<id_{Alice},id_1...,id_{num},*t*,F_p>. Then Alice signs the authorization by using a hash function H_0 and computes War under the inverse of Alice's public key F_{Alice}^{-1} , $War = F_{Alice}^{-1}(H_0(m_w)$, Alice publishes the authorization (m_w,*War*).

4.4. Threshold Proxy Signcrypt Algorithm

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

4.4.1. Identity Authentication of Proxy Signcrypters

First of all, each proxy broadcasts its identity to allthe other n-1 proxy signcrypters. Through a query of public authorization m_w, each proxy signcrypter makes sure whether other proxy signcrypters' identities are valid in the same group. Then, each proxy signcrypter verifies the equation F_{Alice}(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 $list =< id_{j_1}, id_{j_2}, \dots, id_{j_n} >$, where $1 \le j_1 < j_2 < \cdots < j_n \le num.$

4.4.2. Secret Sharing and The Proxy Signcryption Ciphertext Generation

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

1. Each proxy signcrypter's private key is $B_{id_{j_i}} = (B1[j_i], B2[j_i])_{(i=1,2,\dots,n)}$. According to the order in the identity list, each proxy signcrypter id_{j_i}sends the

kth element ($B1[j_i]_k$,$B2[j_i]_k$) of its private key and β_{j_i} to the user id_{j_k}.Then, each proxy signcrypter id_{j_i} performs the following computations:

$$(B1[j_1]_i, B1[j_2]_i, \dots, B1[j_n]_i) \times \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_{j_1} & \beta_{j_2} & \cdots & \beta_{j_n} \\ \vdots & \vdots & & \vdots \\ \beta_{j_1}^{n-1} & \beta_{j_2}^{n-1} & \cdots & \beta_{j_n}^{n-1} \end{pmatrix}^{-1} = (PV1_{i1}, PV1_{i2}, \dots, PV1_{in}) = PV1_i \quad (7)$$

$$(B2[j_1]_i, B2[j_2]_i, \dots, B2[j_n]_i) \times \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_{j_1} & \beta_{j_2} & \cdots & \beta_{j_n} \\ \vdots & \vdots & & \vdots \\ \beta_{j_1}^{n-1} & \beta_{j_2}^{n-1} & \cdots & \beta_{j_n}^{n-1} \end{pmatrix}^{-1} = (PV2_{i1}, PV2_{i2}, \dots, PV2_{in}) = PV2_i \quad (8)$$

2. Each proxy signcrypter id_{j_i} computes y_i=$PV1_i(Y)$ and sends y_i to a trusted third party.Receiving y_i from n proxy signcrypters, *KGC* calculates $Y' = (y_1, \dots, y_n)$ and $Y'' = F_{Alice}^{-1}(Y')$ by using the partial private key of the original signcrypter. Then, *KGC* broadcasts $Y''$to *n* proxy signcrypters.
3. After receiving Y'', each proxy signcrypter id_{j_i} calculates l_i=$PV2_i(Y'')$ and $sig_i = F_{id_{j_i}}^{-1}(Y)$ and broadcasts (l_i, sig_i) to other proxy signcrypters.
4. Each proxy signcrypter id_{j_i} collects information of L=(l_1, ..., l_n) and Sing=($Sing_1$, ..., $Sing_n$), then computes $W_{j_i} = F_{receiver}(L \parallel Sig \parallel R)$ and $Z_{j_i} = H_2(L, Sig, R) \oplus m$.
5. Proxy signcrypters send $(W = W_{j_i}, Z = Z_{j_i}, list)$ to the trusted third party which verifies whether (*W* ,*Z*, *list)* generated by each proxy signcrypter is equal in value. If yes, the final signcryption 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-Signcrypt Algorithm

When the receiver receives the threshold proxy signcryption ciphertext σ =(id_{Alice},*list*, *L*, *Z* ,*W*) , he will perform de-signcrypition algorithm. First, he checks whether the identities of proxy signcrypters in the list are legal by querying public authorization information by F_{Alice} (War)=H_o (m_w) , the *n* queries whether threshold proxy signcrypters 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 $F_{receiver}^{-1}(W) = L' \parallel Sig' \parallel R'$ and $\text{Sig}' = (sig_1', \dots, sig_n')$.
2. Judge $L = L'$. If this equation does not hold, the receiver rejects the signcryption ciphertext. If so, the receiver continues to compute $m' = Z \oplus H_2(L' \parallel Sig' \parallel R')$ and checks whether the type of the message is authorized. If not, the algorithm aborts and returns ⊥.
3. The receiver checks whether the following equation holds or not.

$$F_p(L) = H_1(m' \parallel R' \parallel id_{Alice} \parallel list) = F_{id_1}(sig_1') = \cdots = F_{id_n}(sig_n') \quad (9)$$

If the above equation holds, the receiver accepts the ciphertext and the plaintext is m'. Otherwise, rejects the ciphertext and returns ⊥.

5. Correctness and Security Analysis

5.1. Correctness Analysis

- *Theorem* 1: the verification process in the de-signcryption is correct.
- *Proof*: After receiving the ciphertext σ =(id_{Alice},*list*, *L*, *Z* ,*W*) , the receiver will perform the de-signcrypt algorithm. The receiver makes use of his own private key, and computes $F_{receiver}^{-1}(W) = L' \parallel Sig' \parallel R' = F_{receiver}^{-1}(F_{receiver}(L \parallel Sig \parallel R)) = L \parallel Sig \parallel R$. *L*=*L*' holds due to the above equations. Thus $m' = Z \oplus H_2(L' \parallel Sig' \parallel R') = Z \oplus H_2(L \parallel SigR))$. The receiver calculates the message by using his private key.

$$\begin{aligned} F_P(L) &= H_1(m' \parallel R' \parallel id_{Alice} \parallel list) \\ &= H_1(m \parallel R \parallel id_{Alice} \parallel list) \\ &= F_{id_{jt}}(sig'_{j_i}) \\ &= F_{id_{jt}}(Sig_{j_i}), (i = 1,2, \dots, n) \end{aligned} \quad (10)$$

So the computation process of the de-signcryption 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 ε in section 3.2, where attacker *A* can performs at most q_{Hi} (I=0,1,2) query to hash functions H_i(i=0, 1, 2).The number of threshold signcrypt query is q_{sc}, public key extract query is q_{pke}, secret key extract query is q_{ske} and de-signcrypt query isq_{dsc}, 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 ε' satisfies $\varepsilon' \geq \frac{\varepsilon}{q_{sc}+q_{H_2}}(1 - \frac{q_{dsc}}{n2^{|G|}})$.
- *Proof*: algorithm *C* wants to solve an instance of *MQ* problem$\langle F^*(x_0), y_0 = F^*(x_0)\rangle$, 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 $\bar{F} = \mathrm{T} \circ \mathrm{F} \circ \mathrm{S}$. *C* selects invertible affine transformations $T_0: GF(p)^n \to GF(p)^n$ and $S_0: GF(p)^n \to GF(p)^n$. System partial private key is $(T_0 \circ \mathrm{T}, \mathrm{S} \circ S_0)$. *C* sends $params = \langle G, p, q, H_0, H_1, H_2, F \rangle$ to *A*. Attacker *A* receives system parameters and outputs target identityid^* and its corresponding public key isF^*. 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 $(id_i, list, t, F_P, \tau_i, h0_i, T_i', S_i', F_i)$, where (T_i', S_i') is part ofid_i's private key and its corresponding public key is $F_i = T_i' \circ T_0 \circ \bar{F} \circ S_0 \circ S_i'(\tau_i)$.Then *A* inputs tuple(*id*$_i$, *list*, *t*, F_p) as a query and check whether$(id_i, list, t, F_P, -, -, T_i', S_i', F_i)$ in H_0-*list*. If not, *C* randomly selects $\tau_i \in GF(p)^n$, computes $h0_i = T_i' \circ T_0 \circ \bar{F} \circ S_0 \circ S_i'(\tau_i)$ and returns it to A. And *C* saves $(id_i, list, t, F_P, \tau_i, h0_i, T_i', S_i', F_i)$ into H_0-*list*.If there exists the tuple, *C* will return the corresponding value $h0_i$ to *A*.
- H_1 *query*: A inputs (*m*, *R*, *id*$_i$, *list*) and queries to H_1. *C* searches whether there is a corresponding item in H_1_list. If not, *C* randomly selects $h1_i \in GF(p)^n$, and puts $(m, R, id_i, list, h1_i, T_p', S_p')$ into H_1-*list*, where (T_p', S_p') is a part of proxy signcryption key generated by *id*$_i$, then returns $h1_i$. If there exists a corresponding item, *C* directly returns $h1_i$ as respond.
- H_2 *query*: *A* inputs (L, Sig, R, *id*$_i$), if there exists corresponding z_i in H_2-*list*, then return z_i to *A*, otherwise, randomly select $z_i \in \{0,1\}^{l_m}$ and put the tuple (L, Sig, R, *id*$_i$, z_i, -, -) into H_2-*list*.
- *Secret Key Extract Query*: *A* issues secret key extract query to *id*$_i$. *C* first checks whether $id_i = id^*$. If so, aborts this query. Otherwise *C* searches in H_0_list and extracts the record $(id_i, list, t, F_P, \tau_i, h0_i, T_i', S_i')$. If there exists the record, *C* restores private key $(T_i' \circ T_0 \circ \mathrm{T}, \mathrm{S} \circ S_0 \circ S_i')$ and return it to *A*. Otherwise, *C* selects $T_i': GF(p)^n \to GF(p)^n$ and $S_i': GF(p)^n \to GF(p)^n$, then saves them into H_0_*list*.
- *Public Key Extract Query*: on the input of identity id_i, *C* searches whether there is a corresponding id_iin H_0_*list*. If exists, computes $F_i = T_i' \circ T_0 \circ \bar{F} \circ S_0 \circ S_i'$ 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*$_i$, *list*) as input and execute proxy authorization query where id_i 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 signcrypter 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 $id_i = id^*$, *C* first selects invertible affine transformations $T_p': GF(p)^n \to GF(p)^n$ and $S_p': GF(p)^n \to GF(p)^n$,computes $F_p = T_i' \circ T_0 \circ \bar{F} \circ S_0 \circ S_p'$. Next, performs H_0 query, then puts$(id_i, list, t, F_P, \tau_i, h0_i, -, -, F_i)$ into H_0_list and puts$(-, -id_i, list, -, T_p', S_p')$ into H_1-*list*.
- *Threshold Proxy Signcrypt Query*: inputting $(m, id_s, list, id_r)$, If $id_s = id_r$ or$id_r = id^*$, then aborts. If both the original signcrypter 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 $id_s = id^*$, *C* executes proxy authorization extract query and puts $(id_i, list, t, F_P, \tau_i, h0_i, -, -, F_i)$ into H_0-*list*. Then randomly selects $\mathrm{r} \in GF(p)^n$ and computes $\mathrm{R} = \overline{F}(\mathrm{r})$, gets $h1_i$ through H_1 query and computes$\mathrm{L} = F_P^{-1}(h1_i)$. *C* puts $(m, R, id_i, list, h1_i, T_p', S_p')$ into H_1_list and executes secret key extract query for members in the list and computes Sign. Then *C* runs public key extract query for receiver id_r, and computes $\mathrm{W} = F_r(L \parallel Sig \parallel R)$.*C*issues H_2 query with (*L*, *Sig*, *R*, *id*$_r$), gets z_i and computes $\mathrm{Z} = z_i \oplus \mathrm{m}$. Finally *C* puts (L, Sig, R, *id*$_r$, z_i,W, Z) into H_2-*list*. *C* sends *σ*=(*id*$_s$, *list*, L, W, Z) to attacker *A*. If $id^* \in list$, *C* performs secret key extract query for id_s 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 $sign^* \in GF(p)^n$ and $\mathrm{r} \in GF(p)^n$, computes $\mathrm{R} = \overline{F}(\mathrm{r})$ and $h1_i = F_{id^*}(sign^*)$. Then *C* puts the record $(m, R, id_i, list, h1_i, T_p', S_p')$ into H_1_*list*, and computes $\mathrm{L} = F_P^{-1}(h1_i)$, 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 $\mathrm{W} = F_r(L \parallel Sig \parallel R)$. At last, *C* performs$H_2$ query with tuple (L, Sing, R, *id*$_r$) to get z_i and computes $\mathrm{Z} = z_i \oplus m$, puts (L, Sig, R, *id*$_r$, z_i, W, Z) into H_2-*list*. At last *C* sends *σ*=(*id*$_s$, *list*, L, W, Z) to *A*.
- *Threshold De-signcrypt query*: attacker *A* wants to obtain plaintext corresponding to threshold proxy signcryption ciphertext *σ*=(*id*$_s$, *list*, L, W, Z) and the receiver is *id*$_r$. Suppose $id_r \neq id^*$, then *C* can get receiver's private key by secret key extract query, uses de-signcrypt algorithm to obtain *m* and returns

it to *A*. If $id_r = id^*$ and there exists (L, Sig, R, id_r, z_i,W, Z) in H_2-*list*, *C* gets and returns it. If not, returns invalid ciphertext. Otherwise keeps on searching H_1*_list* to check whether it includes $(m, R, id_i, list, h1_i, T_p', S_p')$. If exists, *C* calculates $m' = \mathrm{Z} \oplus z_i$ and if F_P (L)=$h1_i$, *C* returns m'. If the ciphertext σ 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 $\frac{q_{dsc}}{n2^{|G|-1}}$ during the de-signcrypt query. The probability of not finding the corresponding tuple in H_2-*list*, is less than $\frac{1}{n2^{|G|}}$, and the probability of not finding the corresponding tuple in H_0*_list* is less than $\frac{1}{n2^{|G|}}$. Due to there are q_{dsc} de-signcrypt queries, so the probability of rejecting valid ciphertext is less than $\frac{q_{dsc}}{n2^{|G|-1}}$.

- *Challenge phase*: *A* randomly selects two different messages $\{m_0, m_1\}$ with the same length, the original signcrypter's identity is id_s. *C* randomly selects a bit *b* and generates the threshold proxy signcryption challenge ciphertext. *C* randomly selects $L^* \in GF(p)^n$, $Sign^* \in GF(p)^{n^2}$, $R^* \in GF(p)^n$ and $z^* \in GF(p)^n$, sets $X_0 = L^* \parallel Sign^* \parallel R^*$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 Phase1. *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 σ^* in de-signcrypt 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 access H_2 and gets $F^*(L \parallel Sign \parallel R) = Y_0$. *C* randomly selects a record $(\mathrm{L}, \mathrm{Sig}, \mathrm{R}, id_r, z_i, \mathrm{W}, \mathrm{Z})$, and it will choose a record containing the right element which make $F^*(L^* \parallel Sign^* \parallel R^*) = Y_0$ hold with probability $\frac{1}{q_{H_2}+q_{sc}}$. 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_1issues 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-signcrypt 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 $\neg E_1 \wedge \neg E_2$.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*_list* in guess phase and thus $Pr(E_4) \leq \frac{1}{q_{H_2}+q_{sc}}$. So the advantage of *C* is $\varepsilon' = Pr(E \wedge \neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4)$, where $\varepsilon' \geq \frac{1}{q_{H_2}+q_{sc}}(1 - \frac{q_{dsc}}{n2^{|G|}})$.

- *Theorem* 3: For our scheme, if there is an PPT forger *A* that can win the game with non-negligible advantage ε, where *A* can perform at most $q_{H_i}(i = 0,1,2)$ query to hash functions H_i (*i*=0,1,2), and the number of threshold proxy signcrypt query is q_{sc}, public key extract query is q_{pke}, secret key extract query is q_{ske}, verification query isq_{dsc}, 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 ε' satisfies $\varepsilon' \geq \frac{1}{q_{H_2}+q_{sc}}(1 - \frac{q_{dsc}}{n2^{|G|}})$.
- *Proof*: algorithm *C* wants to solve an instance of IP problem $\langle F^* = T^* \circ T_0 \circ \overline{F} \circ S_0 \circ S^*, T_0 \circ \overline{F} \circ S_0 \rangle$. 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 $\overline{F} = T \circ F \circ S$ .Crandomly selects $\mathrm{T}_0: GF(p)^n \to GF(p)^n$ and $\mathrm{S}_0: GF(p)^n \to GF(p)^n$, and system partial private key is $(T_0 \circ T, S \circ S_0)$. *C* sends $params = \langle G, p, q, H_0, H_1, H_2, \overline{F} \rangle$ 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-*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 signcrypt 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 σ=(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-signcrypt algorithm and returns it to *A*. Otherwise, *C* checks H_2*_list* to find whether there exists(L, Sig, R, id_r, z_i,W, Z). If so, *C* keeps on searching H_1*_list* to check whether it includes $(m, R, id_i, list, h1_i, T_p', S_p')$. If exists, *C* calculates $m' = \mathrm{Z} \oplus z_i$ and if F_P (*L*)=$h1_i$ 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-signcrypt query is $\frac{q_{ver}}{n2^{|G|-1}}$. In fact, the probability of not finding the corresponding tuple in H_2-*list* is less than $\frac{1}{n2^{|G|}}$ during de-signcrypt query, the probability of not finding the corresponding tuple in H_1-*list* is less than $\frac{1}{n2^{|G|}}$. Due to there are q_{ver} de-signcrypt query, so the probability of rejecting valid ciphertext is less than $\frac{q_{ver}}{n2^{|G|-1}}$.

- *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 signcrypter 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 $\frac{1}{q_{H_0}+q_{sc}}$.

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* outputting 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 $\neg E_1 \wedge \neg E_2$. The probability that E_3 occurs is less than $\frac{q_{ver}}{n2^{|G|}}$. We use E_4 to denote the probability of *C* choosing the right value from H_0_list in guess phase and $Pr(E_4) \leq \frac{1}{q_{H_0}+q_{sc}}$. The advantage of *C* is $\varepsilon' = Pr(E \wedge \neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4)$, where $\varepsilon' \geq \frac{1}{q_{H_0}+q_{sc}}(1 - \frac{q_{ver}}{n2^{|G|}})$.

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 σ=(id_{Alice}, *list*, L, W, Z). contains the identity of the original signcrypter and the proxy signcrypters. 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_w, *War*) and verifing the equation F_{Alice} (War) =H_0 (m_w).
- *Non-repudiation*: the signcryption ciphertext σ contains the signature generated by each proxy signcrypter. Due to the difficulty of the IP problem of MPKC, the attacker cannot obtain the private keys of the proxy signcrypters. Therefore, the participant who generates the threshold proxy signcryption ciphertext cannot deny the signcryption it generates.
- *Prevention of misuse*: the original signcrypter generates a proxy signcryption authorization m_w, which indicates the type of agent the identities of the original signcrypter 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 signcrypter wants to revoke proxy authority, or update the proxy signcryption key, the original signcrypter can revoke the license (m_w, *War*), so that the recipient doesn't query the corresponding authorization information when performing verification, and rejects the ciphertext σ.

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.

Table 1. Comparison of security properties.

	Li's Scheme [7]	Lin's Scheme [8]	Yang and Yu Scheme [18]	Meng's Scheme [21]	Our scheme
Confidentiality	√	√	√	√	√
Unforgeability	√	√	√	√	√
Prevention of misuse	×	×	√	√	√
Public verification	√	√	√	√	√
Non-repudiation	√	√	√	√	√
Revocation	×	×	√	√	√
Anti-quantum attack	√	×	×	×	√

7. Conclusions

We propose a threshold proxy signcryption scheme based on MPKC, which can resist quantum attack. 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 Leiwo J., "An Efficient Scheme for Secure Message Transmission using Proxy-signcryption," *in Proceeding of the 20nd 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-resistent 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) ≪ Cost (Signature)+Cost (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.