Small Public Keys and Fast Verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Systems

Albrecht Petzoldt[1], Enrico Thomae[2], Stanislav Bulygin[1], and Christopher Wolf[2]

[1] Technische Universität Darmstadt and
Center for Advanced Security Research Darmstadt (CASED)
apetzoldt@cdc.informatik.tu-darmstadt.de, Stanislav.Bulygin@cased.de
[2] Horst Görtz Institute for IT-security
Faculty of Mathematics
Ruhr-University of Bochum, 44780 Bochum, Germany
Enrico.Thomae@rub.de, Christopher.Wolf@rub.de,
chris@Christopher-Wolf.de

Abstract. Security of public key schemes in a post-quantum world is a challenging task—as both RSA and ECC will be broken then. In this paper, we show how post-quantum signature systems based on $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic ($\mathcal{MQ}$) polynomials can be improved up by about 9/10, and 3/5, respectively, in terms of public key size and verification time. The exact figures are 88% and 59%. This is particularly important for small-scale devices with restricted energy, memory, or computational power. In addition, we provide evidence that this reduction does not affect security and that it is also optimal in terms of possible attacks. We do so by combining the previously unrelated concepts of reduced and equivalent keys. Our new scheme is based on the so-called *Unbalanced Oil and Vinegar* class of $\mathcal{MQ}$-schemes. We have derived our results mathematically and verified the speed-ups through a C++ implementation.

Keywords: Multivariate Quadratic Cryptography, Post-Quantum Cryptography, Implementation, Unbalanced Oil and Vinegar Signature Scheme.

1 Introduction

When finding an old sonnet of Shakespeare, we can usually determine its validity accurately by checking the wording, the ink, the paper, and so on. Similar techniques apply in disputes over last wills - or other documents of historical or financial interest. Even if they are several decades old, we can fairly certainly determine if they have been written by the person in question and sometimes even date them accurately.

For digital documents, this is a much more difficult task. They are electronically signed with the help of so-called *digital signature schemes*. The ones widely used today are Digital Signature Algorithms (DSAs) based on RSA and elliptic curve cryptography (ECDSA). Unfortunately, all these schemes are broken

B. Preneel and T. Takagi (Eds.): CHES 2011, LNCS 6917, pp. 475–490, 2011.
© International Association for Cryptologic Research 2011

if large enough quantum computers will be built. The reason is the algorithm of Shor which breaks all cryptographic algorithms based on the difficulty of factoring and the discrete logarithm (DL) problem [16]. This covers DL over numbers, RSA, and ECDSA. Even if unlikely now, quantum computers may be available in the medium future and are hence a concern for long-term-validity of authentication data. We must be sure that a document signed today is not repudiated 50 years later. Likewise, we do not want a signature that is generated today to be forged in the future. So to guard security even in the presence of quantum computers, post-quantum cryptography is needed and has hence become a vital research area [1]. One possible solution in this context is the so-called *Multivariate Quadratic cryptography*. It is widely believed that it is secure against attacks with quantum computers.

In addition, Multivariate Quadratic (or $\mathcal{MQ}$ for short) signature schemes have nice properties in terms of speed of signature generation and verification which make them superior to DL, RSA and ECDSA. Note that ECDSA is the most efficient of the three. However, even when comparing to signature generation in $\mathcal{MQ}$ and ECC, the former are a factor of 2 - 50 faster on FPGA than the latter [3]. Similar results have been demonstrated for comparison with RSA and ECC in software [21], [4], [5]. One of the main reasons for this higher efficiency is the comparably small finite fields, *e.g.* $\mathbb{F}_{2^8}$ which allows for efficient hardware and software implementations. The other operations usually boil down to vector-matrix functions, which can be implemented efficiently, too. As an immediate consequence, we can use $\mathcal{MQ}$ schemes in restricted devices, *i.e.* with low energy or computational power.

Another point is the high flexibility of $\mathcal{MQ}$-schemes. This allows for the use of sparse polynomials in the *private key* as done in the TTS schemes of Yang, Chen, and Chen [21]. This leads both to a significant reduction of the time needed for signature generation, as well as for the size of the private key. Another way to reduce the private key is by choosing the coefficients of the private maps from smaller fields (*e.g.* $\mathbb{F}_{16}$ instead of $\mathbb{F}_{256}$), [4]. In addition, we want to mention the so-called *similar keys* which exploit linear relations between public and private key [9]. However, they are not applicable to schemes like UOV. Finally, one research direction deals with reducing the public key directly. In [13, 14] it is shown how to reduce the public key size of the UOV scheme by choosing public coefficients in a structured way, cf. Section 3.

1.1 Achievement

Combining two previously unrelated ideas, we deal with reducing the size of the public key. For $\mathcal{MQ}$ schemes like *Unbalanced Oil and Vinegar* (UOV - see below), typical choices of parameters lead to around 80 kByte for the public key. We use the approach of [14] to bring this size down to about 9 kB. We show that we can use this idea to reduce the verification time, too. By choosing them partially to be 0 or 1 only, verification time is reduced by up to 59%. This way, $\mathcal{MQ}$ verification can be performed in low-power, low-energy devices. For example for mobile devices, we can easily imagine a scenario where a server signs data which

needs to be verified by a (comparably restricted) phone. As further contribution, we give arguments that this reduction in size does *not* affect security. This is due to an observation regarding equivalent keys of [19, 20].

In addition, our modification also works for restricting the choice of the coefficients. Using Turán graphs we demonstrate that this further reduction in size and verification time resists all know attacks.

1.2 Organization

The structure of this paper is as follows: After giving some introduction in Section 1, we continue with the background on $\mathcal{MQ}$-schemes and in particular the UOV in Section 2. In Section 3, we review the cyclic construction from [13]. This is followed by security considerations regarding cyclic keys in UOV in Section 4. Using these results, we outline our new constructions, its implementation, efficiency, and security implications in Section 5. The paper concludes with Section 6. Some background on Turán graphs and how this relates to our monomial ordering in Subsection 5.3 can be found in the full version of this paper [15].

2 $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

The main idea behind $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic cryptography is to choose a system $\mathcal{F}$ of m quadratic polynomials in n variables which can be easily inverted. Here $\mathcal{F}$ is called the *central map*. In addition, we need invertible affine maps S and T to hide the structure of the central map $\mathcal{F}$. The public key of the cryptosystem is now composed as $\mathcal{P} = T \circ \mathcal{F} \circ S$. For a secure $\mathcal{MQ}$-system, $\mathcal{P}$ must be difficult to invert. The private key consists of $(\mathcal{F}, T, S)$ and therefore allows efficient inversion of $\mathcal{P}$. More information on $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic schemes can be found in [6, 18].

2.1 Notation

Solving non-linear systems of m equations in n variables over a finite field is a difficult problem in general. Restricting it to the seemingly easy case of quadratic equations is still difficult. Actually this problem is also known as $\mathcal{MQ}$-problem which is proven to be NP-hard in the worst-case [8], even over $\mathbb{F}_2$.
Let $\mathcal{P}$ be an $\mathcal{MQ}$ system of the form

$$\begin{aligned} p^{(1)}(x_1, \ldots, x_n) &= 0 \\ p^{(2)}(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ p^{(m)}(x_1, \ldots, x_n) &= 0, \end{aligned} \tag{1}$$

with

$$p^{(k)}(x_1, \ldots, x_n) := \sum_{1 \le i \le j \le n} p_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} p_i^{(k)} x_i + p_0^{(k)} \quad (k = 1, \ldots, m). \tag{2}$$

Let $\pi^{(k)}$ be the coefficient vector of $p^{(k)}(x_1, \ldots, x_n)$ w.r.t. graded lexicographic ordering of monomials, *i.e.*

$$\pi^{(k)} = (p_{11}^{(k)}, p_{12}^{(k)}, \ldots, p_{1n}^{(k)}, p_{22}^{(k)}, p_{23}^{(k)}, \ldots, p_{nn}^{(k)}, p_1^{(k)}, \ldots, p_n^{(k)}, p_0^{(k)}). \tag{3}$$

Let M_P be the corresponding coefficient matrix

$$M_P := \begin{pmatrix} \pi^{(1)} \\ \vdots \\ \pi^{(m)} \end{pmatrix}. \tag{4}$$

Note that the ordering of monomials (and thus coefficients) in the matrix M_P does not necessarily have to be graded lexicographic ordering. We may want to order monomials of the public key in a certain way. Therewith, the ordering of coefficients (columns of M_P) is then changed accordingly.

2.2 Unbalanced Oil and Vinegar

In this subsection we introduce the Oil and Vinegar Signature Scheme, which was proposed by J. Patarin in [12]. Let $\mathbb{F}_q$ be a finite field. Denote the number of oil variables by $o \in \mathbb{N}$, the number of vinegar variables by $v \in \mathbb{N}$ and set $n := o+v$. Let $V := \{1, \ldots, v\}$ and $O := \{v+1, \ldots, n\}$ denote the sets of indices of vinegar and oil variables. The private key $\mathcal{F} := (f^{(1)}, \ldots, f^{(o)})$ is defined by

$$f^{(k)}(u_1, \ldots, u_n) := \sum_{i \in V, j \in O} f_{ij}^{(k)} u_i u_j + \sum_{i,j \in V, i \leq j} f_{ij}^{(k)} u_i u_j \quad (k = 1, \ldots, o)\,. \tag{5}$$

Remark: We omit the linear part of $\mathcal{F}$ and the constant part of S, because it was shown in [11, 10] that it does not contribute to the security of UOV.

For the inversion of the map $\mathcal{F}$ it is important that the variables in $f^{(k)}$ are not completely mixed, *i.e.* oil variables are only multiplied by vinegar variables and never by oil variables. This construction leads to an efficient way to invert $\mathcal{F}$. If we assign arbitrary values to the vinegar variables we obtain a system of o linear equations in o variables. With high probability this system has a solution. If not we try again with a different choice for the vinegar variables $x_1, \ldots, x_v$. In the public key $\mathcal{P}$, the central map $\mathcal{F}$ is hidden by composing it with a linear map $S : \mathbb{F}_q^n \to \mathbb{F}_q^n$, *i.e.* $\mathcal{P} := \mathcal{F} \circ S$.

Note that, in opposite to other multivariate schemes, the second linear map T is not needed for the security of UOV. So it can be dropped.

Figure 1 shows the signature generation and verification process for UOV.

Signature Generation: To sign a document d, one uses a hash function $\mathcal{H} : \mathbb{F}_q^\star \to \mathbb{F}_q^m$ to compute the hash value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}_q^m$. After that one computes first $\mathbf{u} := \mathcal{F}^{-1}(\mathbf{h})$ and then $\mathbf{x} := S^{-1}(\mathbf{u})$. The signature of the document d is

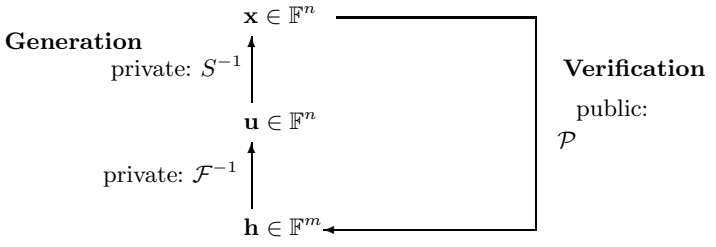

Fig. 1. Signature generation and verification for UOV

$\mathbf{x} \in \mathbb{F}_q^n$. In a slight abuse of notation we write $\mathcal{F}^{-1}(\mathbf{h})$ for finding one (of possibly many) pre-image of $\mathbf{h}$ under $\mathcal{F}$.

Signature Verification: To verify the authenticity of a signature, one computes the hash value $\mathbf{h}$ of the corresponding document and the value $\mathbf{h}' = \mathcal{P}(\mathbf{x})$. If $\mathbf{h} = \mathbf{h}'$ holds, the signature is accepted, otherwise rejected.

In his original paper [12], Patarin suggested to use $o = v$ (Balanced Oil and Vinegar - OV). After this scheme was broken by Kipnis and Shamir in [11], it was proposed in [10] to use $v \geq 2o$ (Unbalanced Oil and Vinegar (UOV)). UOV parameters $q = 2^8$, $(o, v) = (26, 52)$ give 80-bit security against the most efficient attacks currently known [2].

3 Reviewing Cyclic Keys

In this section we review the approach of [13] to create a UOV-based scheme with a partially cyclic public key. Remember that, in the case of the Unbalanced Oil and Vinegar signature scheme [10], the public key $\mathcal{P}$ is given as the concatenation of the central UOV-map $\mathcal{F}$ and a linear invertible map S, *i.e.* $\mathcal{P} = \mathcal{F} \circ S$.

In [13] it is observed, that this equation (after fixing the linear map S), leads to a linear relation between the coefficients of the quadratic monomials of $\mathcal{P}$ and $\mathcal{F}$ of the form

$$p_{ij}^{(k)} = \sum_{r=1}^{n} \sum_{s=r}^{n} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)}, \tag{6}$$

where $p_{ij}^{(k)}$ and $f_{ij}^{(k)}$ are the coefficients of $x_i x_j$ in the k-th component of $\mathcal{P}$ and $\mathcal{F}$ respectively and the α_{ij}^{rs} are given as

$$\alpha_{ij}^{rs} = \begin{cases} s_{ri} \cdot s_{si} & (i = j) \\ s_{ri} \cdot s_{sj} + s_{rj} \cdot s_{si} & \text{otherwise} \end{cases}. \tag{7}$$

Here $s_{ij} \in \mathbb{F}_q$ denote the coefficients of the linear map S. Let $D := \frac{v \cdot (v+1)}{2} + ov$ be the number of non-zero quadratic terms in any component of $\mathcal{F}$ and $D' := \frac{n \cdot (n+1)}{2}$ be the number of quadratic terms in the public polynomials. Let

M_P and M_F be the coefficient matrices of $\mathcal{P}$ and $\mathcal{F}$ respectively (w.r.t. graded lexicographic ordering of monomials). The matrices M_P and M_F are divided into submatrices as shown in Figure 2. Note that, due to the absence of oil $\times$ oil terms in the central polynomials, we have a block of zeros on the right side of M_F.

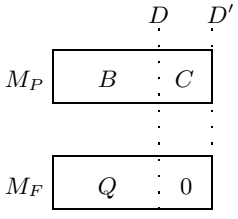

Fig. 2. Layout of the matrices M_P and M_F

Furthermore, the authors of [13] defined the so called transformation matrix $A_{UOV} \in \mathbb{F}_q^{D \times D}$ containing the coefficients α_{ij}^{rs} of equation (6), *i.e.* $A_{UOV} = \left(\alpha_{ij}^{rs}\right)$ for $1 \leq r \leq v, r \leq s \leq n$ for the rows and $1 \leq i \leq v, i \leq j \leq n$ for the columns.

$$A_{UOV} = \begin{pmatrix} \alpha_{11}^{11} & \alpha_{12}^{11} & \dots & \alpha_{vn}^{11} \\ \alpha_{11}^{12} & \alpha_{12}^{12} & \dots & \alpha_{vn}^{12} \\ \vdots & & & \vdots \\ \alpha_{11}^{vn} & \alpha_{12}^{vn} & \dots & \alpha_{vn}^{vn} \end{pmatrix}. \tag{8}$$

With this notation, equation (6) yields

$$B = Q \cdot A_{UOV}. \tag{9}$$

If matrix A_{UOV} is invertible, this equation has a solution for Q. Experiments indicate that this condition is fulfilled with high probability. By solving equation (9) for Q, the authors of [13] were able to insert a partially circulant matrix B into the UOV public key. By doing so, they reduced the public key size of the scheme by a factor of 6. After choosing matrix B, we can use Algorithm 1 to compute the corresponding key.

4 Security of UOV

Due to equivalent keys [19, 20] UOV contains a lot of redundancy. We show which part of the public key is important for security and which part can be chosen such that the public key gets as small as possible.

It is rather intuitive that the linear and constant part of the public key do not provide extra security because we can easily separate them from the quadratic part. This was previously exploited by Kipnis and Shamir in their cryptanalysis of (balanced) Oil and Vinegar [11]. But it is quite surprising that also a fraction of the quadratic part is not essential for security. This is implied by the observation of equivalent keys by Wolf and Preneel [19, 20].

Algorithm 1. Alternative Key Generation for UOV schemes

1: Choose an $o \times D$ matrix B (e.g. partially circulant or generated by an LRS).
2: Choose randomly a linear map $\mathcal{S}$ (represented by an $n \times n$-matrix S). If S is not invertible, choose again.
3: Compute for S the corresponding transformation matrix A_{UOV} (using equations (7) and (8)). If A_{UOV} is not invertible, go back to step 2.
4: Solve the linear system given by equation (9) to get the matrix Q and there with the coefficients of the central polynomials.
5: Compute the public key as $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$.

Definition 1. *Let $(\mathcal{F}, S)$ and $(\mathcal{F}', S')$ be two UOV private keys. They are called equivalent if they result in the same UOV public key, i.e. $\mathcal{F} \circ S = \mathcal{F}' \circ S' =: \mathcal{P}$.*

The set of all private keys resulting in a given public key $\mathcal{P}$ is denoted by $EQ_{\mathcal{P}}$.

In order to produce equivalent keys we use the following transformation Ω on the variables $\mathbf{u}$ that preserves the structure of $\mathcal{F}$.

$$\Omega = \begin{pmatrix} \Omega^{(1)}_{v\times v} & 0 \\ \Omega^{(2)}_{o\times v} & \Omega^{(3)}_{o\times o} \end{pmatrix} \text{ resp. } \Omega = (\omega_{ij})_{i,j=1}^{n} \tag{10}$$

Let $\mathcal{F}(\mathbf{u}) := (f^{(1)}(\mathbf{u}), \ldots, f^{(o)}(\mathbf{u}))$ be a UOV central map (i.e. no quadratic cross terms in oil variables). Let $\mathbf{u} = \Omega \cdot \mathbf{u}'$. The vinegar variables $u_1, \ldots u_v$ are computed as sums of vinegar variables $u'_1, \ldots, u'_v$. Therefore we get (for $k = 1, \ldots, o$):

$$\begin{aligned} f^{(k)}(\mathbf{u}) &= \sum_{i,j\in V} f^{(k)}_{ij} u_i u_j + \sum_{i\in V, j\in O} f^{(k)}_{ij} u_i u_j \\ &= \sum_{i,j\in V} f^{(k)}_{ij} \left(\sum_{l\in V} \omega_{il} u'_l\right) \cdot \left(\sum_{m\in V} \omega_{jm} u'_m\right) \\ &+ \sum_{i\in V, j\in O} f^{(k)}_{ij} \left(\sum_{l\in V} \omega_{il} u'_l\right) \cdot \left(\sum_{m\in V} \omega_{jm} u'_m + \sum_{m\in O} \omega_{jm} u'_m\right) \\ &= \sum_{i,j\in V} \widetilde{f^{(k)}_{ij}} u'_i u'_j + \sum_{i\in V, j\in O} \widetilde{f^{(k)}_{ij}} u'_i u'_j \text{ for some } \widetilde{f^{(k)}_{ij}}(f_{ij}, \omega_{lm}). \end{aligned}$$

Due to $\mathcal{F} \circ S = \mathcal{F} \circ \Omega^{-1} \circ \Omega \circ S$ both $(\mathcal{F}, S)$ and $(\mathcal{F} \circ \Omega^{-1}, \Omega \cdot S)$ are equivalent private keys for the public key $\mathcal{P}$.

Lemma 1. *For every UOV public key $\mathcal{P}$ there exists a UOV private key $(\mathcal{F}, S) \in EQ_{\mathcal{P}}$ such that S has the form*

$$S = \begin{pmatrix} I & \widetilde{S} \\ 0 & I \end{pmatrix} \tag{11}$$

for some $(v \times o)$ matrix $\widetilde{S}$ (except for permutations of rows and columns).

Proof. Let Ω be of form (10) and $\mathcal{F}$ and

$$S = \begin{pmatrix} S^{(1)}_{v\times v} & S^{(2)}_{v\times o} \\ S^{(3)}_{o\times v} & S^{(4)}_{o\times o} \end{pmatrix}$$

an arbitrary private key. There exists a permutation of rows and columns such that $S^{(1)}$, $S^{(4)}$ and $I - S^{(3)}{S^{(1)}}^{-1}S^{(2)}{S^{(4)}}^{-1}$ are invertible[1]. Now we choose Ω such that $\Omega^{(1)}S^{(1)} = I$, $\Omega^{(2)}S^{(2)} + \Omega^{(3)}S^{(4)} = I$ and $\Omega^{(2)}S^{(1)} + \Omega^{(3)}S^{(3)} = 0$, i.e. $\Omega^{(1)} = {S^{(1)}}^{-1}$, $\Omega^{(3)} = {S^{(4)}}^{-1} \cdot (I - S^{(3)}{S^{(1)}}^{-1}S^{(2)}{S^{(4)}}^{-1})^{-1}$ and $\Omega^{(2)} = -\Omega^{(3)}S^{(3)}{S^{(1)}}^{-1}$. □

For the following, we assume w.l.o.g. that the linear map S has the form (11), since the further analysis is not changed by row and column permutations.
The next lemma shows that the vinegar × vinegar coefficients in the public key do not hide any information about the secret map.

Lemma 2. *In the case of S having the form* (11) *we get $f^{(k)}_{ij} = p^{(k)}_{ij}$ for $i, j \in \{1, \ldots, v\}$ and $k \in \{1, \ldots, o\}$.*

Proof. We consider for $k \in \{1, \ldots o\}$ the quadric forms of $p^{(k)}$ and $f^{(k)}$, i.e. $MP^{(k)} = S^T \cdot MF^{(k)} \cdot S$. For S having the form (11) this yields $MP^{(k)}_1 = MF^{(k)}_1$ $(k = 1, \ldots, o)$. □

For a key recovery attack it is sufficient to find any of the equivalent keys. Thus an attacker can search for a private key, with S of the form (11). This means we can assume that the attacker actually knows all coefficients of squared vinegar variables in the private map. This does not effect the overall knowledge of the attacker. So the $p^{(k)}_{ij}$ with $i, j \in V$ in the public map do not hide any secret and thus we can choose them in an arbitrary way.

Proposition 1. *The first $\frac{v(v+1)}{2}$ coefficients of each public polynomial do not provide any security in the sense of key recovery attacks. Arbitrarily fixing these coefficients does not give advantage to an attacker who wants to recover the whole private key.*

By equation (9) we are even able to choose the first $\frac{v(v+1)}{2} + ov$ coefficients of each public polynomial of a special form and thus save memory. Proving the security of this construction is not as obvious as in the latter construction. We have to show that additionally fixing ov coefficients does not give advantage to an attacker in the sense of key recovery attacks

Usually, we fix the coefficients of the central polynomials $\mathcal{F}$ and the linear map S and then compute the public polynomials $\mathcal{P}$. However, for our construction we turn things around by *first* fixing parts of the public polynomials and *then* computing the central polynomials. Intuitively, this should be equally secure. We capture this in the following proposition.

[1] Our experiments showed, that these three conditions are fulfilled for 99.2 % of all UOV private keys without changing rows and columns.

Proposition 2. *Let the $o \times D$ matrix B be an MDS matrix (i.e. every choice of o columns leads to an invertible submatrix). Then, fixing the first $\frac{v(v+1)}{2} + ov$ coefficients of each public polynomial to the elements of B does not give the attacker any advantage in the sense of key recovery attacks.*

Proof. We start our proof with equation (9)

$$B = Q \cdot A_{UOV}.$$

For S having the form (11) we can write this as

$$(B_1|B_2) = (F_1|F_2) \cdot \begin{pmatrix} I & \Sigma \\ 0 & 1 \end{pmatrix} \tag{12}$$

with a $o \times o \cdot v$ matrix F_2 containing the coefficients f_{ij}, $(i \in \{1, \ldots, v\},\ j \in \{v+1, \ldots, n\})$ and a $\frac{v \cdot (v+1)}{2} \times o \cdot v$ matrix Σ linear in the elements of S. This leads to $F_1 = B_1$ and

$$B_2 = F_1 \cdot \Sigma + F_2 = B_1 \cdot \Sigma + F_2. \tag{13}$$

Equation (13) yields $o \cdot o \cdot v$ linear equations in the $(o+1) \cdot o \cdot v$ unknowns s_{ij} and $f_{ij}^{(k)}$. We can use the last $o \cdot v$ of these equations to eliminate the s_{ij} and get $(o-1) \cdot o \cdot v$ linear relations between the coefficients $f_{ij}^{(k)}$. If the map

$$S \mapsto B_2 - B_1 \cdot \Sigma(S) := F_2$$

is injective, there remain exactly $o \cdot v$ coefficients $f_{ij}^{(k)}$, which have to be guessed correctly to obtain a valid private key. The injectivity follows from the fact that all square submatrices of B_1 are invertible, which is the property we obtain by using an MDS matrix[2].

This is exactly the same situation we obtain for the standard UOV scheme. Therefore, fixing the matrix B to an MDS-matrix does not make key recovery attacks easier. □

We use this observation by proposing a variant of the UOV, which is provably as secure as the original scheme, but reduces the public key size by a huge factor.

In comparison to the case of Algorithm 1 the $(o \times D)$ matrix B is now fixed to an MDS matrix and system parameter of the algorithm. In the remainder of this paper, we refer to this scheme as Compressed UOV (see Algorithm 2).

5 The New Construction

We are now investigating, how much additional structure we can hide in B to speed up the verification process. We do this by choosing the elements of the matrix B from the ground field $\mathbb{F}_2$. In order to make sure that message recovery attacks are still difficult, we have to choose the ordering of monomials appropriately, as explained in Subsection 5.3.

[2] For large enough q, e.g. $q = 2^8$, the matrix B with coefficients chosen uniformly at random is MDS with high probability.

Algorithm 2. Key Generation for Compressed UOV

1: Choose randomly a linear map S (represented by an $n \times n$-matrix S). If S is not invertible, choose again.
2: Compute for S the corresponding transformation matrix A_{UOV} (using equations (7) and (8)). If A_{UOV} is not invertible, go back to step 1.
3: Solve the linear system given by equation (9) to get the matrix Q and therewith the quadratic coefficients of the central polynomials.
4: Compute the public key as $\mathcal{P} = \mathcal{F} \circ S$.

5.1 Message Recovery Attacks

Let $M_P = (B|C)$ with B an $(o \times D)$ matrix. After we claimed that fixing B does not give to an attacker advantage in the sense of key recovery attacks, we have to clarify how $B \in \mathbb{F}_2^{o \times D}$ can be chosen without decreasing security against message recovery attacks. Obviously $B = 0$ is a bad choice, as this would imply $C = 0$. We also have to assure that B has full rank, as otherwise C would also not have full rank. In general our goal is that solving our structured system using Gröbner bases is as difficult as solving a random instance over $\mathbb{F}_q$.

We now first introduce our choice of B. Afterwards we explain, why message recovery remains hard.

5.2 Choice of B

The first $(o \times o)$ block in B can be chosen to be the identity matrix $\mathrm{I}_{o \times o}$ as every attacker is able to reach this situation by Gaussian Elimination. Furthermore this ensures B to have full rank. The remaining part B_1 of B has to be chosen in such a way that there are no systematic dependencies between the elements of B, *i.e.* every m columns with $m \geq o$ have a big chance to have full rank. Otherwise we could produce large zero-blocks which would decrease the complexity of Gröbner bases algorithms.

We suggest to choose every element of the matrix B_1 uniformly at random from $\mathbb{F}_2$. Note that for such a B_1 the rank property above is fulfilled with overwhelming probability. Note that B is no longer part of the public key. Once B is constructed, it is fixed, and thus it is a part of the key generation algorithm.

5.3 Ordering of Monomials

In contrast to the method described in [13, 14], we need to choose a special monomial ordering for our construction. In order to understand why this monomial ordering is constructed, let us recall how direct (Gröbner) attacks on multivariate signature schemes work. In the message recovery attack, the attacker is facing the problem of solving the public UOV system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ directly. This system is defined over $\mathbb{F}_q$ and has o equations in $n = o + v$ variables. Such a system has on average $q^{n-o} = q^v$ solutions. Considering the values of v usually used (*e.g.* $v = 52$), such a system has a huge amount of solutions (for $q = 2^8$ and $v = 52$ it

is 2^{416}). Gröbner bases methods have a great difficulty in solving such a system, since they have to describe a huge variety. Since the attacker is interested in only *one* solution for the signature forgery, recovering all solutions is unnecessary. By fixing values of any v variables in the public system, an attacker obtains a quadratic system in o variables and o equations. On average such a system has a unique solution. Solving this new system with Gröbner bases methods is much easier.

Going back to the matrix M_P we see that C (as defined in Section 3) contains coefficients of monomials $x_i x_j$ with $i, j \in \{v+1, \ldots, n\}$, since there we used the graded lexicographic ordering of monomials. Now if the attacker fixes values for the variables $x_{v+1}, \ldots, x_n$, the monomials represented by C will become constants. Therewith, the resulting quadratic system will have only quadratic terms over $\mathbb{F}_2$ coming from the matrix B. Clearly, Gröbner bases computations will be much easier then, since the attacker does not have to deal with $\mathbb{F}_{2^8}$ arithmetics that much. Thus we have to ensure that an attacker is not able to remove too many monomials with coefficients in $\mathbb{F}_{2^8}$ by assigning v variables to some values.

Note that we do not consider monomials of the form x_i^2. If such monomials remain after fixing v variables they do not force us to calculate in $\mathbb{F}_{2^8}$ as they are linear due to the Frobenius homomorphism. Note that for UOV schemes over fields with odd characteristic, it makes sense to consider such monomials.

Denote by $\overline{C}$ the set of monomials whose coefficients are contained in the matrix C. We can represent this set as a graph $G(V, E)$ with $V := \{x_1, \ldots, x_n\}$ being the vertices and $E := \{e(x_i, x_j) \,|\, x_i x_j \in \overline{C}\}$ being the edges. By construction we have $|E| = \frac{o(o+1)}{2}$. In the following our goal is to construct the graph G in such a way that the induced monomial ordering precludes an attacker from removing too many $\mathbb{F}_{2^8}$-terms (independent of the choice of variables he fixes). Note also that by "monomial ordering" we do not mean a monomial well-ordering as in the theory of Gröbner bases, but just some ordering of monomials w.r.t. which the columns of the coefficient matrix M_P are ordered. For the following we need two definitions.

Definition 2. *Let $G(V, E)$ be a graph. A subset $V' \subseteq V$ is called a* k-independent set*, if $|V'| = k$ and $\{e(v_i, v_j) : v_i, v_j \in V'\} \cap E = \emptyset$.*

Definition 3. *For a graph $G(V, E)$ the set $V' \subseteq V$ is called a* k-clique*, if $|V'| = k$ and all the vertices $v_i \in V'$ are pairwise connected,* i.e. *$\{e(v_i, v_j) : v_i, v_j \in V'\} \subseteq E$.*

We observe the following. If G contains an o-independent set, an attacker is able to fix v variables in such a way that all the monomials in $\overline{C}$ become constants.

So our task is to choose the edges of G in such a way, that G does not contain a k-independent set (for minimal k). For fixed k, the problem of finding a graph without k-independent set and minimal number of edges is solved by the complementary Turán graph [17].

So we start with $k = 1$ and construct the complementary Turán graph $\mathrm{CT}(n, 1)$. We then increase k until the number of edges in $\mathrm{CT}(n, k)$ will be

less or equal to $\frac{o\cdot(o+1)}{2}$. If the number of edges in $\mathrm{CT}(n,k)$ is less than $\frac{o\cdot(o+1)}{2}$, we add arbitrarily edges until we reach the number of monomials in $\overline{C}$. By doing so we get a graph G with $\mathrm{CT}(n,k) \leq G$.

Example 1. In our case ($o = 26, v = 52$) we find a solution for $k = 8$ and thus it is assured that at least 30 monomials over $\mathbb{F}_{2^8}$ remain after fixing v variables. The best attack on this parameter set is called HybridF$_5$ [2] and uses fixing v and then guessing two variables before applying Faugères F_5 algorithm [7] to compute a Gröbner basis. But even if we fix/guess $v + 2$ variables, there will remain at least 24 monomials over $\mathbb{F}_{2^8}$. So an attacker can not hope to transfer the system into a smaller field. More details to these experiments can be found in the full version of this paper [15].

Once we have constructed our graph G as above, it defines which monomials are in the set $\overline{C}$. Therefore, we can now define an induced ordering on quadratic monomials, such that monomials from $\overline{C}$ are smaller than those that are not from this set. For the monomials not being in $\overline{C}$ we define real squares (*i.e.* x_i^2 for $i = 1, \ldots, n$) to be bigger than other monomials. Once we defined an ordering of monomials, it is fixed and is a system parameter.

Let us investigate the effect of the new ordering on the construction of matrix A_{UOV}. In Section 3 the columns of the matrix A_{UOV} corresponded to the first D monomials w.r.t. graded lexicographic ordering. Now we have to choose the columns of the transformation matrix in such a way that its columns correspond to the first D monomials in the monomial ordering defined above. With respect to the graph G, if the i-th edge of the complementary graph $\overline{G}$ (which is actually a subgraph of the Turán graph $\mathrm{T}(n,k)$) connects the vertices v_{i_1} and v_{i_2}, we have

$$\widetilde{A_{UOV}} = \begin{pmatrix} \alpha_{11}^{11} & \alpha_{22}^{12} & \ldots & \alpha_{nn}^{11} & \widetilde{\alpha}_1^{11} & \widetilde{\alpha}_2^{11} & \ldots & \widetilde{\alpha}_{D-n}^{11} \\ \alpha_{11}^{12} & \alpha_{22}^{12} & \ldots & \alpha_{nn}^{12} & \widetilde{\alpha}_1^{12} & \widetilde{\alpha}_2^{12} & \ldots & \widetilde{\alpha}_{D-n}^{12} \\ \vdots & & & & & & & \vdots \\ \alpha_{11}^{vn} & \alpha_{22}^{vn} & \ldots & \alpha_{nn}^{vn} & \widetilde{\alpha}_1^{vn} & \widetilde{\alpha}_2^{vn} & \ldots & \widetilde{\alpha}_{D-n}^{vn} \end{pmatrix}. \tag{14}$$

Here, the coefficients α_{ii}^{rs} are given by equation (7) and the $\widetilde{\alpha}_i^{rs}$ are given by

$$\widetilde{\alpha}_i^{rs} = s_{rv_{i1}} \cdot s_{sv_{i2}} + s_{rv_{i2}} \cdot s_{sv_{i1}}. \tag{15}$$

With this notation we get (as in Section 3)

$$B = Q \cdot \widetilde{A_{UOV}}. \tag{16}$$

In the case of $\widetilde{A_{\mathrm{UOV}}}$ being invertible we can use equation (16) to compute the matrix Q and therewith the non zero coefficients of the central map $\mathcal{F}$.

In Algorithm 3 the matrix B chosen as shown in Subsection 5.2 with a fixed matrix $B_1 \in_R \mathbb{F}_2^{o\times(D-o)}$.

5.4 Efficiency of the Verification Process

During the verification process one has to evaluate for each public polynomial the equations

$$P_i(\mathbf{z}) = (z_1, \ldots, z_n) \cdot P_i \cdot (z_1, \ldots, z_n)^T, 1 \leq i \leq o,$$

Algorithm 3. Key Generation for 0/1 UOV

1: Choose randomly a linear map S (represented by an $n \times n$-matrix S). If S is not invertible, choose again.
2: Compute for S the corresponding transformation matrix $\widetilde{A_{\mathrm{UOV}}}$ (using equations (7), (15) and ((14)). If $\widetilde{A_{\mathrm{UOV}}}$ is not invertible, go back to step 1.
3: Solve the linear system given by equation (16) to get the matrix Q and therewith the quadratic coefficients of the central polynomials.
4: Compute the public key as $\mathcal{P} = \mathcal{F} \circ S$.

with $\mathbf{z} = (z_1, \ldots, z_n)$ being the signature of the message and P_i being the (upper triangular) matrix representing the i-th public polynomial.

To evaluate this equation for a randomly chosen P_i one needs $\frac{n\cdot(n+1)}{2} + n$ multiplications in $\mathbb{F}_{2^8}$ for each of the o polynomials, or $o \cdot \frac{n\cdot(n+3)}{2}$ $\mathbb{F}_{2^8}$-multiplications and the same number of additions for the whole key.

For our reduced version we can do better. We first compute $\mathbf{z} \cdot P_i$ $(i = 1, \ldots, o)$. For this we divide each of the matrices P_i into two parts $P_i^{(1)}$ and $P_i^{(2)}$, which we cover by for loops. For an element $a \in P_i^{(1)}$ we test if $a = 0$ or $a = 1$. In the first case, we don't have to do anything, if $a = 1$ we have to carry out one addition. Only for the elements from P_2 we have to perform one multiplication.

By doing so, we can reduce the number of multiplications needed during the verification process to $\frac{o\cdot(o+1)}{2} + n$. For the parameters $(o, v) = (26, 52)$, we get therefore a reduction of 85 %.

However, since we have to perform a number of other operations, for practical implementations we don't get such a hugh reduction in time.

5.5 Security of 0/1 UOV

Since we do not have MDS matrices over $\mathbb{F}_2$, we can not use Proposition 2 to prove the security of our scheme. Therefore we checked the security of our schemes against known attacks, including

1. Direct attacks
2. Rank attacks
3. UOV-Reconciliation attack
4. UOV attack

and found that these attacks cannot use the special structure of our public keys. The results of our experiments can be found in the full version of this paper [15].

5.6 Parameters and Implementation

In this section, we give our choice of parameters and show how they transfer to a practical C++ implementation. More concrete, based on our security considerations, we propose for our scheme the same parameters as for the standard

UOV scheme, namely field size $q = 2^8$, $(o, v) = (26, 52)$. Additionally, Table 1 gives one more conservative parameter set, namely $(q, o, v) = (2^8, 28, 56)$.
We implemented our scheme and the standard UOV in C++.

Key Generation: For the key generation we follow closely Algorithm 3. The linear system in step 3 of the algorithm is solved by inverting the matrix $\widetilde{A_{UOV}}$ and then computing the matrix product $B \cdot \widetilde{A_{UOV}}$. For both we use the M4RIE library for efficient linear algebra over finite fields and Travolta tables. By doing so, we can compute a key pair for 0/1 UOV(2^8,26,52) in roughly 27 sec on an Intel Dual Core 2 with 2.53 MHz.

Signature Generation: The signature generation process works as for the standard UOV Scheme. The running time of the signature generation process is about 0.69 ms.

Signature Verification: The signature verification process works as desribed in Subsection 5.4. For each parameter set listed in Table 2 we carried out 1,000,000 verification processes on the Intel machine as well as on an AMD Athlon XP 2400+ with 2.00 GHz. Table 2 shows the results.

Table 1. Proposed parameters for UOV schemes

Scheme(q, o, v)	system parameter (kB)	public key size (kB)	private key size (kB)	reduction of public key size (%)
UOV(2^8,26,52)	-	78.2	75.3	-
Compr.UOV(2^8,26,52)	69.3	8.9	75.3	88.6
0/1 UOV(2^8,26,52)	8.7	8.9	75.3	88.6
UOV(2^8,28,56)	-	97.6	93.4	-
Compr.UOV(2^8,28,56)	86.5	11.1	93.4	88.6
0/1 UOV(2^8,28,56)	10.8	11.1	93.4	88.6

Table 2. Running time of the verification process

	Intel Dual Core 2 2.53 GHz			AMD Athlon XP 2400+ 2.00 GHz		
(o, v)	UOV	0/1 UOV	reduction factor	UOV	0/1 UOV	reduction factor
(26,52)	0.49 ms	0.21 ms	57 %	0.68 ms	0.29 ms	57 %
(28,56)	0.54 ms	0.22 ms	58 %	0.74 ms	0.32 ms	58 %
(32,64)	0.75 ms	0.30 ms	59 %	1.03 ms	0.43 ms	58 %

6 Conclusion

In this paper, we have shown that $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic public key schemes can benefit from much smaller public key sizes (cf. Table 1) without any degeneration of security. The overall idea requires some flexibility in the private key.

To our knowledge, only the two $\mathcal{MQ}$-schemes UOV and Rainbow have these. UOV was covered in this article. Rainbow has a more difficult internal structure, so we have to leave a concrete application of our improvement to Rainbow as an open question, which we plan to address.

The security arguments made use of the idea of equivalent keys. Hereby, each public key can be assigned many private keys. We have turned this idea around by considering transformations of the public key $\mathcal{P}$ instead and showed that an attacker does not gain from this specific structure.

As we can enforce a specific form on the public key $\mathcal{P}$, we can also use it to speed up public key operations, namely verification of signatures. As we see in Table 2, this reduces the overall time by about 59% or a markable factor of 2.4.

As the construction is very general, it can be used on other platforms (*e.g.* GPU, FPGA) as well. We actually expect similar gains in area reduction or speed there, too.

From a theoretical perspective, forcing a specific structure on the central polynomials $\mathcal{F}$ or the public polynomials $\mathcal{P}$ are equivalent: We can do either. Hence, for specific application domains it might be useful to find a certain trade-off. For example, we could reduce the computational workload on a server based on the maximal available memory on a smart card.

Acknowledgements. We thank Ishtiaq Shah for doing the implementation of our scheme. Furthermore we want to thank our financial supporters. The first author is supported by the Horst Görtz Foundation (HGS) within the project where the third author is the principal investigator. The third author is supported by the DFG grant BU 630/22-1. The second author was supported by the German Science Foundation (DFG) through an Emmy Noether grant where the forth author is principal investigator.

References

[1] Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer, Heidelberg (2009)

[2] Bettale, L., Faugére, J.-C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology 3, 177–197 (2009)

[3] Bogdanov, A., Eisenbarth, T., Rupp, A., Wolf, C.: Time-area optimized public-key engines: $\mathcal{MQ}$-cryptosystems as replacement for elliptic curves? In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 45–61. Springer, Heidelberg (2008)

[4] Chen, A.I.-T., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M., Yang, B.-Y.: Practical-sized instances of multivariate pKCs: Rainbow, TTS, and ℓIC-derivatives. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 95–108. Springer, Heidelberg (2008)

[5] Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S., Yang, B.-Y.: SSE implementation of multivariate pKCs on modern x86 cPUs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)

[6] Ding, J., Gower, J.E., Schmidt, D.: Multivariate Public Key Cryptography. Cambridge University Press, Cambridge (2006)
[7] Faugére, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: ISSAC 2002, pp. 75–83. ACM Press, New York (2002)
[8] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
[9] Hu, Y., Wang, L., Chou, C., Lai, F.: Similar keys of multivariate quadratic public key cryptosystems. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 211–222. Springer, Heidelberg (2005)
[10] Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
[11] Kipnis, A., Shamir, A.: Cryptanalysis of the oil & vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
[12] Patarin, J.: The oil and vinegar signature scheme. Presented at the Dagstuhl Workshop on Cryptography, transparencies (September 1997)
[13] Petzoldt, A., Bulygin, S., Buchmann, J.: A multivariate signature scheme with a partially cyclic public key. In: SCC 2010, pp. 229–235 (2010)
[14] Petzoldt, A., Bulygin, S., Buchmann, J.: Linear recurring sequences for the UOV key generation. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 335–350. Springer, Heidelberg (2011)
[15] Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems (full version), http://eprint.iacr.org/2011/294
[16] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997)
[17] Turán, P.: On an extremal problem in Graph Theory. Matematiko Fizicki Lapok 48, 436–452 (1941)
[18] Wolf, C., Preneel, B.: Taxonomy of public key schemes based on the problem of multivariate quadratic equations, http://eprint.iacr.org/2005/077
[19] Wolf, C., Preneel, B.: Superfluous keys in Multivariate Quadratic asymmetric systems. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 275–287. Springer, Heidelberg (2005)
[20] Wolf, C., Preneel, B.: Equivalent keys in multivariate quadratic public key systems. Journal of Mathematical Cryptology (to appear, 2011)
[21] Yang, B.-Y., Chen, J.-M., Chen, Y.-H.: TTS: High-speed signatures on a low-cost smart card. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 371–385. Springer, Heidelberg (2004)