


# Post quantum proxy signature scheme based on the multivariate public key cryptographic signature

International Journal of Distributed  
Sensor Networks  
2020, Vol. 16(4)  
© The Author(s) 2020  
DOI: 10.1177/1550147720914775  
journals.sagepub.com/home/dsn  


Jiahui Chen<sup>1</sup>, Jie Ling<sup>1</sup>, Jianting Ning<sup>2</sup>, Emmanouil Panaousis<sup>3</sup>,  
George Loukas<sup>4</sup>, Kaitai Liang<sup>3</sup> and Jiageng Chen<sup>5,6</sup> 

## Abstract

Proxy signature is a very useful technique which allows the original signer to delegate the signing capability to a proxy signer to perform the signing operation. It finds wide applications especially in the distributed environment where the entities such as the wireless sensors are short of computational power and needed to be convinced to the authenticity of the server. Due to less proxy signature schemes in the post-quantum cryptography aspect, in this article, we investigate the proxy signature in the post-quantum setting so that it can resist against the potential attacks from the quantum adversaries. A general multivariate public key cryptographic proxy scheme based on a multivariate public key cryptographic signature scheme is proposed, and a heuristic security proof is given for our general construction. We show that the construction can reach *Existential Unforgeability under an Adaptive Chosen Message Attack with Proxy Key Exposure* assuming that the underlying signature is *Existential Unforgeability under an Adaptive Chosen Message Attack*. We then use our general scheme to construct practical proxy signature schemes for three well-known and promising multivariate public key cryptographic signature schemes. We implement our schemes and compare with several previous constructions to show our efficiency advantage, which further indicates the potential application prospect in the distributed network environment.

## Keywords

Post-quantum cryptography, multivariate public key cryptography, general construction, proxy signature, provable security

Date received: 28 December 2018; accepted: 2 March 2020

Handling Editor: José Camacho

## Introduction

The characteristic that some specified agents have the capability to proceed with the signing operations on behalf of the original signer turns out to be very attractive in case of the original signers temporal absence, short of computational power, and so on. It has already been shown in many previous researches that the proxy signature can be very helpful especially for application in the environments such as the wireless sensor networks,<sup>1</sup> Internet of things,<sup>2</sup> distributed shared object systems,<sup>3</sup> grid computing,<sup>4</sup> global

<sup>1</sup>School of Computers, Guangdong University of Technology, Guangzhou, China

<sup>2</sup>School of Computing, National University of Singapore, Singapore

<sup>3</sup>Surrey Centre for Cyber Security, University of Surrey, Guildford, UK

<sup>4</sup>University of Greenwich, London, UK

<sup>5</sup>School of Computer, Central China Normal University, China

<sup>6</sup>Central China Normal University Wollongong Joint Institute, Wuhan, China

### Corresponding author:

Jiageng Chen, School of Computer, Central China Normal University, No. 152 Luoyu Road, Wuhan, Hubei 430079, China.  
Email: chinkako@gmail.com



distribution networks,<sup>5</sup> and so on. However, most of the previous constructions are based on the hardness of number theory such as the integer factorization and discrete logarithm. According to Shor's algorithm,<sup>6</sup> Rivest–Shamir–Adleman (RSA) and some other algorithms based on the number theory will be broken in polynomial time after the emergence of quantum computers. And as a result, it will eventually lead to the break of most of the traditional cryptosystem.

To deal with the upcoming quantum computers, cryptographic researchers from different countries are beginning to devote themselves to explore the cryptosystems that can resist quantum computer attacks, that is, researching on the post-quantum cryptography.<sup>7</sup> Post-quantum cryptography can be divided into five categories: code-based cryptography, lattice-based cryptography, hash-based cryptography, multivariate public key cryptography (MPKC), and isogeny-based cryptography. The National Institute for Standards and Technology (NIST) has announced a formal call for proposals for post-quantum cryptography in fall 2016.<sup>8</sup> Thereafter, it formally provided the first round (round 1) submissions of post-quantum cryptographic standard protocols in December 2017.

The security of MPKC is based on solving a set of random quadratic multivariate equations on a finite field. So far, evidence does not reveal that quantum computers could solve this kind of questions effectively. Plus, MPKC schemes are in general much more effective than RSA in computing. But there are two drawbacks that become obstacles to use MPKCs. The first one is the large key sizes. The second drawback is that the security of MPKCs relies both on the multivariate quadratic (MQ) problem and on the Isomorphism of Polynomials (IP) problem, so the schemes in MPKCs are subjected to not only direct attacks but also structural attacks. This makes many MPKCs insecure, such as Matsumoto–Imai scheme,<sup>9</sup> balanced Oil, and Vinegar.<sup>10</sup> Under this situation, a number of attempts have been undertaken in order to tackle these two problems. For example, Courtois<sup>11</sup> studied provable security against key-only attack on Quartz, but the security against the chosen-message attack is unclear. Beyond these techniques, the Unbalanced Oil and Vinegar (UOV) scheme<sup>12</sup> is a well-known and deeply studied scheme in MPKC, Bulygin et al.<sup>13</sup> then presented an idea to reduce the public key size of the UOV signature scheme and provided provable security against direct attacks. Then, Sakumoto et al.<sup>14</sup> gave provable security of UOV against chosen-message attack, using the idea given in the study by Bellare and Rogaway,<sup>15</sup> which concatenates a random seed  $r$  with the signing message  $M$  so as to make the basic trapdoor one-way function of UOV become full domain hash (FDH). In Crypto2011, Sakumoto et al.<sup>16</sup> proposed provably secure identification/signature schemes based

on the MQ problem, which is a great improvement for the security of MPKCs.

According to NIST Computer Security Resource Center (CSRC): Cryptographic Technology Group,<sup>8</sup> MPKC is popular for its efficient signature scheme in the post-quantum cryptography (PQC) aspect and signature schemes are promising. As shown in the submission, there are nine signature schemes, named double exponentiation with matrix exponent (DME),<sup>8</sup> DualModeMS,<sup>17</sup> GeMSS,<sup>18</sup> Gui,<sup>19</sup> Hi-MQ, lifted unbalanced oil and vinegar scheme (LUOV)<sup>20</sup> (a variant of UOV), multivariate quadratic digital signature scheme (MQDSS),<sup>21</sup> Rainbow,<sup>22</sup> TPSig.<sup>23</sup> Among them, only Rainbow is the old scheme, and others are all published in the recent 5 years and the best scheme is LUOV, which has a combined not more than 20 kB size. However, it is the newest one which need time to confirm its security. Another promising scheme is MQDSS, which is provable secure one and the key size is relatively small. However, its resultant signature is too large compared with the original small message. Also, the running time of two schemes is not fast which in fact is the most advantage of MPKC schemes compared with other PQC schemes. Other schemes such as Rainbow is good at the running time and the security consideration but suffers from large key size.

Also, multivariate signature schemes with special properties, such as proxy signature, ring signature, are proposed. For example, Tang and Xu<sup>24</sup> proposed the first MPKC proxy signature scheme based on the problem of IP. Petzoldt et al.<sup>25</sup> proposed the first provable MPKC threshold ring signature scheme based on the result of Sakumoto et al.<sup>14</sup> Chen et al.<sup>26</sup> proposed the first online/offline signature based on UOV by utilizing the linear construction of the central map of UOV, so that the proposed scheme can be distributed in the wireless sensor networks. In addition, multivariate sequential aggregate signature scheme by Petzoldt et al.<sup>27</sup> and multivariate blind signature scheme by El Bansarkhani et al.<sup>28</sup> are proposed to enrich this area.

Since proxy signature scheme is widely used as a communication solution in distributed sensor networks, that is, the solution in healthcare wireless sensor networks in Verma et al.<sup>29</sup> In this article, we focus on developing multivariate signature schemes with special properties and investigate how to build an MPKC proxy scheme and then we will use this idea to build a series of MPKC proxy schemes based on current promising MPKCs (UOV, Rainbow, MQDSS). A highlight of our work is the general proxy scheme is formally provable security under the assumption that the basing scheme is secure, so based on the promising MPKC signature schemes, our practical resultant proxy schemes from the general construction are considered promising proxy MPKC schemes.

This article is structured as follows: First, we introduce how to build a general MPKC proxy scheme, specifically

we give a formal proof for the general scheme assuming the underlying MPKC scheme is secure. Then we propose three practical proxy signature schemes: Proxy-UOV, Proxy-Rainbow, and Proxy-MQ. Next, we run some experiments to verify the security and efficiencies of our schemes. Finally, we draw a conclusion.

## Preliminaries

In this section, we give the preliminaries of this article.

### Proxy signature scheme

A proxy signature protocol allows an entity, called the original signer, to delegate another entity, called a proxy signer, to sign messages on behalf of itself, in case of temporal absence, lack of time or computational power, and so on. The first efficient proxy signature was introduced by Mambo et al.<sup>30</sup> A proxy signature scheme consists of the following algorithms: *Setup*, *KeyGen*, *Delegate*, *ProxyKeyGen*, *ProxySign*, *ProxyVerify*, where the *Setup* and *KeyGen* correspond with an ordinary signature scheme  $\Pi = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verify})$ . Assume  $(pk_A, sk_A)$  and  $(pk_B, sk_B)$  generated by *KeyGen* are the public/private key pairs of the original signer and the proxy signer, respectively. *Delegate* is a randomized algorithm for the delegation of signing right, that is,  $\theta \leftarrow \text{Delegate}(w, sk_A, pk_A)$  is run by the original signer. The input  $w$  can be regarded as a warrant. In general,  $w$  includes the original signer's public key  $pk_A$ , the proxy signer's public key  $pk_B$ , a delegated time period  $t$ , and other information. *ProxyKeyGen* algorithm generates the proxy key  $psk$  for the proxy signer. For a message  $m$ , the proxy signer can generate a proxy signature by *ProxySign*, that is,  $\sigma \leftarrow \text{ProxySign}(m, (w, \theta), psk)$ . Anyone who receives the proxy signature can verify the validity of the signature by *ProxyVerify*. *ProxyVerify* outputs 1 if the signature is valid; otherwise, it outputs 0.

### MPKC signature scheme

Usually, an MPKC scheme over a finite field  $k$  is defined as

$$P = L_1 \circ F \circ L_2$$

in which  $F$  is a set of  $m$  quadratic multivariate polynomials in  $n$  variables,  $L_1$  is an affine transformation from  $\mathbb{F}_q^m$  to  $\mathbb{F}_q^m$ , and  $L_2$  is an affine transformation from  $\mathbb{F}_q^n$  to  $\mathbb{F}_q^n$ .

For an MPKC digital signature scheme, the setup algorithm *Setup* ( $1^\lambda$ ) takes  $1^\lambda$  as input and then outputs the system parameter  $param$  which mainly contains  $(q, n, m)$ , and all the arithmetic operations hereafter are over this finite field.

### Algorithm 1. $\text{Sign}(M, sk)$ .

---

#### Input

$M$ : the message;  
 $sk$ :  $sk = (L_1, F, L_2)$ , the private key;

#### Output

$\sigma$ : the signature on message  $m$ ;

#### Begin

Step 1. The signer computes  $M_1 = L_1^{-1}(M)$ ;

Step 2. The signer computes  $M_2 = F^{-1}(M_1)$ ;

Step 3. The signer computes  $\sigma = L_2^{-1}(M_2)$ ;

Step 4. Return  $\sigma$ ;

#### End

---

The key generation algorithm *KeyGen*( $param$ ) takes  $param$  as input and then outputs  $pk = P$  and  $sk = (L_1, F, L_2)$ .

The signing algorithm  $\text{Sign}(M, sk)$  is described in Algorithm 1.

Finally, the verification algorithm  $\text{Verify}(\sigma, M, P)$  returns 1 if  $P(\sigma) = M$ , otherwise returns 0.

### Security model for MPKC signature scheme

We quantify the security of MPKC signature scheme from the idea in Bellare and Rogaway.<sup>15</sup> A signature scheme is said to be  $(\epsilon, t, q_s)$ -secure if an attacker, given the public key, allowed to run in time  $t$ , and allowed a chosen-message attack in which he or she can get  $q_s$  legitimate message-signature pairs, can be successful in forging the signature of a new message with probability at most  $\epsilon$ .

*Definition 1.* We say that the MPKC signature scheme is  $(\epsilon, t, q_s)$ -secure if there is no forger  $\mathcal{A}$  who takes a public key generated through  $(pk, \cdot) \leftarrow \text{KeyGen}(1^\lambda)$ , after at most  $q_s$  signature queries, and  $t$  processing time, then outputs a valid signature with probability at least  $\epsilon$ .

### UOV and Rainbow

The UOV scheme is one of the earliest MPKC signature scheme. Even though its construction is very simple, it turns out to be one of the most secure MPKC scheme so far. However, Rainbow is one of the most popular schemes in MPKC schemes and rapid development in recent years. It could be regarded as an extension of UOV and has obvious advantages over efficiency and key size.

The central map  $F$  of UOV is composed of a set of so-called Oil-Vinegar polynomials which have the following form

$$\sum_{i=1}^a \sum_{j=1}^v a_{ij} x_i x'_j + \sum_{i=1}^v \sum_{j=1}^v b_{ij} x'_i x'_j + \sum_{i=1}^a c_i x_i + \sum_{j=1}^v d_j x'_j + e \quad (1)$$

In this polynomial, there are two kinds of variables: Oil variables ( $x_i$ ) and Vinegar variables ( $x'_j$ ). Once we assign a set of random values for Vinegar variables, the central map becomes a set of linear polynomials and can be easily inverted. When  $v > o$ , this scheme is called UOV scheme, the construction of a UOV scheme is as follows

$$P = F \circ T \quad (2)$$

where  $T$  is an affine translation from  $\mathbb{F}_q^n$  to  $\mathbb{F}_q^n$ . The construction does not have to compose an invertible affine transformation on the left.

In the case of Rainbow, Rainbow is an extension of UOV scheme. It could be viewed as a multi-layer UOV scheme. Each layer is an independent UOV and each layer's variables (including Oil variables and Vinegar variables) are Vinegar variables of the next layer. Specifically, let us assume a Rainbow has  $l$  layers. We use  $v_i$  to represent the number of Vinegar variables of the  $i$ th layer and  $o_i$  to represent the number of Oil variables of the  $i$ th layer. Then we have  $v_{i+1} = o_i + v_i$  and  $v_{l+1} = n$ . Each layer's Vinegar variables set and Oil variables set are represented as  $\{x_1, \dots, x_{v_i}\}$ ,  $\{x_{v_i+1}, \dots, x_{v_i+o_i}\}$  and the  $i$ th layer's polynomials have the form of

$$\sum_{i=1}^{v_i} \sum_{j=v_i+1}^{v_i+o_i} a_{ij} x_i x_j + \sum_{i=1}^{v_i} \sum_{j=1}^{v_i} b_{ij} x_i x_j + \sum_{i=1}^{v_i+o_i} c_i x_i + d \quad (3)$$

We can see that the above polynomial has the basic Oil-Vinegar polynomial form. Finally, the construction of a Rainbow scheme is given as follows

$$P = L_1 \circ F \circ L_2 \quad (4)$$

### The MQ-based signature scheme

At CRYPTO 2011 Sakumoto et al.<sup>16</sup> presented a new identification scheme whose security is solely based on the MQ problem. In the scheme, every user chooses a vector  $\mathbf{s} \in \mathbb{F}_q^n$  as his secret key and computes his public key as  $\mathbf{v} = F(\mathbf{s}) \in \mathbb{F}_q^m$ . To identify himself to a verifier, he or she has to show that he or she indeed knows  $\mathbf{s}$  (without revealing any information about  $\mathbf{s}$ ). Thus, to create a zero-knowledge proof of the vector  $\mathbf{s}$ , we need the polar form of the multivariate system  $F$ , which is defined as

$$G(\mathbf{x}, \mathbf{y}) = F(\mathbf{x} + \mathbf{y}) - F(\mathbf{x}) - F(\mathbf{y})$$

Note that  $G(\mathbf{x}, \mathbf{y})$  is bilinear in  $\mathbf{x}$  and  $\mathbf{y}$ , the knowledge of  $\mathbf{s}$  is equivalent to knowing a tuple  $(\mathbf{r}_0, \mathbf{r}_1, \mathbf{t}_0, \mathbf{t}_1, \mathbf{e}_0, \mathbf{e}_1)$  satisfying

$$G(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0 = \mathbf{v} - F(\mathbf{r}_1) - G(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_1$$

and  $(\mathbf{t}_0, \mathbf{e}_1) = (\mathbf{r}_0 - \mathbf{t}_1, F(\mathbf{r}_0) - \mathbf{e}_1)$ . The five-pass identification scheme between a prover and a verifier is as follows

1. The prover chooses randomly  $\mathbf{t}_0, \mathbf{r}_0 \in_R \mathbb{F}_q^n, \mathbf{e}_0 \in_R \mathbb{F}_q^m$ , set  $\mathbf{r}_1 = \mathbf{s} - \mathbf{r}_0$  and computes commitments  $c_0 = \text{Com}(\mathbf{r}_0, \mathbf{t}_0, \mathbf{e}_0), c_1 = \text{Com}(\mathbf{r}_1, G(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0)$  and then sends  $(c_0, c_1)$  to the verifier.
2. The verifier chooses randomly a choice  $\alpha \in_R \mathbb{F}_q$  and sends  $\alpha$  to the prover.
3. After receive  $\alpha$ , the prover computes  $\mathbf{t}_1 = \alpha \mathbf{r}_0 - \mathbf{t}_0, \mathbf{e}_1 = \alpha F(\mathbf{r}_0)$  and then sends  $(\mathbf{t}_1, \mathbf{e}_1)$  to the verifier.
4. The verifier chooses randomly the challenge  $Ch \in_R \{0, 1\}$  and sends  $Ch$  to the prover.
5. If  $Ch = 0$ , the prover sends  $Rsp = \mathbf{r}_0$  back; if  $Ch = 1$ , the prover sends  $Rsp = \mathbf{r}_1$  back.
6. If the verifier chooses 0 as the challenge  $Ch$ , he or she checks whether  $c_0 = \text{Com}(\mathbf{r}_0, \alpha \mathbf{r}_0 - \mathbf{t}_0, \alpha F(\mathbf{r}_0) - \mathbf{e}_1)$  hold. If the verifier chooses 1 as the challenge  $Ch$ , he or she checks whether  $c_1 = \text{Com}(\mathbf{r}_1, \alpha \mathbf{v} - F(\mathbf{r}_1) - G(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_1)$  holds.

This scheme has a cheating probability per round of  $3/4$  when  $q = 2$ . Therefore, one needs at least 133 rounds to reduce the impersonation probability to less than  $2^{-80}$ . Sakumoto et al.<sup>16</sup> propose for their five-pass schemes that  $q = 2, n = 80, m = 80$  to achieve a security level of 80 bits.

Using the Fiat-Shamir paradigm,<sup>31</sup> anyone can transform the MQ identification scheme into a signature scheme, and a good example is the MQDSS<sup>21</sup> which is transformed from five-pass MQ identification scheme. Below we give a short description of the MQ signature scheme. For the full description of the MQ scheme, we recommend to read.<sup>21</sup> The setup and key generation process for the signature scheme work just the same as the identification scheme.

To generate a signature for a message  $m$ , the signer gathers the commitments for all rounds, creates the commitments  $c_0^{(1)}, c_1^{(1)}, \dots, c_0^{(round)}, c_1^{(round)}$ , and then uses a hash function  $\mathcal{H}$  to produce the challenge vector  $Ch$  and compute the according responses  $Rsp^{(1)}, \dots, Rsp^{(round)}$ . Finally, the signature is  $\sigma = (c_0^{(1)} || c_1^{(1)} || \dots || c_0^{(round)} || c_1^{(round)}) || Rsp^{(1)} || \dots || Rsp^{(round)}$ .

To verify the authenticity of a signature, the verifier parses  $\sigma$ , computes the challenge vector  $Ch$ , and tests for each  $i \in 1, \dots, round$  if  $Rsp^i$  is a correct response to  $Ch_i$  according to  $c_0^{(i)}, c_1^{(i)}$ .

### Security model for proxy signature

Schuldt et al.<sup>32</sup> presented the security notion *Existential Unforgeability under an Adaptive Chosen Message Attack with Proxy Key Exposure* (ps-uf-pke) for

multilevel proxy signature scheme. Later, Tang and Xu<sup>24</sup> modified this notion to single-level proxy signature scheme and adopted as the security model for a proxy signature. In the analysis of our proxy scheme, we also use this model, and we recommend to read more information about this model in Tang and Xu.<sup>24</sup> We summarize the security model for a proxy signature in the following.

In our security model for proxy signature, the definition is the same as that in Tang and Xu.<sup>24</sup> In the security model in Tang and Xu,<sup>24</sup> it uses only one list  $psklist$  to store the proxy key which generated by  $\mathcal{C}$ , but it does not use lists to store the original signature query and the proxy query, this is ambiguous for someone to calculate the number and kinds of query oracle. So, in our security model, we use three initialized empty lists:  $OSList$ ,  $delList(w)$ ,  $pskList(w)$  which are maintained by a challenger  $\mathcal{C}$ . The  $OSList$  stores all the signatures which are queried by the original signature query, and the  $delList(w)$  stores the submitted warrants and the corresponding delegation. The  $pskList(w)$  stores all the proxy key which is generated by  $\mathcal{C}$  from the warrants in the  $delList(w)$ . Using these three lists, we can follow the security proof more clearly. More precisely, by calculating the list  $OSList$ , we can know how many times of ordinary signature oracle have been queried, and from  $delList(w)$ , we can know the number of query of proxy signature oracle. The security model is based on the following game which is played between a challenger  $\mathcal{C}$  and an adversary  $\mathcal{A}$ :

- *Setup*. The challenger  $\mathcal{C}$  runs  $\text{Setup}$  with input  $1^k$  and generates  $(pk^*, sk^*)$  for  $u^*$  by running the  $\text{KeyGen}(1^k)$  of an ordinary signature scheme. After that,  $\mathcal{C}$  sends  $pk^*$  to the adversary  $\mathcal{A}$  and stores  $sk^*$ .
- *Queries*. The adversary  $\mathcal{A}$  can adaptively access to any of the following queries which are answered by  $\mathcal{C}$ :
  1. *Ordinary signature*.  $\mathcal{A}$  submits a message  $m$  to  $\mathcal{C}$ ,  $\mathcal{C}$  generates a signature on  $m$  by  $\sigma \leftarrow \text{Sign}(m, sk^*)$ .  $\mathcal{C}$  returns  $\sigma$  to  $\mathcal{A}$  and adds  $(m, \sigma)$  to the  $OSList$  list.
  2. *Delegation to  $u^*$* .  $\mathcal{A}$  transmits  $w$  to  $\mathcal{C}$ ,  $\mathcal{C}$  interacts with  $\mathcal{A}$  through the  $\text{Delegate}$  and the  $\text{ProxyKeyGen}$  with  $(pk^*, sk^*)$ . After the process finished,  $\mathcal{C}$  will obtain a delegation of signing right with  $\theta \leftarrow \text{Delegate}(w, sk^*, pk^*)$  and a proxy key  $sk_p$ . Then,  $\mathcal{C}$  adds  $(w, \theta)$  and  $(w, sk_p)$  to the  $delList(w)$  list and the  $pskList(w)$  list, respectively.
  3. *Delegation from  $u^*$* .
    1. *Delegation of  $sk'$* :  $\mathcal{A}$  submits  $w$  to  $\mathcal{A}$  and want  $u^*$  to give a delegation of signing right to  $u'$ .  $\mathcal{C}$  generates a delegation  $\theta$  for  $u'$  by  $\text{Delegate}$  and adds  $(w, \theta)$  to the  $delList(w)$  list.

2. *Self-delegation*:  $\mathcal{C}$  interacts with itself through  $\text{Delegate}$  and  $\text{ProxyKeyGen}$  on input  $w$ .  $\mathcal{C}$  generates a delegation of signing right  $\theta$  and a proxy key  $sk_p$ , adds  $(w, \theta)$  and  $(w, sk_p)$  to the  $delList(w)$  list and the  $pskList(w)$  list, respectively. Then  $\mathcal{C}$  sends  $\theta$  to  $\mathcal{A}$ .
  1. *Proxy signature*.  $\mathcal{A}$  transmits  $(m, (w, \theta))$  to  $\mathcal{C}$  and wants to obtain a proxy signature of  $m$ .  $\mathcal{C}$  finds the proxy key  $sk_p$  which correspond with  $w$  in  $pskList(w)$ . If  $sk_p$  exists,  $\mathcal{C}$  returns  $\sigma \leftarrow \text{ProxySign}(m, (w, \theta), sk_p)$  to  $\mathcal{A}$ . Otherwise, returns  $\perp$  to  $\mathcal{A}$ .
  2. *Proxy key exposure*.  $\mathcal{A}$  transmits  $w$  to  $\mathcal{C}$ ,  $\mathcal{C}$  returns the proxy key  $sk_p$  to  $\mathcal{A}$  if such a key exists in  $pskList(w)$ . Otherwise, returns  $\perp$  to  $\mathcal{A}$ .
- *Forgery*. The successful forgery of  $\mathcal{A}$  can be one of the following forms:
  1. *Forge an ordinary signature of  $u^*$* .  $\mathcal{A}$  outputs  $(m, \sigma)$  which can be verified by  $\text{Verify}$  and  $m$  has not been submitted in an ordinary signature query.
  2. *Forge a proxy signature of  $u^*$* .  $\mathcal{A}$  outputs  $(m, (w, \theta), \sigma)$  where the  $\sigma$  corresponds to the public key  $pk^*$ . This forgery is said to be valid if it can be verified by  $\text{ProxyVerify}$  with  $(m, (w, \theta))$  has not been queried and  $w$  has not been queried.
  3. *Self-proxy signature on behalf of  $u^*$* .  $\mathcal{A}$  outputs  $(m, (w, \theta), \sigma)$  where  $\sigma$  corresponds to the public key  $pk_p$ . This forgery is said to be valid if it can be verified by  $\text{ProxyVerify}$  with  $w$  has not been queried.

If one of the above cases happens, the game returns 1. Otherwise, it returns 0.

**Definition 2.** An adversary  $\mathcal{A}$  is said to be a  $(\epsilon', t', q'_d, q'_s)$ -forger of a proxy signature scheme if  $\mathcal{A}$  has advantage at least  $\epsilon'$  in the above game, runs in time at most  $t$ , and makes at most  $q'_d$  and  $q'_s$  delegation and signing queries to the challenger. A proxy signature scheme is said to be  $(\epsilon', t', q'_d, q'_s)$ -secure if no  $(\epsilon', t', q'_d, q'_s)$ -forger exists.

## The general construction of MPKC proxy signature scheme

### General proxy signature scheme

Assume we have an above MPKC signature scheme, we now describe our general proxy signature scheme as follows.

*Setup*: Let  $n$  and  $m$  be two positive integers,  $\mathbb{F}_q$  is a finite field and all the arithmetic operations hereafter

are over this finite field.  $\mathcal{H}: \{0, 1\}^* \rightarrow \mathbb{F}_q^n$  is a cryptographic hash function.

**KeyGen:** This algorithm generates the public key and the private key of a user. The detail process is the same as that of the key generation of an MPKC signature scheme. After this algorithm, we set the private key of user  $A$  is  $sk_A = (L_{1A}, L_{2A}, F_A)$ , and the public key  $pk_A = P_A$ , where  $P_A = L_{1A} \circ F_A \circ L_{2A}$ . Similarly, the private key and public key of user  $B$  are  $sk_B = (L_{1B}, L_{2B}, F_B)$  and  $pk_B = P_B$ , respectively.

**Delegate:** On input a warrant  $w$  where  $w = (pk_A, pk_B, t)$ , this algorithm is performed by user  $A$  and generates a delegation of signing right to user  $B$ .

1. Randomly choose two invertible affine transformations  $L'_1$  and  $L'_2$ , which in the forms of  $L_{1A}$  and  $L_{2A}$ , respectively. Then, compute  $L_1 = L'_1 \circ L_{1A}^{-1}$ ,  $L_2 = L_{2A}^{-1} \circ L'_2$ , and
 
$$P'_A = L_1 \circ P_A \circ L_2 \quad (5)$$
2. Compute  $\theta = \text{Sign}(\mathcal{H}(w||P'_A), sk_A)$ .
3. Send  $(L_1, L_2, P'_A)$  and the warrant  $(w, \theta)$  to user  $B$  through an authenticated channel.

**ProxyKeyGen:** This algorithm generates a proxy key for the proxy signer with input  $(L_1, L_2, P'_A, w, \theta)$ . It is performed by user  $B$  as follows:

1. Verify the validity of  $P'_A$  and  $\theta$ . If they are true, goto the next step. Otherwise, output 0.
2. Select two random invertible affine transformations  $L''_1$  and  $L''_2$ , respectively, compute

$$L_{1p} = L_1 \circ L_{1B}^{-1} \circ L''_1 \quad (6)$$

$$L_{2p} = L''_2 \circ L_{2B}^{-1} \circ L_2 \quad (7)$$

and

$$F_p = L''_1 \circ L_{1B} \circ P_A \circ L_{2B} \circ L''_2 \quad (8)$$

3. Compute a signature on  $(w, \theta, F_p)$  through running  $\text{Sign}$  with  $sk_B$ , that is,  $\sigma_{prx} = \text{Sign}(\mathcal{H}(w||\theta||P'_A), sk_B)$ .
4. Output  $sk_p = (L_{1p}, L_{2p}, F_p)$  as a proxy key of user  $B$  which uses to generate proxy signatures on behalf of user  $A$ , and the corresponding public key is  $pk_p = P'_A$ .

**Remark 1.** Note that since  $F_p = L''_1 \circ L_{1B} \circ P_A \circ L_{2B} \circ L''_2$ , to invert  $F_p$ , we do not need the secret key  $F_A$  of user  $A$ , but the public key  $P_A$  of user  $A$ , what we need to choose is the linear transformations such that the map  $F_p$  can still

be easily inverted. This is the key point in the construction; otherwise, the construction cannot work. In some cases of MPKCs, the linear transformations are totally random, such as our proxy signature for MQ-based signature shown in section ‘‘Proxy-MQ: our MQ-based proxy signature scheme,’’ and the proxy signature in Tang and Xu.<sup>24</sup> In some cases of MPKCs, we need to choose some special linear transformations, for example, in section ‘‘Practical implementations for these two schemes,’’ we will show how to choose the linear transformations in the practical implementation of our proposed proxy schemes for UOV and Rainbow.

**ProxySign:** Suppose  $M$  is the message to be signed. This algorithm generates a proxy signature on the message  $M$  by user  $B$ .

User  $B$  applies  $L_{1p}$ ,  $L_{2p}$ , and the central map  $F_p$  to the basic MPKC signature algorithm described in Algorithm 1 to generate the signature  $\sigma$  on  $M$

$$\sigma = \text{Sign}(\mathcal{H}(M), sk_p) \quad (9)$$

Then the proxy signature on message  $M$  by user  $B$  is  $(\sigma, (w, \theta, P'_A, \sigma_{prx}))$ .

**ProxyVerify:** On input  $(M, \sigma, (w, \theta, P'_A, \sigma_{prx}))$ , anyone can verify the validity of the proxy signature by executing this algorithm. This algorithm includes the following steps:

1. Check the validity of  $\theta$  on  $w$  by running  $\text{Verify}$  with  $pk_A$ :  $\text{Verify}((w, P'_A), \theta, pk_A) \stackrel{?}{=} \text{true}$ . If it is true, goto the next step. Otherwise, output 0.
2. Check the validity of  $\sigma_{prx}$  on  $(w, \theta, P'_A)$  by running  $\text{Verify}$  with  $pk_B$ :  $\text{Verify}((w, \theta, P'_A), \sigma_{prx}, pk_B) \stackrel{?}{=} \text{true}$ . If it is true, goto the next step. Otherwise, output 0.
3. Check the validity of  $\sigma$  on message  $M$  by running  $\text{Verify}$  with  $P'_A$ :  $\text{Verify}(M, \sigma, P'_A) \stackrel{?}{=} \text{true}$ . If it is true, output 1. Otherwise, output 0.

The verifier accepts the proxy signature if and only if the three conditions of **ProxySign** are all true. Otherwise, the verifier rejects the proxy signature.

## Security analysis of the general proxy signature scheme

**Theorem 1.** If the basic MPKC signature scheme is  $(\epsilon, t, q_s)$  secure, then the general proxy signature scheme is  $(\epsilon', t', q'_d, q'_s)$  secure, where  $\epsilon \geq \epsilon'/2q'_d$ ,  $t = t'$  and  $q_s = q'_s + q'_d$ .

**Proof.** Let  $\mathcal{A}$  be an adversary who can  $(\epsilon', t', q'_d, q'_s)$  break our proxy signature scheme, then there exists an

attacker  $\mathcal{C}$  who can  $(\epsilon, t, q_s)$  break the corresponding MPKC signature scheme using  $\mathcal{A}$ . Assume that  $\mathcal{C}$  receives a random public key  $pk' = P'$  of MPKC signature scheme and has the right to access to an MPKC signing oracle  $Oracle_{sig}(m, sk)$ . Before beginning the security game, the attacker  $\mathcal{C}$  flips a uniform coin  $c$ . The result of  $c$  is hidden from  $\mathcal{A}$ , unless the security game aborts. If  $c = 0$ ,  $\mathcal{C}$  sets  $pk^* = pk'$ , and  $sk^* = \emptyset$ , where  $\emptyset$  means empty set. Otherwise,  $\mathcal{C}$  generates a fresh key pair  $(pk^*, sk^*) \leftarrow \text{KeyGen}$  where  $pk^* = (P^*)$ , and chooses  $i^* \in \{1, 2, \dots, q'_d\}$ .

As the challenger in the security game,  $\mathcal{C}$  will maintain three lists  $OSList$ ,  $delList(w)$ ,  $pskList(w)$ . Here, the  $delList$  list stores the intermediate result which will be considered in the following. Furthermore,  $\mathcal{A}$  is allowed to make  $q'_s$  ordinary signature queries and  $q'_d$  delegation queries which  $\mathcal{C}$  will answer in the security game as follows:

- *Ordinary signature.* On input  $m$  from  $\mathcal{A}$ , if  $c = 0$ ,  $\mathcal{C}$  simply makes query to the MPKC signing oracle and obtains an answer  $\sigma$ ; if  $c = 1$ ,  $\mathcal{C}$  generates a signature  $\sigma$  by running  $\text{Sign}(m, sk^*)$ .  $\mathcal{C}$  adds  $(m, \sigma)$  to the  $OSList$  list and sends  $\sigma$  to  $\mathcal{A}$ .
- *Delegation to  $u^*$ .*  $\mathcal{A}$  transmits a delegation message  $(w, \theta, L'_{1d}, L'_{2d}, P'_d)$  where  $w = (pk_d, pk^*, t)$ .  $\mathcal{C}$  verifies whether both  $P'_d = L'_{1d} \circ P_d \circ L'_{2d}$  and  $\theta = \text{Sign}(\mathcal{H}(w||P'_d), sk_d)$  are correct. If  $c = 0$ ,  $\mathcal{C}$  chooses randomly two invertible affine transformations  $L'_1$  and  $L'_2$ , and computes  $L_{1p} = L'_{1d} \circ L_1 \circ L'^{-1}_{1d}$ ,  $L_{2p} = L'^{-1}_{2d} \circ L_2$ , and  $F_{pu^*} = L'_1 P_d \circ L'_2$ . Let  $pk_p = P'_d$  and  $sk_p = (L_{1p}, L_{2p}, F_{pu^*})$ . makes a query to the MPKC signing oracle for  $(w, \theta, P'_d, pk_p)$  and obtains a signature  $\sigma_{prx}$ . In this case,  $\sigma_{prx}$  would be added to the  $OSList$  list. If  $c = 1$  and this is not the  $i^*$ th query,  $\mathcal{C}$  similarly chooses randomly two invertible affine transformations  $L_1$  and  $L_2$ , and computes  $L_{1p} = L'_{1d} \circ L_1 \circ L'^{-1}_{1d}$ ,  $L_{2p} = L'^{-1}_{2d} \circ L_2$ , and  $F_{pu^*} = L_1 \circ P_d \circ L_2$ . Let  $pk_p = P'_d$  and  $sk_p = (L_{1p}, L_{2p}, F_{pu^*})$ . Then  $\mathcal{C}$  runs  $\sigma_{prx} = \text{Sign}(\mathcal{H}(w||\theta||pk_p), sk^*)$ . If  $c = 1$  and this is the  $i^*$ th query,  $\mathcal{C}$  directly lets  $pk_p = pk'$ ,  $sk_p = \phi$  and runs  $\sigma_{prx} = \text{Sign}(\mathcal{H}(w||\theta||pk_p), sk^*)$ . Finally,  $\mathcal{C}$  stores  $(w, \theta, P'_d)$  and  $(w, sk_p)$  to the  $delList(w)$  and  $pskList(w)$ , respectively.
- *Delegation from  $u^*$ .*
  1. *Delegation of  $sk^*$ .*  $\mathcal{A}$  submits  $w$  to  $\mathcal{C}$ , where  $w = (pk^*, pk_d, t)$ .  $\mathcal{C}$  chooses randomly two invertible affine transformations  $(L_{1d'}, L_{2d'})$  and computes  $P^{*'} = L_{1d'} \circ P^* \circ L_{2d'}$ . If  $c = 0$ , then  $\mathcal{C}$  makes a query to the MPKC signing oracle and obtains a signature  $\theta$  on  $w||P^{*'}$ . The  $\theta$  later is added to the  $OSList$  list by  $\mathcal{C}$ . If  $c = 1$ , then  $\mathcal{C}$  generates  $\theta$  by running  $\theta \leftarrow \text{Sign}(\mathcal{H}(w||P^{*'}), sk^*)$  and sends the delegation message  $(w, \theta, L_{1d'}, L_{2d'}, P^{*'})$  to

$\mathcal{A}$ . Of course,  $\mathcal{C}$  adds  $(w, \theta, P^{*'})$  to the  $delList(w)$  list.

2. *Self-delegation.*  $\mathcal{C}$  interacts with itself with  $w = (pk^*, pk^*, t)$  which submitted by  $\mathcal{A}$ . If  $c = 0$  or  $c = 1$  and this is not the  $i^*$ th query,  $\mathcal{C}$  chooses randomly two invertible affine transformations  $(L_{1p}, L_{2p})$ , computes  $P^{*'} = L_{1p} \circ P^* \circ L_{2p}$ , makes a query to the MPKC signing oracle, and obtains a signature  $\theta$  on  $w||P^{*'}$  with  $P^*$ . Then,  $\mathcal{C}$  also makes a query to the MPKC signing oracle for  $(w, \theta, P^{*'})$  and obtains a signature  $\sigma_{prx}$ . If  $c = 1$  and this is the  $i^*$ th query,  $\mathcal{C}$  directly lets  $pk_p = pk'$  and computes  $\sigma_{prx} = \text{Sign}(\mathcal{H}(w||\theta||pk_p), sk^*)$ . Finally,  $\mathcal{C}$  adds  $(w, \theta, pk_p)$  to the  $delList(w)$  and  $(w, sk_p)$  to the  $pskList(w)$ . If the  $\theta$  is obtained by the MPKC signing oracle,  $\mathcal{C}$  also adds it to the  $OSList$  list.

- *Proxy signature.* Once receiving  $(m, (w, \theta))$  submitted by  $\mathcal{A}$ ,  $\mathcal{C}$  finds the relevant information with  $w$  from  $delList(w)$  and  $pskList(w)$ .  $\mathcal{C}$  parses  $pk_p$  and the proxy key as  $sk_p$ . Then,  $\mathcal{C}$  makes a query to the MPKC signing oracle for  $m$  and obtains a signature  $\sigma$  if  $c = 0$ . Otherwise,  $\mathcal{C}$  computes  $\sigma \leftarrow \text{Sign}(\mathcal{H}(m), sk_p)$ . Then  $\mathcal{C}$  sends  $(m, \sigma, (w, \theta, pk_p, \sigma_{prx}))$  to  $\mathcal{A}$ .
- *Proxy key exposure.* On input  $w$  from  $\mathcal{A}$ ,  $\mathcal{C}$  finds relevant information from  $delList(w)$  and  $pskList(w)$  and parses it as  $(sk_p, (w, \theta, pk_p, \sigma_{prx}))$ . If  $sk_p = \phi$ ,  $\mathcal{C}$  aborts the game. Otherwise,  $\mathcal{C}$  returns  $(sk_p, (w, \theta, pk_p, \sigma_{prx}))$  to  $\mathcal{A}$ .

If the above game is not forced to abort by  $\mathcal{C}$ ,  $\mathcal{A}$  will eventually output a forgery. The forgeries are classified into two different cases:

Case 1:  $\mathcal{A}$  forges (1) a valid MPKC signature  $(m, \sigma)$  or (2) a valid proxy signature  $(m, (w, \theta, pk_p, \sigma_{prx}), \sigma)$  which the corresponding public key  $pk_p$  was not generated by  $\mathcal{C}$ , or (3) a valid proxy signature  $(m, (w, \theta, pk_p, \sigma_{prx}), \sigma)$  where  $w$  was not submitted to the ordinary signature query.

Case 2:  $\mathcal{A}$  forges a valid signature which is not in case 1.

In the case  $c = 0$ ,  $\mathcal{C}$  sets  $pk^* = pk'$ . If  $\mathcal{A}$  constructs a valid forgery in case 2,  $\mathcal{C}$  will abort the game. Otherwise, if  $\mathcal{A}$  constructs a valid forgery in case 1, then

- If the forgery is of type 1, that is,  $(m, \sigma)$ , it shows that  $\mathcal{A}$  has not requested a signature on  $m$ . Then,  $\mathcal{C}$  will not have submitted  $m$  to an MPKC signature oracle. That is,  $\sigma$  is a valid forgery of an MPKC signature under the public key  $pk'$ .

- If the forgery is of type 2, that is,  $(m, (w, \theta, pk_p, \sigma_{prx}), \sigma)$  is a valid signature for  $(w, \theta, pk_p)$  under the public key  $pk_p = pk'$ , then  $\mathcal{C}$  will not have submitted  $(w, \theta, pk_p)$  to MPKC signing oracle. Therefore,  $\sigma_{prx}$  will be a valid MPKC signature forgery under the public key  $pk'$ .
- If the forgery is of type 3, that is,  $(m, (w, \theta, pk_p, \sigma_{prx}), \sigma)$  derives that  $\theta$  is a valid forgery for  $w$ , and  $\mathcal{C}$  will therefore not have submitted  $w$  to the signing oracle. Hence  $\theta$  is a valid forgery of an MPKC signature under the public key  $pk'$ .

Now, let us consider the case  $c = 1$  where  $\mathcal{C}$  inserts  $pk'$  as a proxy public key. In this case, if the forgery is in case 1, then  $\mathcal{C}$  will abort the game. However, if the forgery is in case 2 which forgery  $(m, (w, \theta, pk_p, \sigma_{prx}), \sigma)$  where  $pk_p = pk'$ , then  $\mathcal{C}$  outputs  $(m, \sigma)$  as a valid forgery for signature scheme. Otherwise,  $\mathcal{C}$  aborts. Note that if  $\mathcal{A}$  constructs such a forgery, then  $\mathcal{A}$  will not have queried the proxy key  $(w, \theta, pk_p, \sigma_{prx})$  with  $pk_p$ .

We define the following events associated with the above security game:  $E_1$  be the event that  $\mathcal{A}$  constructs a forgery in case 1,  $E_2$  be the event that  $\mathcal{A}$  constructs a forgery in case 2, and  $E_3$  denotes that  $\mathcal{A}$  guesses the correct value of  $i^*$  in a forgery for case 2. The success probability of  $\mathcal{A}$  is  $Pr[E_1] + Pr[E_2]$ . Then the success probability of  $\mathcal{C}$  can be

$$\begin{aligned} \epsilon &= Pr[c = 0 \wedge E_1] + Pr[c = 1 \wedge E_2 \wedge E_3] \\ &= 1/2 \cdot Pr[E_1] + Pr[E_3 | c = 1 \wedge E_2] \cdot Pr[c = 1 | E_2] \\ &= 1/2 \cdot Pr[E_1] + 1/q'_d \cdot 1/2 \cdot Pr[E_2] \\ &\geq \frac{\epsilon'}{2q'_d}. \end{aligned}$$

**Remark 2.** Note that the above proof is only a heuristic security proof, since the underlying signature schemes in the area of MPKC are mostly not provable secure, more discussion will be done next, and we propose the additional analysis in section ‘‘Practical implementations for these two schemes’’ for our practical implementation.

Furthermore, we can obtain that anyone can determine the proxy signer by the verification of the warrant  $w$  and the signature  $\sigma_{prx}$ . Then, the proxy signer is required to sign  $\theta$  and the public key of the proxy signature. Under the assumption that the underlying signature scheme is secure, we can conclude that any proxy signer cannot deny the proxy signature he or she created due to the existence of  $\sigma_{prx}$ . At the same time, the private key of the original signer is only directly used to sign the warrant  $w$ . And no one can obtain the private key of the original signer from the proxy key because of

the selected random transformations in Delegate. The above discussions show that our scheme meets all the security properties of a proxy signature scheme.

## The proposed MPKC proxy signature schemes

In this section, we will propose three proxy schemes based on three well-known MPKC schemes: UOV,<sup>12</sup> Rainbow<sup>22</sup> and MQ-based scheme.<sup>16</sup>

### Proxy-UOV: proxy scheme based on UOV

Now we describe the process of our proxy scheme based on UOV using our general construction.

**Setup:** Let  $n$  and  $m$  be two positive integers,  $k$  is a finite field and all the arithmetic operations hereafter are over this finite field.  $\mathcal{H}: \{0, 1\}^* \rightarrow k^n$  is a cryptographic hash function.

**KeyGen:** This algorithm generates the public key and the private key of a user. The detail process is the same as that of the key generation of an MPKC signature scheme. After this algorithm, we set the private key of user  $A$  is  $sk_A = (F_A, T_A)$ , and the public key  $pk_A = P_A$ , where  $P_A = F_A \circ T_A$ . Similarly, the private key and public key of user  $B$  are:  $sk_B = (F_B, T_B)$ ,  $pk_B = P_B$ , respectively.

**Delegate:**  $A$  randomly chooses a bijective affine transformation  $T$ , then computes  $T'_A = T \circ T_A$ ,  $F'_A = F_A \circ T$  and  $P'_A = F'_A \circ T'_A$ .

The affine  $T$  should be kept secret by  $A$ .  $A$  sends  $(T'_A, F'_A, P'_A)$  and the warrant  $(w, \theta)$  to  $B$  through an authenticated channel, where  $w = (pk_A, pk_B, t)$ ,  $t$  is a time period which denotes that  $w$  is valid in time  $t$  and  $\theta$  is a signature on  $w$  generated by  $A$  using our proposed signing algorithm, that is,  $\theta = \text{Sign}(\mathcal{H}(w), sk_A)$ .

**ProxyKeyGen:** After receiving  $(T'_A, F'_A, P'_A, w, \theta)$ ,  $B$  first checks the validity of  $P'_A, \theta$  by checking if  $P'_A = F'_A \circ T'_A$  and  $\text{Verify}(w, \theta, pk_A) = 1$ . Then  $B$  selects a random bijective affine transformation  $T'$  and computes  $T_p = T'^{-1} \circ T'_A$ , and  $F_p = F'_A \circ T'$ . Let  $sk_p = (F_p, T_p)$ , and  $pk_p = P'_A$ . Then  $sk_p$  is a private key for ordinary signature, and the corresponding public key is  $pk_p$ , that is because the following equality holds

$$\begin{aligned} F_p \circ T_p &= F'_A \circ T' \circ T'^{-1} \circ T'_A \\ &= F'_A \circ T \circ T_A = F'_A \circ T'_A = P'_A \end{aligned} \quad (10)$$

Then  $B$  computes a signature  $\sigma_{prx}$  by running

$$\sigma_{prx} = \text{Sign}((\mathcal{H}(w, \theta), P'_A), sk_B) \quad (11)$$



and sets  $sk_p$  as the proxy signing key that  $B$  uses to generate proxy signatures on behalf of  $A$  and sets  $pk_p$  and  $(w, \theta, P'_A, \sigma_{prx})$  as the proxy verifying key.

**ProxySign:** Suppose  $M$  is the message to be signed. This algorithm generates a proxy signature on the message  $M$  by user  $B$ .

User  $B$  applies  $T_p$  and the central map  $F_p$  to the basic MPKC signature algorithm described in Algorithm 1 to generate the signature  $\sigma$  on  $M$

$$\sigma = \text{Sign}(\mathcal{H}(M), sk_p) \quad (12)$$

Then the proxy signature on message  $M$  by user  $B$  is  $(\sigma, (w, \theta, P'_A, \sigma_{prx}))$ .

**ProxyVerify:** On input  $(M, \sigma, (w, \theta, P'_A, \sigma_{prx}))$ , anyone can verify the validity of the proxy signature by executing this algorithm. This algorithm includes the following steps:

1. Check the validity of  $\theta$  on  $w$  by running **Verify** with  $pk_A$ :  $\text{Verify}((w, P'_A), \theta, pk_A) \stackrel{?}{=} \text{true}$ . If it is true, go to the next step. Otherwise, output 0.
2. Check the validity of  $\sigma_{prx}$  on  $(w, \theta, P'_A)$  by running **Verify** with  $pk_B$ :  $\text{Verify}((w, \theta, P'_A), \sigma_{prx}, pk_B) \stackrel{?}{=} \text{true}$ . If it is true, go to the next step. Otherwise, output 0.
3. Check the validity of  $\sigma$  on message  $M$  by running **Verify** with  $P'_A$ :  $\text{Verify}(M, \sigma, P'_A) \stackrel{?}{=} \text{true}$ . If it is true, output 1. Otherwise, output 0.

The verifier accepts the proxy signature if and only if the three conditions of **ProxySign** are all true. Otherwise, the verifier rejects the proxy signature.

### Proxy-Rainbow: proxy scheme based on Rainbow

Since the main difference of this proxy signature schemes lies on the **Delegate** step and **ProxyKeyGen** process compared to the general construction, we just only describe the processes **Delegate** step and **ProxyKeyGen**.

**Setup:** Let  $n$  and  $m$  be two positive integers,  $k$  is a finite field, and all the arithmetic operations hereafter are over this finite field.  $\mathcal{H}: \{0, 1\}^* \rightarrow k^n$  is a cryptographic hash function.

**KeyGen:** This algorithm generates the public key and the private key of a user. The detail process is the same as that of the key generation of an MPKC signature scheme. After this algorithm, we set the private key of user  $A$  is  $sk_A = (L_{1A}, L_{2A}, F_A)$ , and the public key  $pk_A = P_A$ , where  $P_A = L_{1A} \circ F_A \circ L_{2A}$ .

Similarly, the private key and public key of user  $B$  are  $sk_B = (L_{1B}, L_{2B}, F_B)$  and  $pk_B = P_B$ , respectively.

**Delegate:**  $A$  randomly chooses two invertible affine transformations  $L'_1$  and  $L'_2$  respectively, then computes  $L_1 = L_{1A} \circ L'_1$ ,  $L_2 = L'_2 \circ L_{2A}$  and  $F'_A = L'_1 \circ F_A \circ L'_2$ .  $P'_A = L_1 \circ F'_A \circ L_2$ . The affine  $L'_1$  and  $L'_2$  should be kept secret by  $A$ .  $A$  sends  $(L_1, L_2, F'_A, P'_A)$  and the warrant  $(w, \theta)$  to  $B$  through an authenticated channel, where  $w = (pk_A, pk_B, t)$ ,  $t$  is a time period which denotes that  $w$  is valid in time  $t$  and  $\theta$  is a signature on  $w$  generated by  $A$  using Rainbow signing algorithm, that is,  $\theta = \text{Sign}(w, sk_A)$ .

**ProxyKeyGen:** After receiving  $(L_1, L_2, F'_A, P'_A, w, \theta)$ ,  $B$  randomly chooses two invertible affine transformations  $L''_1$  and  $L''_2$ , respectively, and computes  $L_{1p} = L_1 \circ L''_1$ ,  $L_{2p} = L''_2 \circ L_2$ , and  $F_p = L''_1 \circ F'_A \circ L''_2$ . Let  $sk_p = (L_{1p}, F_p, L_{2p})$ , and  $pk_p = P'_A$ . Then  $sk_p$  is a private key for ordinary signature, and the corresponding public key is  $pk_p$ , that is, because the following equality holds

$$\begin{aligned} L_{1p} \circ F_p \circ L_{2p} &= L_1 \circ L''_1 \circ L''_2 \circ F'_A \circ L''_1 \circ L''_2 \\ &= L_1 \circ L''_1 \circ L''_2 \circ F'_A \circ L_2 = P'_A \end{aligned} \quad (13)$$

Then  $B$  computes a signature  $\sigma_{prx}$  by running

$$\sigma_{prx} = \text{Sign}((w, \theta, pk_p), sk_p) \quad (14)$$

and sets  $sk_p$  as the proxy signing key that  $B$  uses to generate proxy signatures on behalf of  $A$ , and sets  $pk_p$  and  $(w, \theta, pk_p, \sigma_{prx})$  as the proxy verifying key.

### Practical implementations for these two schemes

In Petzoldt et al.,<sup>33</sup> the result indicates that  $q = 2^8, m = 26, v = 52$  has security level higher than  $2^{80}$  for UOV scheme, where  $q$  is the order of the finite field,  $m$  is the number of polynomials and is equal to the number of Oil variate, and  $v$  is the number of Vinegar variate. We will choose this parameter set.

Next, also the extremely important setting of our construction, we should choose appropriate affine transformations that could preserve the special structure of UOV scheme. Specifically, after composing an affine transformation, the polynomials in the new central map, the public key should still stay in the form of Oil-Vinegar polynomials. If we represent a UOV scheme's central polynomial by its corresponding matrix, then the matrices of the polynomials in central map should be in the form of following.

Next, also the extremely important setting of our construction, we should choose appropriate affine transformations that could preserve the special structure of UOV scheme. Specifically, after composing an affine transformation, the polynomials in the new

central map, the public key should still stay in the form of Oil–Vinegar polynomials. If we represent a UOV scheme’s central polynomial by its corresponding matrix, the Vinegar variables are denoted by its first  $v = 52$  variables. Then the matrices of the polynomials in central map should be in the form of Figure 1. In Figure 1, the gray areas represent the random entries while blank areas denote zero entries. The rest part of this article follows the same rules. Thereby, our problem is transformed to choose an affine transformation that could keep the shape of the above matrix of central equation. To achieve that goal, we could pick the invertible affine transformations of the form as shown in Figure 2.

Once  $T$  and  $T'$  are choosing in this form, we will get that  $F_p = F_A \circ T \circ T'$ , which means that the matrices form of central map  $F_p$  is

$$F_p = \begin{bmatrix} *_{v \times v} & *_{v \times o} \\ *_{o \times v} & 0_{o \times o} \end{bmatrix} \begin{bmatrix} *_{v \times v} & 0_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix} \begin{bmatrix} *_{v \times v} & *_{o \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix} \\ = \begin{bmatrix} *_{v \times v} & *_{v \times o} \\ *_{o \times v} & 0_{o \times o} \end{bmatrix}$$

Thus, this will make sure that the map  $F_p$  can still be easily inverted.

Currently, the Rainbow prevails now is two-layer based and layer structure (18,12,12) and  $GF(2^8)$  is enough to resist all the attacks mentioned above (security level is greater than  $2^{80}$ ).<sup>34</sup> Thereby, we plan to use this parameter set of Rainbow for our Proxy-Rainbow scheme. The Rainbow’s corresponding matrices have the following forms in Figure 3.

Moreover, to keep the multi-layer structure of central equation, the structure of the left affine transformation  $L_1$  and  $L'_1$  should have the shapes as shown below. By choosing these structures, we can make sure that the map  $F_p$  of Proxy-Rainbow can still be easily inverted (Figures 4 and 5).

### Proxy-MQ: our MQ-based proxy signature scheme

Using our general method to propose a proxy signature scheme based on the basic MQ-based signature scheme described in section “The general construction of MPKC proxy signature scheme,” we need to make some changes in the key generation phase KeyGen.

Below is the full description of our proxy signature for the MQ scheme.

**Setup:** Let  $n$  and  $m$  be two positive integers,  $\mathbb{F}_q$  is a finite field and all the arithmetic operations hereafter are over this finite field.  $\mathcal{H}: \{0, 1\}^* \rightarrow \mathbb{F}_q^n$  is a cryptographic hash function.

**KeyGen:** Let  $\mathbf{e}$  be a vector from  $\mathbb{F}_q^n$  that every element is randomly chosen in  $k$  (i.e.  $\mathbf{e} = \{1, \dots, 1\}^T$ ,

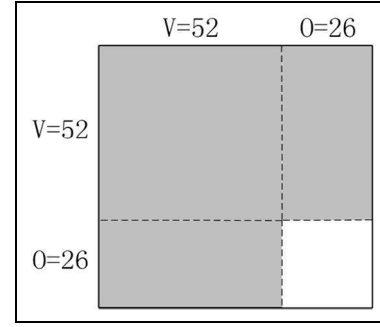


Figure 1. Oil–Vinegar scheme corresponding matrix.

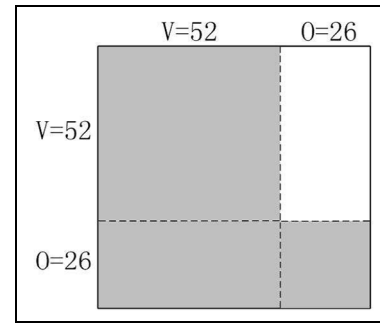


Figure 2. The affine transformation form for Proxy-UOV.

the superscript T denotes transposition), to get  $A$ s private key,  $A$  first randomly chooses a bijective affine transformation  $T_A$ , and its private key  $\mathbf{s}_A$  is calculated by  $\mathbf{s}_A = T_A(\mathbf{e})$ , then the corresponding public key  $pk_A$  is  $(F_A, \mathbf{v}_A)$  that satisfies  $\mathbf{v}_A = F_A(\mathbf{s}_A)$ . Similarly,  $B$ ’s private key  $sk_B$  is calculated by  $\mathbf{s}_B = T_B(\mathbf{e})$  where  $T_B$  is a randomly bijective affine transformation, and the corresponding public key  $pk_B$  is  $(F_B, \mathbf{v}_B)$  that satisfies  $\mathbf{v}_B = F_B(\mathbf{s}_B)$ . Then  $\mathbf{e}$ ,  $pk_A = (F_A, \mathbf{v}_A)$  and  $pk_B = (F_B, \mathbf{v}_B)$ , are published to the public bulletin board.

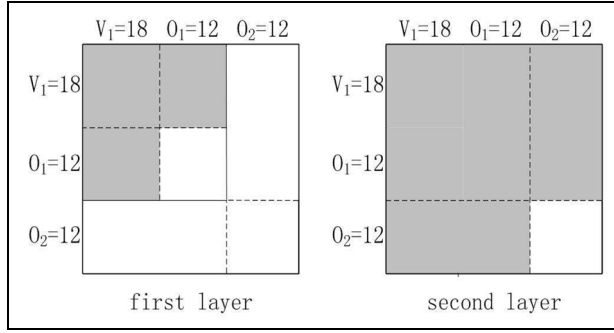
**Delegate:** Let  $\mathbf{s}_A = T_A(\mathbf{e})$ ,  $A$  randomly chooses a bijective affine transformation  $T$ , then computes

$$T'_A = T \circ T_A$$

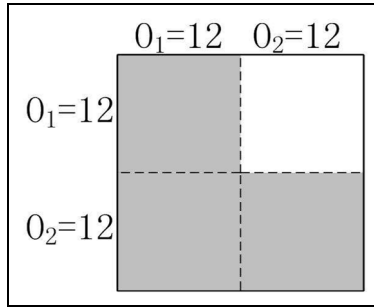
$$F'_A = F_A \circ T$$

$$\mathbf{v}'_A = F'_A \circ T'_A(\mathbf{e})$$

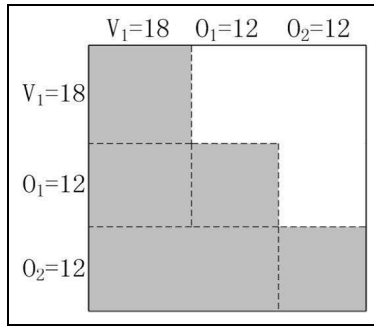
The affine  $T$  should be kept secret by  $A$ .  $A$  sends  $(T'_A, F'_A, \mathbf{v}'_A)$  and the warrant  $(w, \theta)$  to  $B$  through an authenticated channel, where  $w = (pk_A, pk_B, t)$  and  $t$  is a time period which denotes that  $w$  is valid in time  $t$ , and  $\theta$  is a signature on  $w$  generated by  $A$  using our proposed signing algorithm, that is,  $\theta = \text{Sign}(w, sk_A)$ .



**Figure 3.** Rainbow scheme corresponding matrix.



**Figure 4.** The left affine transformation form for Proxy-Rainbow.



**Figure 5.** The right affine transformation form for Proxy-Rainbow.

**ProxyKeyGen:** Let  $\mathbf{s}_B = T_B(\mathbf{e})$ , after receiving  $(T'_A, F'_A, \mathbf{v}'_A, w, \theta)$ ,  $B$  selects a random bijective affine transformation  $T'$  and computes

$$\begin{aligned} \mathbf{s}_p &= T'^{-1} \circ T'_A(\mathbf{e}) \\ F_p &= F'_A \circ T' \\ \mathbf{v}_p &= \mathbf{v}'_A \end{aligned}$$

Let  $pk_p = (F_p, \mathbf{v}_p)$ ,  $sk_p = \mathbf{s}_p$ . Then  $sk_p$  is a private key for ordinary signature, and the corresponding public key is  $pk_p$ , that is because the following equality holds

$$\begin{aligned} F_p(\mathbf{s}_p) &= F'_A \circ T' \circ T'^{-1} \circ T'_A(\mathbf{e}) \\ &= F'_A \circ T \circ T'_A(\mathbf{e}) \\ &= \mathbf{v}'_A = \mathbf{v}_p \end{aligned}$$

Then  $B$  computes a signature  $\sigma_{prx}$  by running

$$\sigma_{prx} = \text{Sign}((w, \theta, pk_p), sk_B) \quad (15)$$

and sets  $sk_p$  as the proxy signing key that  $B$  uses to generate proxy signatures on behalf of  $A$ , and sets  $pk_p$  and  $(w, \theta, pk_p, \sigma_{prx})$  as the proxy verifying key.

**ProxySign:** Suppose  $M$  is the message to be signed. This algorithm generates a proxy signature on the message  $M$  by user  $B$ .

User  $B$  applies  $L_{1p}$ ,  $L_{2p}$  and the central map  $F_p$  to the basic MPKC signature algorithm described in Algorithm 1 to generate the signature  $\sigma$  on  $M$

$$\sigma = \text{Sign}(\mathcal{H}(M), sk_p) \quad (16)$$

Then the proxy signature on message  $M$  by user  $B$  is  $(\sigma, (w, \theta, P'_A, \sigma_{prx}))$ .

**ProxyVerify:** On input  $(M, \sigma, (w, \theta, P'_A, \sigma_{prx}))$ , anyone can verify the validity of the proxy signature by executing this algorithm. This algorithm includes the following steps:

1. Check the validity of  $\theta$  on  $w$  by running **Verify** with  $pk_A$ :  $\text{Verify}((w, P'_A), \theta, pk_A) \stackrel{?}{=} \text{true}$ . If it is true, goto the next step. Otherwise, output 0.
2. Check the validity of  $\sigma_{prx}$  on  $(w, \theta, P'_A)$  by running **Verify** with  $pk_B$ :  $\text{Verify}((w, \theta, P'_A), \sigma_{prx}, pk_B) \stackrel{?}{=} \text{true}$ . If it is true, goto the next step. Otherwise, output 0.
3. Check the validity of  $\sigma$  on message  $M$  by running **Verify** with  $P'_A$ :  $\text{Verify}(M, \sigma, P'_A) \stackrel{?}{=} \text{true}$ . If it is true, output 1. Otherwise, output 0.

The verifier accepts the proxy signature if and only if the three conditions of **ProxySign** are all true. Otherwise, the verifier rejects the proxy signature.

### Practical implementation for Proxy-MQ

After the changes, it is easy to see that we modify the randomly selected vector into an invertible transformation and a public known random vector, so the scheme is based not only on MQ problem, but also on IP problem. In contrast to Sakumoto et al.,<sup>16</sup> we suggest to use a determined system for the MQ-based signature scheme (i.e.  $m = n$ ). The reason for this is that a greater number of variables does not increase the security of this scheme.<sup>35</sup> And we propose for the scheme the parameters  $k = GF(2), (m, n) = (84, 84), r = 193$ , where  $r$  is

the number of the signer gathers the commitments in our modified MQ signature scheme.

### A toy example and deployment in real distributed scenario

To illustrate how our proxy signature scheme works, we propose a toy example for Proxy-UOV in this section (others are similar). The following example comes from our running test on Magma.<sup>36</sup>

In our toy example, we choose the parameter of Proxy-UOV as  $q = 2^2, m = 2, n = 4, o = 2, v = 2$ .

First, we set the private key of user  $A$  is  $sk_A = (F_A, T_A)$ , the public key  $pk_A = P_A$ , the private key and public key of user  $B$  are  $sk_B = (F_B, T_B)$  and  $pk_B = P_B$ , respectively

$$F_A = \begin{pmatrix} x_1x_3 + \alpha^2x_2x_4 + x_3^2 + x_4^2 \\ \alpha^2x_1x_4 + x_3x_4 + x_4^2 \end{pmatrix}$$

$$T_A(x_1, x_2, x_3, x_4) = \begin{pmatrix} 1 & \alpha & \alpha^2 & 0 \\ \alpha^2 & \alpha^2 & \alpha & 1 \\ \alpha^2 & 0 & \alpha^2 & \alpha \\ \alpha & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$P_A = \begin{pmatrix} x_1^2 + \alpha x_1x_2 + x_1x_3 + x_1x_4 + \alpha^2x_2x_3 \\ + x_2x_4 + \alpha^2x_3x_4 + x_4^2 \\ \alpha^2x_1^2 + \alpha x_1x_2 + \alpha x_1x_3 + \alpha^2x_1x_4 + x_2x_3 \\ + x_2x_4 + \alpha^2x_3x_4 + \alpha^2x_4^2 \end{pmatrix}$$

$$F_B = \begin{pmatrix} \alpha x_1x_3 + \alpha x_2x_4 + x_2x_3 + \alpha x_3^2 + x_3x_4 + x_4^2 \\ x_1x_4 + \alpha^2x_2x_3 + x_3x_4 \end{pmatrix}$$

$$T_B(x_1, x_2, x_3, x_4) = \begin{pmatrix} \alpha & 1 & \alpha & \alpha \\ \alpha^2 & \alpha & 0 & 1 \\ 0 & \alpha & 0 & \alpha \\ 1 & \alpha^2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$P_B = \begin{pmatrix} x_1^2 + \alpha x_1x_2 + \alpha^2x_1x_3 + \alpha x_1x_4 + \alpha^2x_2^2 \\ + \alpha x_2x_3 + \alpha x_3x_4 + x_4^2 \\ \alpha x_1^2 + x_1x_2 + \alpha x_1x_3 + x_1x_4 + x_2x_3 \\ + \alpha x_3x_4 + x_4^2 \end{pmatrix}$$

where  $P_A = F_A \circ T_A$ ,  $P_B = F_B \circ T_B$ .

Delegate:  $A$  randomly chooses a bijective affine transformation  $T$  as follows

$$T(x_1, x_2, x_3, x_4) = \begin{pmatrix} 0 & \alpha & 0 & 0 \\ \alpha & \alpha & 1 & 1 \\ 0 & 0 & \alpha^2 & \alpha \\ 0 & 0 & \alpha & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Then  $A$  computes  $T'_A = T \circ T_A$ ,  $F'_A = F_A \circ T'_A$ , and  $P'_A = F'_A \circ T'_A$ . The results are

$$T'_A(x_1, x_2, x_3, x_4) = \begin{pmatrix} 1 & 1 & \alpha^2 & \alpha \\ \alpha & \alpha & 0 & 1 \\ 1 & 0 & 0 & \alpha^2 \\ 1 & 0 & 1 & \alpha^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$F'_A = \begin{pmatrix} \alpha x_1x_3 + \alpha^2x_2x_3 + \alpha^2x_2x_4 + x_3x_4 + \alpha^2x_4^2 \\ \alpha x_2x_3 + \alpha x_3^2 + \alpha^2x_3x_4 \end{pmatrix}$$

$$P'_A = \begin{pmatrix} \alpha x_1x_2 + x_1x_3 + \alpha x_1x_4 + x_2x_3 + x_2x_4 \\ + \alpha^2x_3^2 + \alpha^2x_3x_4 + x_4^2 \\ \alpha x_1^2 + \alpha^2x_1x_2 + \alpha^2x_1x_3 + \alpha x_3x_4 + \alpha^2x_4^2 \end{pmatrix}$$

In addition,  $A$  generates a warrant  $w = (pk_A, pk_B, t)$  (assume that  $\mathcal{H}(w) = (0, \alpha)$ ), signs it using the regular UOV signing algorithm, and gets  $\theta = \text{Sign}(\mathcal{H}(w), sk_A) = (0, 0, \alpha, 1)$ ,

Finally,  $A$  sends  $(T'_A, F'_A, P'_A)$  and the warrant  $(w, \theta)$  to  $B$  through an authenticated channel.

ProxyKeyGen: After receiving  $(T'_A, F'_A, P'_A, w, \theta)$ ,  $B$  first checks the validity of  $P'_A, \theta$  by checking  $P'_A = F'_A \circ T'_A$  and  $\text{Verify}(w, \theta, pk_A) = P_A(0, 0, \alpha, 1) = (0, \alpha)$ . Then  $B$  selects a random bijective affine transformation  $T'$  and computes  $T_p = T'^{-1} \circ T'_A$ , and  $F_p = F'_A \circ T'$ , where

$$T'(x_1, x_2, x_3, x_4) = \begin{pmatrix} 1 & \alpha & \alpha^2 & 1 \\ \alpha^2 & \alpha^2 & 0 & \alpha^2 \\ 0 & 0 & 0 & \alpha \\ 0 & 0 & \alpha & \alpha^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$T_p(x_1, x_2, x_3, x_4) = \begin{pmatrix} 1 & 0 & \alpha & 1 \\ 1 & \alpha^2 & \alpha & 1 \\ \alpha & 0 & \alpha^2 & 1 \\ \alpha^2 & 0 & 0 & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$F_p = \begin{pmatrix} \alpha x_1x_3 + x_1x_4 + \alpha^2x_2x_4 + \alpha x_3^2 + \alpha x_3x_4 + x_4^2 \\ \alpha x_1x_4 + \alpha x_2x_4 + \alpha x_3x_4 \end{pmatrix}$$

$$P'_A = \begin{pmatrix} \alpha x_1x_2 + x_1x_3 + \alpha x_1x_4 + x_2x_3 + x_2x_4 \\ + \alpha^2x_3^2 + \alpha^2x_3x_4 + x_4^2 \\ \alpha x_1^2 + \alpha^2x_1x_2 + \alpha^2x_1x_3 + \alpha x_3x_4 + \alpha^2x_4^2 \end{pmatrix}$$

Let  $sk_p = (F_p, T_p)$ , and  $pk_p = P'_A$ . Then  $sk_p$  is a private key for ordinary signature, and the corresponding public key is  $pk_p$ .

Assume the hash value of  $(w, \theta, P'_A)$  is  $(\alpha, \alpha)$ ,  $B$  computes a signature  $\sigma_{prx}$  by running

$$\sigma_{prx} = \text{Sign}(\mathcal{H}(w, \theta, P'_A), sk_B) = (1, 1, 1, \alpha)$$

and sets  $sk_p$  as the proxy signing key that  $B$  uses to generate proxy signatures on behalf of  $A$ , and sets  $pk_p$  and  $(w, \theta, P'_A, \sigma_{prx})$  as the proxy verifying key.

ProxySign: Suppose  $\mathcal{H}(M) = (0, \alpha^2)$  is the message to be signed.

**Table 1.** Computing complexity requirements by ours and compared with other schemes.

	Proxy-UOV	Proxy-Rainbow	Proxy-MQ
Initialization	$\mathcal{O}(2n^2 + 2mn^2)$	$\mathcal{O}(2m^2 + 2n^2 + 2mn^2)$	$\mathcal{O}(2n^2 + 2mn^2)$
Proxy share generation	$\mathcal{O}(3n^3 + 2mn^2 + n^2 + S)$	$\mathcal{O}(3m^3 + 4n^3 + 4mn^2 + 2uS)$	$\mathcal{O}(2n^3 + 3mn^2 + 2round \cdot mn^3)$
Proxy signature generation	$\mathcal{O}(n^3 + n^2 + S)$	$\mathcal{O}(m^3 + n^3 + n^2 + uS)$	$\mathcal{O}(round \cdot mn^3)$
Signature verify	$\mathcal{O}(3n)$	$\mathcal{O}(3n)$	$\mathcal{O}(3round \cdot mn^3)$
	Tang's scheme <sup>24</sup>	Mambo's scheme <sup>30</sup>	HZ scheme <sup>37</sup>
Initialization	$\mathcal{O}(2m^2 + 2n^2 + 2mn^2)$	$2T_p + T_e + T_m \approx \mathcal{O}(2\log(n_r) + n_{rsa}\log(n_r) + n_r^2)$	$2T_p + 5T_e + 2T_m \approx \mathcal{O}(2\log(n_r) + 5n_{rsa}\log(n_r) + 2n_r^2)$
Proxy share generation	$\mathcal{O}(5m^3 + 5n^3 + 5mn^3 + 3qmn^3)$	$T_e + T_m \approx \mathcal{O}(n_r\log(n_r) + n_r^2)$	$6T_e + 4T_m + T_H \approx \mathcal{O}(6n_r\log(n_r) + 4n_r^2 + \log(n_h))$
Proxy signature generation	$\mathcal{O}(qmn^3)$	$T_e \approx \mathcal{O}(n_r\log(n_r))$	$4T_e + 6T_m + 2T_H \approx \mathcal{O}(4n_r\log(n_r) + 6n_r^2 + 2\log(n_h))$
Signature verify	$\mathcal{O}(3qmn^3)$	$2T_e + T_m \approx \mathcal{O}(2n_r\log(n_r) + n_r^2)$	$5T_e + 3T_m + 2T_H \approx \mathcal{O}(5n_r\log(n_r) + 3n_r^2 + 2\log(n_h))$

UOV: Unbalanced Oil and Vinegar; MQ: multivariate quadratic; RSA: Rivest–Shamir–Adleman; HZ: Hu–Zhang.

User  $B$  generates the signature  $\sigma$  on  $M$  using the proxy key  $sk_p$

$$\sigma = \text{Sign}(\mathcal{H}(M), sk_p) = (\alpha^2, 0, \alpha, \alpha)$$

Then the proxy signature on message  $M$  by user  $B$  is  $(\sigma, (w, \theta, P'_A, \sigma_{prx}))$ .

**ProxyVerify:** On input  $(M, \sigma, (w, \theta, P'_A, \sigma_{prx}))$ , a verifier  $V$  can verify the validity of the proxy signature by executing this algorithm. This algorithm includes the following steps:

1. Compute  $P_A(\theta) = (0, \alpha)$  and check whether it is equal to  $\mathcal{H}(w)$ . Since it is true, go to the next step.
2. Compute  $P_B(\sigma_{prx}) = (\alpha, \alpha)$  and check whether it is equal to  $\mathcal{H}(w, \theta, P'_A)$ . Since it is true, go to the next step.
3. Compute  $P'_A(\sigma) = (0, \alpha^2)$  and check whether it is equal to  $\mathcal{H}(M)$ . Since it is also valid, the algorithm outputs 1.

In addition, let us consider a communication scenario in a distributed sensor network and see how we can deploy our proxy scheme into this scenario. To develop a secure communication in a distributed sensor network, sensors digitally sign the data (i.e. the message  $M$ ) and communicate to a server. It is expected that the server must be convinced to the authenticity of the sender (i.e. sensor deployed in the distributed network) and the sender (i.e. sensor) must also be convinced to

the authenticity of the receiver (i.e. server). To develop such authentication method, a designated receiver delegation method is the most promising. In this method, the deploying authority (DA) (server administrator or network developer) delegates its signing power to sensor and also designates the trained professional as a receiver. Thus, DA is original signer, sensor is a proxy signer and the server is designated receiver. As shown in the above toy example, in this scenario, the DA is acting as  $A$ , the sensors are acting as  $B$ , and finally, it can generate a proxy signature on message  $M$  as  $(\sigma, (w, \theta, P'_A, \sigma_{prx}))$ . Finally, the server is acting as a verifier  $V$  and thus can convince to the authenticity of the sensors.

## Performance and comparisons of our proxy signature schemes

The complexity and running time of each procedure of our proxy signature schemes, and comparison with Tang's scheme<sup>24</sup> (the only proxy scheme of MPKC we have known) and Mambo's scheme<sup>30</sup> (the first proxy scheme), Hu–Zhang (HZ) scheme<sup>37</sup> (an improved efficient secure proxy scheme) is shown in Table 1. We set the parameters in HZ scheme<sup>37</sup> with  $(n = 2, t = 1)$ . From Table 1, we can see that the proxy signature generation and the proxy signature verification of our schemes are more efficient than Tang's schemes in Table 1. According to recent research result, the time of generating a large prime is more than 20 times than

**Table 2.** Running time requirements by Proxy-UOV and Proxy-Rainbow and compared with other schemes.

	Proxy-UOV	Proxy-Rainbow	Mambo scheme	HZ scheme
Initialization (ms)	19,376	11,912	404,277	443,885
Proxy share generation (ms)	52,301	30,691	141	10,245
Proxy signature generation (ms)	187	61	47	4502
Signature verify (ms)	31	16	249	3646

UOV: Unbalanced Oil and Vinegar; HZ: Hu–Zhang.

**Table 3.** Running time requirements by Proxy-MQ and compared with Tang scheme.

	Proxy-MQ	Tang's scheme
Initialization (ms)	24,043	24,328
Proxy share generation (ms)	92,008	122,873
Proxy signature generation (ms)	30,521	48,642
Signature verify (ms)	74,215	75,832

MQ: multivariate quadratic.

that of generating a system of polynomials, and an exponentiation evaluation costs about half times than an evaluation operation, while choosing proper parameters with security level more than 80 bits. So, in this situation, all our schemes and Tang's scheme<sup>24</sup> have better initialization time and verify process time than that of Mambo's scheme<sup>30</sup> and HZ scheme.<sup>37</sup>

Notation for Table 1:  $u, m, n$ : the number of layers, polynomials, and variables in a scheme, respectively;  $S$ : average time required by a Gaussian Elimination function to solve linear equations;  $q$ : the length of output bits of the hash function for Tang scheme;<sup>24</sup>  $round$ : the iterative rounds for MQ-based signature;  $T_p, T_e, T_m, T_H$ : the time for generating a large prime, one exponentiation computation, one modular multiplication computation, and hash function computation, respectively;  $n_r, n_h$ : the size of the public key in a secure RSA scheme and the size of a secure hash function, respectively. The running time of our proposed Proxy-UOV and Proxy-Rainbow schemes (using our suggested parameters) compared with Mambo's scheme<sup>30</sup> and HZ scheme<sup>37</sup> (2048 bits) is shown in Table 2. From the experiments' results, we can see that our schemes' proxy signature generation time is between RSA and elliptic curve cryptography (ECC), while the initialization and verifying time are much faster than both of them. Consequently, even though our scheme is a bit slower than the proxy share generation time, but it is applicable in real life.

While running in Magma, the memory will be overflowed when using the suggested parameters of Proxy-MQ and Tang's scheme in Tang and Xu,<sup>24</sup> we have to modify the parameter to formulate the running time ( $(n = 42, m = 42, q = 2, round = 2)$  for Proxy-MQ,

$(n = 42, m = 42, q = 2, round = 2)$  for Tang's scheme), and the result is shown in Table 3.

From the result shows, the proxy signature scheme based on MQ is slightly more efficient than Tang's scheme<sup>24</sup> and is a promising proxy scheme by taking into account its post-quantum property in multivariate cryptography.

## Conclusion

A general proxy signature scheme based on MPKC signature scheme is proposed, which invokes three number of times of the underlying signature scheme, and can satisfy all the security requirement of a proxy scheme. Through formal security analysis, our general scheme is proved to be cured on *Existential Unforgeability under an Adaptive Chosen Message Attack with Proxy Key Exposure* assuming that the underlying MPKC signature is *Existential Unforgeability under an Adaptive Chosen Message Attack*. Thereafter, we propose three practical proxy schemes based on three promising MPKC schemes: UOV, Rainbow, and MQ-based scheme. Although the security of the underlying MPKC is still an open problem, the method to construct our proxy scheme and the method to formally prove the security of our proxy scheme are good exploration in the area of MPKC.

In the future work, we plan to construct other primitives based on multivariate signature scheme, such as identity-based signature, forward secure signature, and so on.

## Declaration of conflicting interests


The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the Key Areas Research and Development Program of Guangdong Province under Grant 2019B010139002, National Natural Science Foundation of China under Grant 61972094, National Natural Science Foundation of China under Grant 61902079,

and the project of Guangzhou Science and Technology (Grant 201902020006 and 201902020007).

## ORCID iD

Jiageng Chen  <https://orcid.org/0000-0001-9033-2575>

## References

1. Varadharajan V, Allen P and Black S. An analysis of the proxy problem in distributed systems. In: *Proceedings of the 1991 IEEE computer society symposium on research in security and privacy*, Oakland, CA, 20–22 May 1991, pp.255–275. New York: IEEE.
2. Park HU and Lee IY. A digital nominative proxy signature scheme for mobile communication. In: *Proceedings of the international conference on information and communications security*, Xi'an, China, 13–16 November 2001, pp.451–455. Berlin; Heidelberg: Springer.
3. Leiwo J, Hnle C, Homburg P, et al. Disallowing unauthorized state changes of distributed shared objects. In: *Proceedings of the IFIP international information security conference*, Beijing, China, 22–24 August 2000, pp.381–390. Boston, MA: Springer.
4. Foster I, Kesselman C, Tsudik G, et al. A security architecture for computational grids. In: *Proceedings of the 5th ACM conference on computer and communications security*, San Francisco, CA, November 1998, pp.83–92. New York: ACM.
5. Bakker A, Van Steen M and Tanenbaum AS. A law-abiding peer-to-peer network for free-software distribution. In: *Proceedings of the 2001 IEEE international symposium on network computing and applications*, Cambridge, MA, 8–10 October 2001, pp.60–67. New York: IEEE.
6. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput* 1997; 26: 1484–1509.
7. Bernstein DJ. Introduction to post-quantum cryptography. In: Bernstein DJ, Buchmann J and Dahmen E (eds) *Post-quantum cryptography*. Heidelberg: Springer, 2009, pp.1–14.
8. NIST CSRC: Cryptographic Technology Group. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2016, <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>
9. Matsumoto T and Imai H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: *Proceedings of the 7th international workshop on the theory and application of cryptographic techniques (EUROCRYPT'08)*, Davos, 25–27 May 1988, pp.419–453. Berlin: Springer
10. Patarin J. The oil and vinegar signature scheme. In: *Dagstuhl workshop on cryptography*, Schloss Dagstuhl, 22–26 September 1997. Dagstuhl.
11. Courtois N. Generic attacks and the security of Quartz. In: Desmedt Y (ed.) *Proceedings of the 6th international workshop on practice and theory in public key cryptography (PKC 2003)*, Miami, FL, 6–8 January 2003. Berlin: Springer, 2003, pp.351–364.
12. Kipnis A, Patarin J and Goubin L. Unbalanced oil and vinegar signature schemes. In: *Proceedings of the 18th annual international conference theory and application of cryptographic techniques (EUROCRYPT'99)*, Prague, 2–6 May 1999, pp.206–222. Berlin: Springer.
13. Bulygin S, Petzoldt A and Buchmann J. Towards provable security of the unbalanced oil and vinegar signature scheme under direct attacks. In: Gong G and Gupta K (eds) *Progress in Cryptology—INDOCRYPT 2010, lecture notes in computer science*, 6498. Berlin; Heidelberg: Springer, 2010, pp.17–32.
14. Sakumoto K, Shirai T and Hiwatari H. On provable security of UOV and HFE signature schemes against chosen-message attack. In: *Proceedings of the 4th international conference on post-quantum cryptography (PQCrypto'2011)*, Taipei, Taiwan, 29 November–2 December 2011, pp.68–82. Berlin: Springer.
15. Bellare M and Rogaway P. The exact security of digital signatures-how to sign with RSA and Rabin. In: *Proceedings of the 15th annual international conference on theory and application of cryptographic techniques (EUROCRYPT'06)*, Saragossa, 12–16 May 1996, pp.399–416. Berlin: Springer.
16. Sakumoto K, Shirai T and Hiwatari H. Public-key identification schemes based on multivariate quadratic polynomials. In: *Proceedings of the 31st annual international cryptography conference (CRYPTO' 2011)*, Santa Barbara, CA, 14–18 August 2011, pp.706–723. Berlin: Springer.
17. Faugère J-C, Perret L and Ryckeghem J. DualModeMS: a dual mode for multivariate-based signature 20170918 draft (First round submission to the NIST post-quantum cryptography call). November 2017, [https://www.polsys.lip6.fr/Links/NIST/DualModeMS\\_specification.pdf](https://www.polsys.lip6.fr/Links/NIST/DualModeMS_specification.pdf)
18. Casanova A, Faugère J-C, Macario-Rat G, et al. GeMSS: a great multivariate short signature (First round submission to the NIST post-quantum cryptography call), November 2017, [https://www.polsys.lip6.fr/Links/NIST/GeMSS\\_specification.pdf](https://www.polsys.lip6.fr/Links/NIST/GeMSS_specification.pdf)
19. Petzoldt A, Chen M-S, Yang B-Y, et al. Design principles for HFEV-based multivariate signature schemes. In: *Proceedings of the 21th international conference on the theory and application of cryptography and information security (ASIACRYPT'2015)*, Auckland, New Zealand, November 29–December 3 2015, pp.311–334. Berlin: Springer.
20. Beullens W and Preneel B. Field lifting for smaller UOV public keys. In: Patra A and Smart N (eds) *International conference in cryptology in India—INDOCRYPT 2017, LNCS, vol. 10698*. Heidelberg: Springer, 2017, pp.227–246.
21. Chen M-S, Hülsing A, Joost A, et al. From 5-pass MQ-based identification to MQ-based signatures. In: *Advances in cryptography—ASIACRYPT 2016, LNCS, vol. 10032*. Heidelberg: Springer, 2016, pp.135–165.
22. Ding J and Schmidt D. Rainbow a new multivariable polynomial signature scheme. In: *Proceedings of the 3th international conference on applied cryptography and network security (ACNS'2005)*, New York, 7–10 June 2005, pp.164–175. Berlin: Springer.
23. Peretz Y. On multivariable encryption schemes based on simultaneous algebraic Riccati equations over finite fields. *Finite Fields Th App* 2016; 39: 1–35.

24. Tang S and Xu L. Towards provably secure proxy signature scheme based on isomorphisms of polynomials. *Future Gener Comp Sy* 2014; 30: 91–97.
25. Petzoldt A, Bulygin S and Buchmann J. A multivariate based threshold ring signature scheme: applicable algebra in engineering. *Commun Comput* 2013; 24: 255–275.
26. Chen J, Tang S, He D, et al. Online/offline signature based on UOV in wireless sensor networks. *Wirel Netw* 2017; 23: 1719–1730.
27. Petzoldt A, Szepieniec A and Mohamed MSE. A practical multivariate blind signature scheme (Cryptology ePrint Archive, Report2017/131), 2017, <http://eprint.iacr.org/2017/131>
28. El Bansarkhani R, Mohamed MSE and Petzoldt A. MQSAS: a multivariate sequential aggregate signature scheme. In: *Proceedings of the 19th international conference on information security (ISC 2016)*, Honolulu, HI, 3–6 September 2016, pp.426–439. Cham: Springer.
29. Verma GK, Singh BB and Singh H. Bandwidth efficient designated verifier proxy signature scheme for healthcare wireless sensor networks. *Ad Hoc Netw* 2018; 81: 100–108.
30. Mambo M, Usuda K and Okamoto E. Proxy signatures: delegation of the power to sign messages. *IEICE T Fund Elec Commun Comp Sci* 1996; 79: 1338–1354.
31. Fiat A and Shamir A. How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko A (ed.) *Proceedings of the 6th annual international cryptology conference (CRYPTO'06)*, Santa Barbara, CA. Berlin: Springer, 1987, pp.186–194.
32. Schuldt J, Matsuura K and Paterson K. Proxy signatures secure against proxy key exposure. In: *Proceedings of the 11th international conference on theory and practice of public key cryptography (PKC 2008)*, Barcelona, 9–12 March 2008, pp.141–161. Berlin: Springer.
33. Petzoldt A, Bulygin S and Buchmann J. Linear recurring sequences for the UOV key generation. In: *Proceedings of the 14th international conference on theory and practice of public key cryptography (PKC 2011)*, Taormina, 6–9 March 2011, pp.335–350. Berlin: Springer.
34. Petzoldt A, Bulygin S and Buchmann J. A multivariate signature scheme with a partially cyclic public key. In: Gong G and Gupta KC (eds) *Proceedings of SCC*. Berlin: Springer, 2010, pp.33–48.
35. Patarin J and Goubin L. Trapdoor one-way permutations and multivariate polynomials. In: *Proceedings of the 1st international conference on information security and cryptology (ICISC 1997)*, Beijing, China, 11–14 November 1997, pp.53–66. Berlin: Springer.
36. Bosma W, Cannon J and Playoust C. The Magma algebra system I: the user language. *J Symb Comput* 1997; 24: 235–265.
37. Hu J and Zhang J. Cryptanalysis and improvement of a threshold proxy signature scheme. *Comp Stand Inter* 2009; 31: 169–173.