MQ Signature and Proxy Signature Schemes with Exact Security Based on UOV Signature

Shaohua Tang[1,⋆], Jiahui Chen[2], Lingling Xu[3], Xiaoyu Li[4]

School of Computer Science & Engineering,
South China University of Technology, Guangzhou, China
[1,3]{csshtang,csllxu}@scut.edu.cn
[2,4]{c.jiahui01,l.xy09}@mail.scut.edu.cn

Abstract. Multivariate public key cryptography which relies on MQ (Multivariate Quadratic) problems is one of the main approaches to guarantee the security of communication in the post-quantum world. In this paper, we propose a combined MQ signature scheme based on the yet unbroken UOV (Unbalanced Oil and Vinegar) signature if parameters are properly chosen. Our scheme can not only reduce the public key size of the UOV signature, but also provide more tighter bound of security against chosen-message attack in the random oracle model. On the other hand, we propose a proxy signature scheme based on our proposed combined signature scheme. Additionally, we give a strict security proof for our proxy signature scheme. Finally, we present experiments for all of our proposed schemes and the baseline schemes. Comparisons with related schemes show that our work has some advantages on performance along with more strict security.

Keywords: Multivariate Quadratic Problem, Multivariate Public Key Cryptography, UOV Signature, Exact Security, Proxy Signature

1 Introduction

In recent years, finding alternative public key cryptosystems, which have resistance to attacks by quantum computers, has become a vital challenge, i.e., if a practical quantum computer is built, and the public key cryptosystems used today, for example RSA, ECC, El Gamal, etc., are broken. The multivariate public key cryptosystem (MPKC) is one of the promising candidates for post-quantum cryptography. MPKC is based on the problem of solving a system of multivariate quadratic polynomials, which is called an MQ problem [12]. Except resistance to quantum computer attacks, MPKCs enjoy other useful properties. In particular, they are quite fast compared to conventional schemes and require only very moderate resources, since the arithmetic operations are usually performed over a small finite field. This makes MPKCs excellent candidates for use in resource-constrained devices, like WSN nodes, RFIDs and smart cards, etc.

⋆ Corresponding author: Shaohua Tang.

However, there are two drawbacks that become obstacles to use MPKCs. The first one is the largeness of its key sizes, and the second is that the security of MPKC relies both on the MQ problem and on the Isomorphism of Polynomials problem. Therefore, MPKC schemes face not only direct attacks but also structural attacks. Under this situation, quite a few attempts have been undertaken to tackle these two problems. For example, Petzoldt *et al.* [21] undertook an attempt to reduce the public key size, based on yet unbroken (under proper parameter choice) Unbalanced Oil and Vinegar (UOV) scheme [17]. Sakumoto *et al.* [23] proposed provably secure identification/signature schemes based on the MQ problem. There has been plenty of proposals of MPKCs, as mentioned in [12] and [13]. On the disadvantage for the designers, the cryptanalytic process has also been substantial. New proposals aim mainly at fixing problems exposed by the cryptanalysis, but then it often happens that "fixed" proposals get broken again. This is due to the fact that little attention has been given to provable security of MPKC schemes. Although Courtois [11] studied provable security against key-only attack on Quartz, the security against chosen-message attack is still unclear. Beyond these techniques, the UOV [17] is a well-known and deeply studied scheme in MPKC. Petzoldt *et al.* [9] presented an idea to reduce the public key size of the UOV signature and provided a provable security under direct attacks. Then, they used such an idea to create a multivariate signature scheme [22] whose public key is mainly given by a linear recurring sequence (LRS). Both the above work, however, did not present sufficient provable security of UOV. Then, Sakumoto *et al.* [24] gave provable security of UOV against chosen-message attack, using the idea given in [3], which concatenates a random seed r into the signing message M so as to make the basic one-way trapdoor function of UOV become full domain hash (FDH). However, the authors paid no attention to the public key size of UOV. In this paper, we overcome the above two drawbacks of MPKC by proposing a combined signature scheme based on UOV that can (1) reduce the public key size of the UOV signature, and (2) not only provide provable security against chosen-message attack, but also provide exact security analysis so as to get a better security bound.

On the other hand, 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 the original signer, in case of temporal absence, lack of time or computational power, etc. The first efficient proxy signature was introduced by Mambo, Usuda and Okamoto [19]. However, almost all proxy signature schemes so far are based on the difficult problem of integer factoring, discrete logarithm, and/or elliptic curve, and can not resist the attacks of future quantum computers. Then, Tang *et al.* ([26], [27]) proposed the first post-quantum proxy signature scheme based on the problem of Isomorphisms of Polynomials (IP), but its efficiency can still have the room to be improved. Due to recent results by Berthomieu *et al.* [5] to solve IP problems, in order to maintain the practical security, IP-based schemes need big parameters then their performances are affected. By contrast, our proxy signature ultimately based on UOV is a more promising scheme since the trapdoor of UOV is still secure and efficient.

1.1 Our Contribution

We propose a combined signature scheme based on UOV signature, which is consisted of two strategies: reducing the size of public key, and making the trapdoor function uniform. More importantly, we formally show that our new scheme can not only reduce the public key size of the UOV signature, but also can provide more tighter security proof.

Thereafter, we propose a proxy signature scheme based on our proposed signature scheme, which includes the stages of initialization, delegation and proxy key generation, generation of proxy signature, and the verification of proxy signature. We also give a strict security proof for our proxy signature scheme under the security of our proposed signature scheme.

Finally, we present experiments and comparisons for all of our proposed schemes and the baseline schemes.

1.2 Organization

The rest of the paper is organized as follows. In Section 2, we describe the preliminaries, security models, and basic UOV signature scheme in detail. We present our proposed combined signature scheme in Section 3. Then, in Section 4, the proposed proxy signature scheme based on our proposed signature scheme is described. We present the security analysis in Section 5. Experiments and comparisons are given in Section 6. Finally, in Section 7, we concludes the paper with a discussion.

2 Preliminaries

In this section, we give the preliminaries of this paper, which includes definitions of used notations, security models, and basic UOV signature scheme along with some strategies about it.

2.1 Schemes

Signature scheme. We start by recalling the definition of a signature scheme.

Difinition 1. *A signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is defined as follows.*

The key-generation algorithm Gen *is a probabilistic algorithm which outputs a pair of matching public and private keys* (pk, sk) *for a given* 1^λ.

The signing algorithm Sig *is a probabilistic algorithm which takes as input the message* M *to be signed and a secret key* sk*, and returns a signature* $\sigma = \mathsf{Sig}_{sk}(M)$.

The verification algorithm Ver *takes as input a message* M*, a candidate signature* σ *and public key* pk*, and returns a bit* $\mathsf{Ver}_{pk}(M, \sigma)$*. If* $\sigma \leftarrow \mathsf{Sig}_{sk}(M)$*, then* $\mathsf{Ver}_{pk}(M, \sigma) = 1$. The signature is accepted, only if the bit is equal to 1; otherwise, rejected.

The full domain hash (FDH) scheme. Similar to earlier work [4], we suggested to hash m onto the full domain of the UOV function before verification. That is, $\mathsf{Hash}_{FDH} : (0,1)^* \rightarrow k^n$, which hashes strings uniformly into k^n; and the signature on m is $\mathsf{Sig}_{FDH}(m) = f^{-1}(\mathsf{Hash}_{FDH}(m))$, where f^{-1} stands for the signing function. We call this the *Full-Domain-Hash (FDH)* scheme.

Proxy signature scheme. A delegator and a proxy are included in the proxy scheme. A proxy signature scheme is an extension of an ordinary signature scheme with the following additional algorithms: (Delegate, ProxyKeyGen), ProxySign, and ProxyVerify. The pair of algorithms (Delegate, ProxyKeyGen) is for delegation of signing rights. ProxySign is run by the proxy and outputs a proxy signature $\sigma(m)$. ProxyVerify is run by the public to verify the validity of a proxy signature, and outputs a bit equals to either 1 or 0.

2.2 Models

Security model for signature. We say that a signature scheme is $(t', \varepsilon') - secure$, if an attacker, given y selected randomly from $(0,1)^*$ and limited to running in time $t'(\lambda)$, succeeds in finding $f^{-1}(y)$ with probability at most $\varepsilon'(\lambda)$, where f^{-1} stands for the signing function.

We quantify UOV as a uniform one-way function.

Difinition 2. *We say that a UOV function generator is* $(t'(\lambda), \varepsilon'(\lambda)) - secure$ *if there is no inverting algorithm that takes* P *and* y *as inputs and outputs a preimage* x *such that* $P(x) = y$ *at* $t'(\lambda)$ *processing time with probability at least* $\varepsilon'(\lambda)$*, where* P *is obtained by running* $\mathsf{Gen}(1^\lambda)$ *and* y *is randomly chosen from* k^n*. The standard asymptotic definition of security requests that the success probability of any PPT (probabilistic, polynomial time) algorithm is a negligible function of* λ*.*

Note that it is safe to assume UOV is $(t', \varepsilon') - secure$ and then the values of (t', ε') can be provided based on the perceived cryptanalytic strength of UOV.

Exact security of FDH. The "exact security" of the reduction, which was used to prove the security of the FDH signature scheme, was first provides by Bellare and Rogaway [4] and analyzed in Theorem 1 of [4]. It claims that if a signature scheme is $(t', \varepsilon') - secure$, and q_{sig} and q_{hash} , which are legitimate message-signature pairs and invocation times of the (ideal) hash function respectively, are given, then the FDH scheme based on such scheme is $(t, q_{sig}, q_{hash}, \varepsilon) - secure$, where

$$t(\lambda) = t'(\lambda) - (q_{hash}(\lambda) + q_{sig}(\lambda) + 1) \cdot O(\lambda^3) \tag{1}$$

and

$$\varepsilon(\lambda) = (q_{hash}(\lambda) + q_{sig}(\lambda)) \cdot \varepsilon'(\lambda). \tag{2}$$

Definition 3. *We say that a UOV-based FDH signature scheme is* $(t, q_{sig}, q_{hash}, \varepsilon) - secure$ *if there is no forger* $\mathcal{A}$ *who takes a public key pk generated via*

$(pk, \cdot) \leftarrow \mathsf{Gen}(1^\lambda)$, *after at most* $q_{hash}(\lambda)$ *queries to the random oracle,* $q_{sig}(\lambda)$ *signature queries, and* $t(\lambda)$ *processing time, then outputs a valid signature with probability at least* $\varepsilon(\lambda)$.

Note that, for large q_{sig} and q_{hash}, even if the UOV signature scheme is quite strong, the guarantee on the FDH signature scheme could still be quite weak. For example, if we allow the forger to see at least $q_{sig}(\lambda) = 2^{59}$, then even if the UOV inversion probability was originally as low as 2^{-61}(which is an excellent bound), the forging probability is over $1/2$, which is not good enough. To avoid this, one would try to prove a better security bound for FDH than that outlined in [10], or one would try to reconstruct the FDH scheme so as to have "tighter" reductions of ε to ε' [4]. And we will show in Section 5 that our scheme enjoys the former.

Security model for proxy signature. Schuldt *et al.* [25] presented the security notion *Existential Unforgeability under an Adaptive Chosen Message Attack with Proxy Key Exposure* (ps-uf-pke) for multi-level proxy signature scheme. And Tang *et al.* [27] 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 Tang *et al* 's model. For more detail on this model, we refer the reader to [27].

Definition 4. *An adversary* $\mathcal{A}$ *is said to be a* $(\varepsilon, t, q_d, q_s)$*-forger of a proxy signature scheme if* $\mathcal{A}$ *has advantage at least* ε *in the 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* $(\varepsilon, t, q_d, q_s)$*-secure if no* $(\varepsilon, t, q_d, q_s)$*-forger exists.*

2.3 UOV Signature Scheme

The idea of the Oil and Vinegar trapdoor was first proposed by J. Patarin [20]. However, the original scheme was broken by Kipnis and Shamir [18], and was recommended in [17] to choose $v > o$ (Unbalanced Oil and Vinegar). The UOV scheme is a single field construction, so we work solely in the polynomial ring $K[X]$, where K is a finite field and $X = \{x_1, ..., x_n\}$. We divide X into two variable sets: vinegar variables $(x_i)_{i \in V}$, where $V = \{1, ..., v\}$ and oil variables $(x_i)_{i \in O}$, $O = \{v + 1, ..., n\}$. Here $|V| = v$, $|O| = o$ and $v + o = n$. We define o quadratic polynomials $q_k(X) = q_k(x_1, ..., x_n)$ by

$$q_k(X) = \sum_{i \in V, j \in O} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}, k = 1, ..., o. \tag{3}$$

The map $Q = (q_1(X), \ldots, q_o(X))$ can be easily inverted. First, we choose the values of the v vinegar variables $x_1, \ldots, x_v$ at random. Then we can get a system of o linear equations in the o variables $x_{v+1}, \ldots, x_n$ which can be solved by Gaussian elimination. If it does not have a solution, we simply choose other values of $x_1, \ldots, x_v$ and repeat the above procedure.

The public key P of the UOV scheme consists of o quadratic polynomials in n variables:

$$P = (p^{(1)}, ..., p^{(o)})$$
$$= \left(\sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(1)} x_i x_j + \sum_{i=1}^{n} p_i^{(1)} x_i + p_o^{(1)}, ..., \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(o)} x_i x_j + \sum_{i=1}^{n} p_i^{(o)} x_i + p_o^{(o)} \right).$$

To hide the structure of Q in the public key, one concatenates it with an invertible affine map $T : F^n \to F^n$, then the public key of the UOV signature scheme is $P = Q \circ T$.

2.4 Reducing the Size of Public Key of UOV

There are a lot of skills, e.g., [21] and [22], to reduce the size of public key of UOV. One of the most widely use is using cyclic matrix to construct the coefficients of its public polynomials [21]. In the construction of reducing the public key size of the basic UOV, one can choose randomly an affine invertible map T (given as a matrix $M_T = (t_{ij})_{i,j=1}^{n}$ and an n- vector c_T). Then let $q_{ij}^{(k)}$ be the coefficients of quadratic terms of the central map polynomials from Q, due to $P = Q \circ T$, we have

$$p_{ij}^{(r)} = \sum_{k=1}^{n} \sum_{l=k}^{n} a_{kl}^{ij} \cdot q_{kl}^{(r)} = \sum_{k=1}^{v} \sum_{l=k}^{n} a_{kl}^{ij} \cdot q_{kl}^{(r)} (1 \leq i \leq j \leq n, r = 1, ..., o) \quad (4)$$

with

$$a_{kl}^{ij} = \begin{cases} t_{ki} \cdot t_{li} (i = j) \\ t_{ki} \cdot t_{lj} + t_{kj} \cdot t_{li} (i \neq j), \end{cases} \quad (5)$$

where $p_{ij}^{(k)}$ is the coefficients of quadratic terms of the central map polynomials from P. Let M_P and M_Q be the Macaulay matrices of P and Q respectively (in graded lexicographical order), we can find that $D = \frac{v \cdot (v+1)}{2} + o \cdot v$ is the number of non-zero quadratic terms in any component of Q. If we set a transformation $D \times D$ matrix A containing the coefficients $a_{ij}^{(rs)}$, then we have

$$M' = Q \cdot A, \quad (6)$$

where M' is a submatrix of M_P composed of the first D columns. After fixing the private key (Q, T) of UOV, then we can insert a partially circulant matrix (M') into the UOV public key so as to reduce the public key size by solving this equation.

2.5 Making the Trapdoor Function Uniform

In most of the signature schemes, standard trapdoor functions are not uniform, and the message must be padded before handling . There are many ways of padding. For example, to sign m, we could modify the scheme to compute $H(1||m), H(2||m), ..,$ until a member $y = H(i||m)$ of the domain is found and

then return $(i, f^{-1}(y))$. Another alternative is to apply the construction of Section 4.2 in [2] to make them uniform, etc.

In the case of UOV scheme, Sakumoto *et al.* [24] discussed its non-uniformity of signatures. They figure out that, suppose the signing algorithm chooses a set of vinegar variables such that the rank of the matrix is equal to i, and entries of the matrix are polynomials of the first degree on $x_{n+1}, ..., x_{n+v}$ of the private map q_i, then the private map is a q^{n-i}-to-1 mapping. Since The higher the rank is, the higher the probability of ending the loop is, then such a set of vinegar variables is a q^{n-i}-to-1 mappings to be output more frequently. Thus the signature distribution is non-uniform. So the essential problem to make the trapdoor function uniform is to make the output x'_v uniformly distributed, i.e., each map q_i of Q is a q^{n-i}-to-1 mapping. To achieve this, one can take the first padding strategy and let y be generated via $y \leftarrow H(m||r)$, where r is a random salt.

3 Our Proposed Combined Signature Scheme

In this section, we first choose valid parameters for the original UOV scheme and study its exact security, then we describe the construction of our proposed signature scheme which is combined with the above two strategies.

3.1 Choosing Valid Parameter

At the beginning of our construction, we would like to choose appropriate parameters for the original UOV scheme so as to make it "secure" with exact security satisfying at least $t'(\lambda)/\varepsilon'(\lambda) < 2^{64}$. We choose $q = 2^5, n = 26$, and $v = 52$, then we can enjoy a property of UOV function described by Proposition 1 below.

Proposition 1. *The $(2^5, 26, 52)$-UOV function generator is $(t'(\lambda), \varepsilon'(\lambda)) -$ secure such that $t'(\lambda)$ and $\varepsilon'(\lambda)$ satisfy $t'(\lambda)/\varepsilon'(\lambda) < 2^{72.4}$ under current attack techniques.*

proof. In Definition 2, we claim that the UOV function generator is $(t'(\lambda), \varepsilon'(\lambda))-$ *secure* if there is no inverting algorithm that takes P and y as inputs and outputs a preimage x such that $P(x) = y$ at $t'(\lambda)$ processing time with probability at least $\varepsilon'(\lambda)$, where P is obtained by running $\mathsf{UOVGen}(1^\lambda)$ and y is randomly chosen from k^n. The standard asymptotic definition of security requests that the success probability of any PPT (probabilistic, polynomial time) algorithm is a negligible function of λ. For the exact security of UOV, we interest in exactly how much time $t'(\lambda)$ an inverting algorithm uses and what success probability $\varepsilon'(\lambda)$ it achieves in this time. For example, the asymptotically best algorithm that inverts UOV [17] takes time about $q^{v-n-1}n^4$ to break a (q, v, n)-UOV scheme. So one might be willing to assume that the UOV is $(t', \varepsilon') - secure$ for any (t', ε') satisfying $t'(\lambda)/\varepsilon'(\lambda) < Cq^{v-n-1}n^4$, for some particular constant C.

We note that it is not clear whether the UOV signature schemes is secure or not, even if their underlying trapdoor functions are assumed to be one-way.

Luckily, we can figure out the best bound of attacking UOV and make this the exact security of UOV.

The Kipnis and Shamir attack. The goal of this attack is to find the pre-image of the oil subspace $O = \{x \in K_n : x_1 = ... = x_v = 0\}$ under the affine invertible transformation T . To achieve this, it forms a random linear combination $P = \sum_{j=1}^{o} \beta_j H_j$, multiplies it with the inverse of one of the H_i and figures out the invariant subspaces of this matrix. As described in [17], the Kipnis and Shamir attack takes time about $q^{v-n-1}n^4$ to break a (q,v,n)-UOV scheme. Taking into account the recommended parameters in [16], i.e. q=16, n=16, v=48, the Kipnis and Shamir attack may take time at least 2^{64} to break this scheme. So the exact security of UOV under Kipnis and Shamir attack satisfies $t'(\lambda)/\varepsilon'(\lambda) < 2^{64}$, this means that it needs at least 2^{65} time complexity to invert the trap-door function with success probability greater than $1/2$.

Exhaustive search attack. The best exhaustive search algorithm is described in [8], which breaks MQ(n, m, F_2) in $2^{n+2} \cdot log_2 n$ bit operations. This means that it would break a $(2, 64, 128)$-UOV scheme (the recommended parameters in [16]) in $7 \cdot 2^{66}$ bit operations. Additionally, a traditional exhaustive search algorithm needs $q \cdot (n+1) \cdot q^n$ bit operations to break a (q,n,v)-UOV scheme.

Note that the best exhaustive search attack algorithm is valid only under the field of order 2. To the best of our knowledge, there is still no other fast algorithms solving arbitrary fields.

Direct attack. There are many algorithms working on UOV, such as Gröbner basis techniques and its variants F_4 [15] . The idea of these direct attack on UOV is to add v linear equations. In this way, the number of variables can be reduced to n. On the other hand, a system with $v+n$ variables and n equations is expected to have q^v solutions on average. Therefore, adding a total of v linear equations will lead to one solution on average. Repeating this experiment a few times, we will find at least one solution. Still the concrete complexity of these algorithms are not fixed, but experts believe [1] that these methods will go up to a certain degree D_0 and then require the solution of a system of linear equations with T variables, where T is greater than $\binom{v+n}{D_0-1}$, and this will take at least $poly(n+v) \cdot T^2$ bit operations, where $poly(n+v) = 3(n+v)(n+v-1)/2$ under some hard assumptions. Solving this system of linear equations would not outperform complexity of 2^{64} when $n \geq 16$ [17] in the UOV scheme.

Rank Attack. In the case of the basic UOV scheme, one can find that all the matrices Q_i representing the homogeneous quadratic parts of the central equations have full rank n. And this prevents the MinRank attack. Furthermore, all variables $x_1, ..., x_n$ appear in each of the o central equations, which prevents HighRank attacks.

Hybrid Attack. Despite that UOV is secure under the Kipnis and Shamir attack with the recommended parameters in [16], Faugère and Perret [14] reported that the UOV function with only 64-bit output is broken in a complexity bounded by $2^{40.3}$ for certain recommended parameter, e.g. q=16, n=16, v=48, by using a so-called hybrid attack algorithm which is a combination of

exhaustive search attack and direct attack. After that, they presented in [6] the exact complexity of their hybrid approach for solving multivariate systems over finite fields. They concluded that the complexity of the hybrid approach is $C_{hyb} = q^{k'}\left(\frac{n}{\sqrt{2\pi}}\right)^{\omega}\left(\frac{\left(\frac{3n-k'}{2}-1-\sqrt{nk'}\right)^{(3n-k-1)/2-\sqrt{nk'}}}{(n-k'-1)^{(n-k-1/2)}\left(\frac{n+k'}{2}-\sqrt{nk}\right)^{(n+k'+1)/2-\sqrt{nk'}}}\right)^{\omega}$, where $2 \leq \omega \leq 3$ is the linear algebra constant for small values of k', and k' is the best theoretical trade-off for hybrid solving. More precisely, k' satisfies: $\log(C_{hyb}) \sim k'\log(q)+\omega(0.995n+0.5\log(n)+0.144k'-1.099\sqrt{nk'}-0.919)$, and is calculated by setting the logarithmic derivative of C_{hyb} to 0.

Note that in all these formulations, n is number of the equations of the system which is equal to o and is also the variables needed to be solved in the system, since the hybrid attack on UOV will firstly randomly specialize (i.e. fix) v variables and then solve a system having the same number (n) of variables and equations.

Thus, in the choice of parameters that $q = 2^5, n = 26, v = 52$, we can figure out the exact security of UOV satisfies $t'(\lambda)/\varepsilon'(\lambda) < 2^{5\cdot(52-26-1)} \cdot 26^4 < 2^{143}$ under the Kipnis and Shamir attack, $t'(\lambda)/\varepsilon'(\lambda) < 2^5 \cdot 27 \cdot 2^{125} < 2^{135}$ under the exhaustive search attack, $t'(\lambda)/\varepsilon'(\lambda) < (3 \cdot 84 \cdot 83/2) \cdot \left(78\right)^2 < 2^{85}$ under direct attack with feasible assumption that D_0 is equal to 9 (if $n+v$ is larger, we expect to have $D_0 = \sqrt{n+v}$), and $t'(\lambda)/\varepsilon'(\lambda) < (\log(32) + 2(0.995 \cdot 26 + 0.5 \cdot \log(26) + 0.144 - 1.099\sqrt{26} - 0.919))/\log(2) < 2^{72.4}$ under hybrid attack with $\omega = 2$, and the best trade-off $k' = 1$ calculated by substituting the parameters.

After all the above analysis, we can charily conclude a property of UOV scheme that the $(2^5, 26, 52)$-UOV function generator is $(t'(\lambda), \varepsilon'(\lambda))-secure$ such that $t'(\lambda), \varepsilon'(\lambda)$ satisfy $t'(\lambda)/\varepsilon'(\lambda) < 2^{72.4}$ under current best attack techniques.

3.2 Construction of Our Combined Signature Scheme

The details of the combined scheme are as follows.

Key Generation. The key generation algorithm Gen is described in Algorithm 1.

Algorithm 1 Gen(K, o, v, D)

Input:

K: the ground field (e.g. $K = GF(2^5)$);

o,v: the number of Oil and Vinegar variables respectively;

D: the number of non-zero quadratic terms, i.e. $D = \frac{v\cdot(v+1)}{2} + o \cdot v$;

Output:

(T, Q): the private key to sign the message;

P: the public key corresponding to (T, Q);

Begin

1: Choose a vector $\mathbf{b} = (b_0, ..., b_{D-1})$ at random;

2: Choose an $n \times n$ invertible matrix T at random, where $n = o + v$;

3: Set the entries of the first D columns of P to $p_{ij} = b_{(j-i) \bmod D}$;

4: Solve for $i = 0, ..., o-1$ the linear systems given by Eq. (4) (for $j < D$) to get

non-zero coefficients of the quadratic terms of the central map Q;
5: Choose coefficients of the linear and constant terms of the central map at random;
6: Compute remaining coefficients of public polynomials by composing Q and T;
7: **return** (Q, T, P);
End

Gen takes as inputs the the ground field, the number of Oil and Vinegar variables, and the number of non-zero quadratic terms, then it adopts the strategy of our described "Reducing the size of the public key" above, and returns the public/private key pairs. In the construction of the proceeding of reducing the public key size, the algorithm Gen inserts a matrix which is constrained by the private central map Q into the public central map P, so this part of P can be reduced.

Signature Generation. Since our idea of the proposed scheme is to make it into FDH-like scheme, the construction of signing algorithm Sig of our scheme is to use the strategy of our described "Making the trapdoor function uniform" strategy above. It firstly chooses a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^q$, and its detail is described in Algorithm 2.

Algorithm 2 Sig($m, (T, Q), P$)

Input:
m: the message to sign;
(T, Q): the private key to sign the message;
P: the public key corresponding to (T, Q);
Output:
V: the signature on message m;
Begin
1: Select $x_v{}' \in_R k^v$;
2: **repeat**
3: Select $r \in_R \{0,1\}^l$;
4: Let $y \leftarrow H(m||r)$;
5: **until** $\{z_n | Q(z_n, x_v{}') = y\} \neq \emptyset$;
6: Select $x_n{}' \in_R \{z_n | Q(z_n, x_v{}') = y\}$;
7: Let $x \leftarrow T^{-1}(x_n{}', x_v{}')$;
8: **return** $V = (H, x, r)$;
End

Signature Verification. Finally, the verification algorithm Ver(H,V,m) returns 1 if $P(x) = H(m||r)$, otherwise it returns 0.

Note that although our combined scheme is simply a combination of two widely-used strategies described above, it can conquer the two main drawbacks of an MPKC signature scheme. More precisely, it can not only reduce the size of public key, but also sustain the security strength of the trapdoor function and

get better security bound in the random oracle model. The formal analysis of this is shown in Section 5.

4 Our Proposed Proxy Signature Scheme

In this section, we propose a MQ proxy signature scheme based on our combined signature scheme, and our proxy signature scheme includes the stages of initialization, delegation and proxy key generation, generation of proxy signature, and the verification of proxy signature.

4.1 Initialization

We assume that all users can be uniquely identified by their public keys. So a delegator and a proxy can be uniquely identified by their public keys. This requirement can be met in practice by requiring the certification authority not to issue certificates for two different users on the same public key.

Suppose that Alice and Bob are users, where Alice is the delegator, and Bob is the proxy signer. According to our proposed signature scheme, the private key of Alice sk_A is consisted of (Q_A, T_A) which is a pair built by the key generation algorithm Gen of our proposed signature scheme in Algorithm 2, and the corresponding public key pk_A is P_A that satisfies $P_A = Q_A \circ T_A$. Similarly, the private key of Bob sk_B consists of (Q_B, T_B) which is a pair built by the key generation algorithm Gen of our proposed signature scheme in Algorithm 2, and the corresponding public key pk_B is P_B that satisfies $P_B = Q_B \circ T_B$.

The public keys of Alice and Bob, i.e., $pk_A = P_A$ and $pk_B = P_B$, are published to the public bulletin board.

4.2 Delegation and Proxy Key Generation

At this stage, a delegation token represents the proxy signing power authorized to Bob by Alice, and is computed by Alice and delivered to Bob. Then Bob can generate the proxy signing key by invoking its own private key and the delegation token. The detailed procedures, denoted by $\mathsf{Delegate}$ and $\mathsf{ProxyKeyGen}$ algorithm, are described as follows.

$\mathsf{Delegate}$. Alice randomly chooses a bijective affine transformation T, then computes ${T_A}' = T_A \circ T$ and ${P_A}' = P_B \circ {T_A}'$. The affine T should be kept secret by Alice. Alice sends $({T_A}', {P_A}')$ and the warrant $(w, cert)$ to Bob via 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 $cert$ is a signature on w generated by Alice using our proposed signing algorithm, that is, $cert = \mathsf{Sig}(w, sk_A, pk_A)$.

$\mathsf{ProxyKeyGen}$. After receiving $({T_A}', {P_A}', w, cert)$, Bob computes $T_p = T_B \circ {T_A}'$, and $Q_p = Q_B$. Let $sk_p = (Q_p, T_p)$, and $pk_p = {P_A}'$. Then sk_p is a private key for ordinary signature, and the corresponding public key is pk_p, because the following equation holds.

$$Q_p \circ T_p = Q_B \circ T_B \circ {T_A}' = P_B \circ {T_A}' = {P_A}'. \tag{7}$$

Then Bob computes a signature σ_{prx} by running

$$\sigma_{prx} = \mathsf{Sig}((w, cert, pk_p), sk_B, pk_B), \tag{8}$$

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

Note that the proxy signer Bob cannot derive the original signer Alice's private key (Q_A, T_A) from the received message $(T_A{}', P_A{}', w, cert)$ in the algorithm Delegation, since the affine transformation T is kept secret by Alice.

4.3 Generation of Proxy Signature

The proxy signature algorithm, denoted by ProxySign, can let the proxy signer Bob generate a proxy signature on behalf of the delegator Alice. For any given message m, Bob invokes the ordinary signing algorithm Sig of our proposed signature scheme to sign it, using $sk_p = (Q_p, T_p)$ as private key and $pk_p = P_A{}'$ as public key, and obtains $\sigma = \mathsf{Sig}(m, sk_p, pk_p)$. As a result, $(\sigma, (w, cert, pk_p, \sigma_{prx}))$ is the proxy signature on m by Bob on behalf of Alice.

4.4 Verification of Proxy Signature

The proxy signature verification algorithm, denoted by ProxyVerify, can let anyone verify the validity of a proxy signature, and is described in Algorithm 3.

Algorithm 3 ProxyVerify(m, $(\sigma, (w, cert, pk_p, \sigma_{prx}))$)

Input:
m: the message;
$(w, cert, pk_p, \sigma_{prx}))$: the proxy signature;
Output:
1 or 0;
Begin
1: Get Alice's public key $pk_A = P_A$ and Bob's public key $pk_B = P_B$ from the public bulletin board;
2: Check the validity of the signature $cert$ on w : $\mathsf{Ver}(w, cert, pk_A) == 1$? ;
3: Check the validity of the signature σ_{prx} on $(w, cert, pk_p)$:
$\mathsf{Ver}((w, cert, pk_p), \sigma_{prx}, pk_B) == 1$? ;
4: Check whether or not t described in w is a valid time period;
5: Verify the proxy signature by invoking the algorithm $\mathsf{Ver}(m, \sigma, pk_p) == 1$? ;
6: **If** the above conditions of 2,3,4,5 hold true **Then**
7: **Return** 1;
8: **Else**
9: **Return** 0;
End

Remark. If the time period t in the warrant w is expired, the delegated signing privilege is revoked automatically. Besides, the original signer can also broadcast a signed message to announce the invalidation of the warrant w. Then the proxy signature generated by Bob hereafter will become invalid.

5 Security Analysis

In this section, we analyze our proposed combined signature scheme according to the claimed properties of both the public key size reducing and the security proof. Then, we also present a strict security proof for our proxy signature scheme.

Since the construction of our signature scheme contains the strategy "reducing the size of public key", so it can reduce the size of public key of UOV from $\log_2(q) \cdot o \cdot \left(\frac{(o+v+1)\cdot(o+v+2)}{2}\right)$ bits to $\log_2(q) \cdot \left(o \cdot \left(\frac{(o+v+1)\cdot(o+v+2)-2D}{2}\right) + D\right)$ bits, where $D = \frac{v\cdot(v+1)}{2} + o \cdot v$. More details about this property we refer readers to [21].

Proposition 2. *Our proposed function is as secure as the basic function of UOV under current attack techniques.*

proof. We are going to prove it in the security aspects of both structural attack and the direct attack.

In the aspect of structural attack, there is still no valid proof to the original UOV scheme, and in fact after using the strategy of "reducing public key size", some of the structure of our public key gets lost, so the known attack methods under the structure of UOV is not useful in our scheme. However, one can easily figure out that our scheme is also based on both the MQ problem and the IP problem (the same as the original UOV scheme). Because of that we can charily claim that our proposed scheme is at least as secure as the basic scheme of UOV on this side.

In the aspect of direct attack, it is obvious that our scheme has the same security strength of UOV scheme on exhaustive attack. Now we may give formal proof to other direct attack such as Gröbner attack or hybrid attack.

In case of our scheme, the public key P is represented by a matrix $M_P = (B|C)$, where B is an $o \times D$ matrix from the vector b, and C is the remaining coefficients of P which is computed by composing Q and T. So the public key polynomials can be written as

$$p^{(k)} = \sum_{i,j=1}^{v} a_{ij}^{(k)} x_i x_j + \sum_{i=1,..,v,j==1,..,o} b_{ij}^{(k)} x_i x_j + \sum_{i,j=1}^{o} c_{ij}^{(k)} x_i x_j + \sum_{i=1}^{o+v} d_i^{(k)} x_i + e^{(k)},$$

for $k = 1, ..., o$, where coefficients $a_{ij}^{(k)}$ and $b_{ij}^{(k)}$ are elements of the matrix B and are chosen completely at random. Then a solution $x' = (x'_1, ..., x'_{o+v})$ can be seen as $x' = \left((x_i{}')_{i=1,...,v}, (x_j{}')_{j=1,...,o}\right)$ substituted in values $(x_j{}')_{j=1,...,o}$ for variables $(x_j)_{j=1,...,o}$. Thus one obtains a quadratic system with o equations and v variables $(x_i)_{i=1,...,v}$ of the form

$$\widetilde{p}^{(k)} = \sum_{i,j=1}^{v} a_{ij}^{(k)} x_i x_j + \sum_{i=1}^{v} \widetilde{d}_i^{(k)} x_i + \widetilde{e}^{(k)}, k = 1, ..., o.$$

Since the coefficients $a_{ij}^{(k)}$ are completely random, the system we need to solve is therefore

$$\widetilde{P}((x_i)_{i=1,...,v}) = h.$$

When the attacker uses Gröbner methods to solve, he/she would usually fix v variables firstly so as to end up with an $o \times o$ system. This is due to the fact that a random quadratic system with o equations and v variables over $GF(q)$ is expected to have q^{v-o} solutions, so by assigning values to some $v - o$ variables, the system still has o equations, but o variables, and is then found by using Gröbner basis techniques. Now the attacker who observes the matrix M_P and does not know which monomial was used is not able to figure out where the random part is. This is because even it is expected that the attacker can obtain the coefficients of monomials $x_i x_j, i, j = 1, ..., v$ via the reverse computation after computing Q from b and T, it can also be shown that the coefficients of the matrix C satisfy certain quadratic relations, so in principle, in order to do the reverse computation, one still has to go through all o-subsets of P. Without loss of generality, let $v = \alpha \cdot o, \alpha \geq 1$ and $P(x) = h$ be an $o \times (o + v)$ system of equations obtained above. The attacker fixes v variables to concrete values. Since $v = \alpha \cdot o$, we can expect on average $\frac{\alpha}{\alpha+1}v$ variables to be fixed in the set of monomials $x_i x_j, i, j = 1, ..., v$ and $\frac{1}{\alpha+1}v$ variables to be fixed in the set of monomials $x_i x_j, i, j = 1, ..., o$, so the number of remained variables not to be fixed in the set of monomials $x_i x_j, i, j = 1, ..., v$ is $\frac{a}{\alpha+1}o$. After substituting in values in the system $P(x) = h$, the attacker obtains a system which has o equations and $\frac{a}{\alpha+1}o$ variables. So, the attacker intrinsically faces solving a "random" system.

Hence, according to the analysis above, our proposed generation function is as secure as the basic generation function of UOV under current best attack techniques.

Proposition 3. *If the function of our scheme is $(t', \varepsilon')-$secure, our signature scheme is $(\varepsilon, t, q_{sig}, q_{hash})-$secure, where $\varepsilon(\lambda) \leq \frac{1}{(1-\frac{1}{q_{sig}+1})^{q_{sig}+1}} \cdot q_{sig} \cdot \varepsilon'(\lambda)$ and $t(\lambda) \geq t'(\lambda) - (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1))$, and t_{UOV} is the time to compute the UOV function.*

proof. Assume that in our scheme, there is an attacker $\mathcal{A}$ who takes as input a public key pk generated by $\mathsf{Gen}(1^\lambda)$, after at most $q_{hash}(\lambda)$ queries to the random oracle, $q_{sig}(\lambda)$ signature queries, and $t(\lambda)$ processing time, then outputs a valid signature with probability at least $\varepsilon(\lambda)$. We assume that $\mathcal{A}$ never repeats a hash query or a signature query. We build an inverter $\mathcal{B}$ which takes as input P generated via $(P, \cdot) \leftarrow Gen(1^\lambda)$ and a challenge $y \in_R k^n$, then finds a preimage x such that $P(x) = y$ at $t'(\lambda)$ processing time with probability at least $\varepsilon'(\lambda)$. We also call the inverter $\mathcal{B}$ the simulator. The simulator $\mathcal{B}$ starts running $\mathcal{A}$ for getting the public key. Then $\mathcal{A}$ makes hash oracle queries and signing queries. $\mathcal{B}$

will answer hash oracle queries and signing queries itself. We assume for simplicity that when $\mathcal{A}$ requests a signature on the message m, it has already made the corresponding hash query on m. If not, $\mathcal{B}$ continues and makes the hash query itself. $\mathcal{B}$ firstly sets a counter $i = 0$, a list $L \leftarrow \emptyset$, and selects a length of the random salt which is large enough.

Answering random oracle queries. When $\mathcal{A}$ makes a hash oracle query for m, $\mathcal{B}$ increase i by one, suppose $(m_i||r_i)$ is a random oracle query. $\mathcal{B}$ picks a random $d_i \in_R k^n$ then returns $h_i = P(d_i)$ and sets $L \leftarrow L \cup \{m_i, r_i, h_i\}$ with probability p , or returns $h_i = y$ and sets $L \leftarrow L \cup \{m_i, r_i, h_i\}$ with probability $1 - p$, where p is a fixed probability and will be determined later on.

Answering signing oracle queries. When $\mathcal{A}$ makes a signing query for m, it has already requested the hash for m, so $m = m_i$ for some i. the simulator $\mathcal{B}$ picks $r_i \in_R \{0,1\}^l$ and $d_i \in_R k^{o+v}$ and computes $y_i \leftarrow P(x_i)$. If $(m_i, r_i, \cdot) \in L$ then $\mathcal{B}$ aborts; otherwise $\mathcal{B}$ sets $L \leftarrow L \cup (m_i, r_i, y_i)$ and answers (x_i, r_i).

Output. Eventually, $\mathcal{A}$ outputs a forgery (x, r) of some message m. If the answer was y, we get x such that $P(x) = y$, thus $\mathcal{B}$ outputs the preimage x; otherwise $\mathcal{A}$ fails and aborts.

Analysis. The probability that $\mathcal{B}$ answers to all signature queries is at least $p^{q_{sig}}$. Then $\mathcal{B}$ outputs y (this means that $\mathcal{B}$ does not abort) with probability $1 - p$. So with probability at least

$$\alpha(p) = \varepsilon \cdot (p^{q_{sig}}) \cdot (1 - p),$$

$\mathcal{A}$ outputs a forgery of a certain message. The function $\alpha(p)$ is maximal for $p = 1 - 1/(q_{sig} + 1)$. Consequently we obtain:

$$\alpha(p_{max}) = \frac{1}{q_{sig}} \cdot (1 - \frac{1}{q_{sig}+1})^{q_{sig}+1},$$

so

$$\varepsilon(\lambda) \leq \frac{1}{(1-\frac{1}{q_{sig}+1})^{q_{sig}+1}} \cdot q_{sig} \cdot \varepsilon'(\lambda).$$

The runing time of $\mathcal{B}$ is the running time of $\mathcal{A}$ added to the time needed to compute h_i values, and so is at most $t + (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1)))$. Then

$$t \geq t' - (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1)).$$

Note that in this proposition, the exact security bound of $\varepsilon(\lambda)$ is better than that of [24].

Proposition 4. *If our signature scheme is $(\varepsilon, t, q_{sig}, q_{hash})-$secure, then our proxy signature scheme is $(\varepsilon'', t'', q'_{del}, q'_{sig})-$secure, where $\varepsilon''(\lambda) \leq 2 \cdot q'_{del} \cdot \varepsilon(\lambda)$.*

proof. We will show that if there exists an adversary $\mathcal{A}$ who can $(\varepsilon'', t'', q'_{del}, q'_{sig})$-break the proxy signature scheme, then we can construct an algorithm $\mathcal{B}$ who can $(\varepsilon, t, q_{sig}, q_{hash})$-break our proposed signature scheme.

The challenger is given a challenge public key $pk' = P_A{}''$ of our signature scheme and can access to a signing oracle $O_{sig}(m, \sigma)$ for the secret key sk' corresponding to pk'. Before interacting with the adversary $\mathcal{A}$ in the security game, $\mathcal{B}$ flips a fair coin c. If $c = 0$, $\mathcal{B}$ sets $pk^* = pk'$, and $sk^* = \emptyset$; otherwise, $\mathcal{B}$ generates a fresh key pair $(pk^*, sk^*) \leftarrow \mathsf{Gen}$ where $pk^* = P^*$, and chooses $i^* \in \{1, 2, ..., q''_{del}\}$. $\mathcal{B}$ runs $\mathcal{A}$ with input pk^*. As the challenger in the security game, $\mathcal{B}$ will maintain a set of lists $pskList(w)$ where each list $pskList(w)$ will hold all proxy keys generated by $\mathcal{B}$ with the warrant w. While running, $\mathcal{A}$ is allowed to make q'_{sig} ordinary signature queries and q'_{del} delegation queries which $\mathcal{B}$ will answer as follows:

Ordinary signature. On input m from $\mathcal{A}$, if $c = 0$, $\mathcal{B}$ simply makes query to his own signing oracle of our proposed signature scheme and obtains an answer σ; if $c = 1$, $\mathcal{B}$ generates a signature σ by running $\mathsf{Sig}(m, sk^*, pk^*)$ and returns σ to $\mathcal{A}$.

Delegation to u^*. $\mathcal{A}$ submits the delegation message $(w, cert, T'_{del}, P'_{del})$ where $w = (pk_{del}, pk', t)$ and $pk_{del} = P_{del}$. $\mathcal{B}$ then verifies whether both $cert = \mathsf{Sig}(w, sk_{del}, pk_{del})$ and $P'_{del} = P_{del} \circ T'_{del}$ are correct. If $c = 0$, $\mathcal{B}$ computes $T_p = T_{del} \circ T'_{del}$, $Q_p = Q_{del}$. Let $pk_p = P'_{del}$ and $sk_p = (Q_p, T_p)$. $\mathcal{B}$ then makes a query to his own signing oracle of our proposed signature scheme for $(w, cert, pk_p)$ and obtains a signature σ_{prx}. If $c = 1$ and this is not the i^*-th query, $\mathcal{B}$ similarly computes $T_p = T_{del} \circ T'_{del}$, $Q_p = Q_{del}$ where $pk_p = P'_{del}$ and $sk_p = (Q_p, T_p)$. Then $\mathcal{B}$ runs $\sigma_{prx} = \mathsf{Sig}((w, cert, pk_p), sk^*, pk^*)$. If $c = 1$ and this is the i^*-th query, $\mathcal{B}$ directly lets $pk_p = pk'$, $sk_p = \emptyset$ and runs $\sigma_{prx} = \mathsf{Sig}((w, cert, pk_p), sk^*, pk^*)$. Finally, $\mathcal{B}$ stores $((w, cert, pk_p, \sigma_{prx}), sk_p)$ in $pskList(w)$.

Delegation from u^*.

(1)*Delegation of sk^*.* On input w from $\mathcal{A}$ where $w = (pk^*, pk_{del}, t)$, $\mathcal{B}$ chooses randomly a bijective affine transformations T and computes $P'^* = P^* \circ T$. If $c = 0$, then $\mathcal{B}$ makes query to his signing oracle for w and obtains a signature $cert$. If $c = 1$, then $\mathcal{B}$ generates $cert$ by running $cert \leftarrow \mathsf{Sig}(w, sk^*, pk^*)$ and sends the delegation message $(w, cert, T, P'^*)$ to $\mathcal{A}$.

(2) *Self-delegation.* According to the input $w = (pk^*, pk^*, t)$ submitted by $\mathcal{A}$, $\mathcal{B}$ will do as follows. If $c = 0$ or $c = 1$ and this is not the i^*-th query, $\mathcal{B}$ chooses randomly a bijective affine transformations T, computes $P'^* = P^* \circ T$, makes query to his signing oracle for w and obtains a signature $cert$, and also queries for $(w, cert, (P^*, P'^*))$ and obtains σ_{prx}. If $c = 1$ and this is the i^*-th query, $\mathcal{B}$ directly lets $pk_p = pk'$, computes $\sigma_{prx} = \mathsf{Sig}((w, cert, pk_p), sk^*, pk^*)$ and stores $((w, cert, pk_p, \sigma_{prx}), sk_p)$.

Proxy signature. On input (w, m) from $\mathcal{A}$, $\mathcal{B}$ checks the proxy key in $pskList(w)$ and parses it as $sk_p, (w, cert, pk_p, \sigma_{prx})$. Then, if $c = 0$, $\mathcal{B}$ makes query to his signing oracle for m and obtains a signature σ_m; if $c = 1$, $\mathcal{B}$ computes $\sigma_m \leftarrow \mathsf{Sig}(m, sk_p, pk_p)$. Then $\mathcal{B}$ returns $(m, (w, cert, pkp, \sigma_{prx}), \sigma_m)$ to $\mathcal{A}$.

Proxy key exposure. On input w, $\mathcal{B}$ checks the proxy key in $pskList(w)$ and parses it as $sk_p, (w, cert, pk_p, \sigma_{prx})$. If $sk_p = \emptyset$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $(sk_p, (w, cert, pk_p, \sigma_{prx})$ to $\mathcal{A}$.

Note that unless an abort occurs, the value of c will be completely hidden from $\mathcal{A}$. If no abort occurs, $\mathcal{A}$ will eventually output a forgery. The forgeries are classified into two different categories:

Category A forgeries are either a valid type (i) forgery (m, σ), a valid type (ii) forgery $(m, \sigma_m, (w, cert, pk_p, \sigma_{prx}))$ where pk_p was not generated by $\mathcal{B}$, or a valid forgery type (iii) forgery $(m, \sigma_m, (w, cert, pk_p \sigma_{prx}))$ where w was not submitted to the ordinary signature query.

Category B forgeries are all valid forgeries that are not in category A, i.e. a type (ii) or type(iii) forgery where $\mathcal{B}$ has generated the public proxy key pk_p.

In the case $c = 0$, $\mathcal{B}$ sets $pk^* = pk'$. If $\mathcal{A}$ constructs a valid category B forgery, $\mathcal{B}$ will abort. otherwise, if $\mathcal{A}$ constructs a valid category A forgery, then If the forgery is of type (i), i.e. (m, σ), then $\mathcal{A}$ will not request a signature on m, and $\mathcal{B}$ will therefore not have submitted m to his own signature oracle. Hence, σ is a valid forgery of a signature of our signature scheme under the public key pk'. If the forgery is of type (ii), i.e. $(m, \sigma_m, (w, cert, pkp, \sigma_{prx}))$, σ_{prx} is a valid signature for $(w, cert, pk_p)$ under the public key pk', then $\mathcal{B}$ will not have submitted $(w, cert, pk_p)$ to his own signing oracle. Hence σ_{prx} will be a valid signature forgery of our signature scheme under the public key pk'. If the forgery is of type (iii), i.e. $(m, \sigma_m, (w, cert, pk_p, \sigma_{prx}))$, $cert$ will be a valid forgery for w, and $\mathcal{B}$ will therefore not have submitted w to his signing oracle. Hence cert is a valid forgery of a signature under the public key pk'.

In the case $c = 1$ where $\mathcal{B}$ inserts pk' as a proxy public key, if the forgery is category A forgery, $\mathcal{B}$ will abort. On the other hand, if the forgery is a category B forgery, then $\mathcal{B}$ outputs (m, σ_m) as a valid forgery for our underlying signature scheme. Otherwise, $\mathcal{B}$ aborts. Note that if $\mathcal{A}$ constructs such a forgery, then $\mathcal{A}$ will not have queried the proxy key $(w, cert, pkp, \sigma_{prx})$.

Analysis. Let E_1 be the event that $\mathcal{A}$ constructs a category A forgery, E_2 be the event that $\mathcal{A}$ constructs a category B forgery, and E_3 denote that $\mathcal{B}$ guesses the correct value of i^* in a category B forgery. The success probability of $\mathcal{A}$ is $Pr[E_1] + Pr[E_2]$.

So the success probability of $\mathcal{B}$ can be

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

Then $\varepsilon'' \leq 2 \cdot q'_{del} \cdot \varepsilon$.

6 Experiments and Comparisons

In this section, we analyze some important performance metrics of the proposed schemes and the baseline schemes on [17], [24], and [27]. All experiments are implemented in MAGMA [7] V2.12 running on a PC equipped with Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33GHZ CPU, and with 2GB RAM, and the operating system is Windows 7.

6.1 Performance and comparisons of our signature scheme

Suppose that the length of the prime p in binary expression is L bits, Table 1 shows the performance requirements by our proposed signature scheme and the baseline schemes.

Table 2 shows the security bound in the random oracle model of our proposed signature scheme and baseline schemes in [24].

Table 3 shows time requirements by running our proposed signature scheme and baseline schemes in [17] and [24].

Table 1. Performance Requirements by Ours and the Baseline Scheme

	Kipnis Scheme [17]	Sakumoto Scheme [24]	Ours
Public key size (bit)	$L \cdot o \cdot \left(\frac{(o+v+1)\cdot(o+v+2)}{2}\right)$	$L \cdot o \cdot \left(\frac{(o+v+1)\cdot(o+v+2)}{2}\right)$	$L \cdot \left(o \cdot \left(\frac{(o+v+1)\cdot(o+v+2)-2D}{2}\right) + D\right)$
Computation on key generation	$O(o \cdot n^2)$	$O(o \cdot n^2)$	$O(D^2 + o \cdot n^2)$
Computation on signature generation	$O(o \cdot v + S)$	$O(o \cdot v + S)$	$O(o \cdot v + S)$
Computation on signature verify	$O(n)$	$O(n)$	$O(n)$

Notation for Table 1:
o,v: the number of Oil and Vinegar variables respectively;
n: $n = v + o$;
D: $D = \frac{v\cdot(v+1)}{2} + o \cdot v$;
S: average time required by a Gaussian Elimination function to solve o linear equations in o variables as unknowns.

Table 2. Security Bound in the Random Oracle of Ours and the Baseline Scheme

	Sakumoto Scheme [24]	Ours
Security bound($\varepsilon(\lambda)$)	$\frac{(q_{hash}+q_{sig}+1)}{1-(q_{hash}+q_{sig})q_{sig}2^{-l}} \cdot \varepsilon'(\lambda)$	$\frac{1}{(1-\frac{1}{q_{sig}+1})^{q_{sig}+1}} \cdot q_{sig} \cdot \varepsilon'(\lambda)$
Security bound($t(\lambda)$)	$t'(\lambda) - (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1))$	$t'(\lambda) - (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1))$

Notation for Table 2:
q_{hash}: number of queries to the random oracle;
q_{sig}: number of signature queries;
l: the length of the random salt;
t_{UOV}: the time to compute the UOV function.

6.2 Performance and comparisons of our proxy signature scheme

The complexity and running time of each procedure of our proxy signature scheme, and comparison with Tang scheme [27] is shown in Table 4.

Table 3. Running Time Requirements by Ours and Baseline Schemes

	Kipnis Scheme [17]	Sakumoto Scheme [24]	Ours
Running time of key generation(ms)	18330	18330	83585
Running time of signature generation(ms)	187	187	187
Running time of signature verify(ms)	31	31	31

Table 4. Performance Requirements by Ours and Compared with Tang Scheme

	Our Time Complexity	Our Running Time (ms)	Time Complexity of Tang scheme [27])
Initialization	$\mathcal{O}(2 \cdot (D^2 + o \cdot n^2))$	158685	$\mathcal{O}(2m^2 + 2n^2 + 2mn^2)$
Delegation computation by Alice	$\mathcal{O}((o+1) \cdot n^2 + S)$	13853	$\mathcal{O}(3m^3 + 3n^3 + 2mn^3 + qmn^3)$
Proxy key generation computation by Bob	$\mathcal{O}(n^2 + S)$	187	$\mathcal{O}(2m^3 + 2n^3 + 3mn^3 + 2qmn^3)$
Computation on proxy signature generation	$\mathcal{O}(ov + S)$	187	$\mathcal{O}(qmn^3)$
Computation on signature verify	$\mathcal{O}(3n)$	31	$\mathcal{O}(3qmn^3)$

Notation for Table 4:
o,v: the number of Oil and Vinegar variables respectively in our scheme;
m: the number of polynomials in a scheme (in our scheme it is equal to o);
n: the number of variables in polynomials(in out scheme, $n = v + o$);
D: $D = \frac{v \cdot (v+1)}{2} + o \cdot v$;
S: average time required by a Gaussian Elimination function to solve o linear equations in o variables.
q: the length of output bits of the hash function.

7 Conclusion

In this paper, after introducing the exact security of UOV signature, we propose a combined signature scheme based on UOV, which is consist of two widely used strategies: reducing the size of public key, and making the trapdoor function uniform. Then we formally show that our combined scheme can not only reduce the public key size of the UOV signature scheme but also can provide exact security proof. Compared with the basic UOV scheme, the exact security bound of our signature scheme satisfies $\varepsilon(\lambda) \leq \frac{1}{(1-\frac{1}{q_{sig}+1})^{q_{sig}+1}} \cdot q_{sig} \cdot \varepsilon'(\lambda)$ and $t(\lambda) \geq t'(\lambda) - (q_{hash} + q_{sig} + 1)(t_{UOV} + O(1))$, which is a better security bound than original UOV. where q_{sig} is number of signature queries, q_{hash} is the number of queries to the random oracle, and t_{UOV} is the time to compute the UOV function. What's more, we propose a proxy signature scheme based on our proposed signature scheme and also give a strict security proof for our proxy signature scheme.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. U1135004 and 61170080, Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011), and High-level Talents Project of Guangdong Institutions of Higher Education (2012).

References

1. Bardet, M., Faugère, J.C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proc. Intl Conference on Polynomial System Solving pp. pp.71–74 (2004)
2. Bellare, M., Micali, S.: How to sign given any trapdoor permutation. J. ACM 39(1), 214–233 (Jan 1992), http://doi.acm.org/10.1145/147508.147537
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73. CCS '93, ACM, New York, NY, USA (1993), http://doi.acm.org/10.1145/168588.168596
4. Bellare, M., Rogaway, P.: The exact security of digital signatures-how to sign with RSA and Rabin. In: Maurer, U. (ed.) Advances in Cryptology of EUROCRYPT 1996, Lecture Notes in Computer Science, vol. 1070, pp. 399–416. Springer Berlin Heidelberg (1996), http://dx.doi.org/10.1007/3-540-68339-9_34
5. Berthomieu, J., Faugère, J.C., Perret, L.: Polynomial-time algorithms for quadratic isomorphism of polynomials. CoRR abs/1307.4974 (2013)
6. Bettale, L., Faugère, J.C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Math. Cryptology pp. pp. 177–197 (2009)
7. BOSMA, W., CANNON, J., PLAYOUST, C.: The magma algebra system I: The user language. Journal of Symbolic Computation 24(3-4), 235 – 265 (1997), http://www.sciencedirect.com/science/article/pii/S074771719690125X
8. Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in F_2. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, Lecture Notes in Computer Science, vol. 6225, pp. 203–218. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15031-9_14
9. Bulygin, S., Petzoldt, A., Buchmann, J.: Towards provable security of the Unbalanced Oil and Vinegar signature scheme under direct attacks. In: Gong, G., Gupta, K. (eds.) Progress in Cryptology - INDOCRYPT 2010, Lecture Notes in Computer Science, vol. 6498, pp. 17–32. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-17401-8_3
10. Coron, J.S.: On the exact security of full domain hash. In: Bellare, M. (ed.) Advances in Cryptology -CRYPTO 2000, Lecture Notes in Computer Science, vol. 1880, pp. 229–235. Springer Berlin Heidelberg (2000), http://dx.doi.org/10.1007/3-540-44598-6_14
11. Courtois, N.: Generic attacks and the security of Quartz. In: Desmedt, Y. (ed.) Public Key Cryptography -PKC 2003, Lecture Notes in Computer Science, vol. 2567, pp. 351–364. Springer Berlin Heidelberg (2002), http://dx.doi.org/10.1007/3-540-36288-6_26
12. Ding, J., Gower, J., Schmidt, D.: Multivariate public key cryptosystems (2006)
13. Ding, J., Yang, B.Y.: Multivariate public key cryptography. In: Bernstein, D., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 193–241. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-540-88702-7_6
14. Faugère, J.C., Perret, L.: On the security of UOV. Cryptology ePrint Archive Report 2009/483 (2009), http://eprint.iacr.org/
15. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, 61 – 88 (1999), http://www.sciencedirect.com/science/article/pii/S0022404999000055

16. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. Extended version available at http://citeseer.ist.psu.edu/231623.html
17. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) Advances in Cryptology -EUROCRYPT '99, Lecture Notes in Computer Science, vol. 1592, pp. 206–222. Springer Berlin Heidelberg (1999), http://dx.doi.org/10.1007/3-540-48910-X_15
18. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) Advances in Cryptology -CRYPTO '98, Lecture Notes in Computer Science, vol. 1462, pp. 257–266. Springer Berlin Heidelberg (1998), http://dx.doi.org/10.1007/BFb0055733
19. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security. pp. 48–57. CCS '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/238168.238185
20. Patarin, J.: The oil and vinegar signature scheme. In: Dagstuhl Workshop on Cryptography. vol. 80 (1997)
21. Petzoldt, A., Bulygin, S., Buchmann, J.: A multivariate signature scheme with a partially cyclic public key. In: Proceedings of SCC pp. 229–235 (2010)
22. Petzoldt, A., Bulygin, S., Buchmann, J.: Linear recurring sequences for the UOV key generation. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) Public Key Cryptography -PKC 2011, Lecture Notes in Computer Science, vol. 6571, pp. 335–350. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-19379-8_21
23. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Rogaway, P. (ed.) CRYPTO 2011 6841, pp. 706–723 (2011)
24. Sakumoto, K., Shirai, T., Hiwatari, H.: On provable security of UOV and HFE signature schemes against chosen-message attack. In: Yang, B.Y. (ed.) Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 7071, pp. 68–82. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-25405-5_5
25. Schuldt, J., Matsuura, K., Paterson, K.: Proxy signatures secure against proxy key exposure. In: Cramer, R. (ed.) Public Key Cryptography -PKC 2008, Lecture Notes in Computer Science, vol. 4939, pp. 141–161. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-78440-1_9
26. Tang, S., Xu, L.: Proxy signature scheme based on isomorphisms of polynomials. In: Xu, L., Bertino, E., Mu, Y. (eds.) Network and System Security, Lecture Notes in Computer Science, vol. 7645, pp. 113–125. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-34601-9_9
27. Tang, S., Xu, L.: Towards provably secure proxy signature scheme based on isomorphisms of polynomials. Future Generation Computer Systems 30(0), 91 – 97 (2014), http://www.sciencedirect.com/science/article/pii/S0167739X13001179