Fast Verification for Improved Versions of the UOV and Rainbow Signature Schemes

Albrecht Petzoldt[1], Stanislav Bulygin, and Johannes Buchmann[1,2]

[1] Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
{apetzoldt,buchmann}@cdc.informatik.tu-darmstadt.de

[2] Center for Advanced Security Research Darmstadt - CASED
Mornewegstraße 32, 64293 Darmstadt, Germany
johannes.buchmann@cased.de

Abstract. Multivariate cryptography is one of the main candidates to guarantee the security of communication in the post-quantum era. While multivariate signature schemes are fast and require only modest computational resources, the key sizes of such schemes are quite large. In [14] Petzoldt et al. proposed a way to reduce the public key size of certain multivariate signature schemes like UOV and Rainbow by a large factor. In this paper we show that by using this idea it is possible to speed up the verification process of these schemes, too. For example, we are able to speed up the verification process of UOV by a factor of 5.

Keywords: Multivariate Cryptography, UOV Signature Scheme, Rainbow Signature Scheme, Key Size Reduction, Fast Verification

1 Introduction

When quantum computers arrive, classical public-key cryptosystems like RSA and ECC will be broken [1]. The reason for this is Shor's algorithm [18] which solves number theoretic problems like integer factorization and discrete logarithms in polynomial time on a quantum computer. So, to guarantee the security of communication in the post-quantum era, we need alternatives to those classical schemes. Besides lattice-, code-, and hash-based cryptosystems multivariate cryptography seems to be a candidate for this.
Additionally to its (believed) resistance against quantum computer attacks, multivariate cryptosystems are very fast, especially for signatures [2,3]. Furthermore they require only modest computational resources, which makes them appropriate for the use on low-cost devices like smartcards and RFID chips. However, multivariate schemes are not widely used yet, mainly because of the large size of their public and private keys.
In [14], [16] and [17] Petzoldt et al. showed different possibilities to decrease the public key size of the Unbalanced Oil and Vinegar (UOV) and Rainbow signature schemes. The key idea is it to insert a highly structured matrix into

the coefficient matrix of the public key. Therefore, the coefficient matrix of the public key has the form $M_P = (B|C)$, where B is a matrix of a very special form (e.g. partially circulant or generated by an LFSR) and C is a matrix without visible structure. By doing so, they were able to decrease the public key size of UOV by 86 %, namely from 99.9 kB to 13.4 kB.
In this paper we show that this idea can not only be used to decrease the size of the public key, but also to speed up the verification process. We use the rich structure of the matrix B to reduce the number of field multiplications needed during the verification process by a large factor (for cyclicUOV this factor is about 80 %). We derive our results both theoretically and show them using a C implementation of the schemes.
The structure of this paper is as follows: In Section 2 we give a short overview on multivariate signature schemes and describe the UOV and Rainbow signature schemes. Section 3 reviews the approach of [14] and [16] to create UOV and Rainbow schemes with structured public keys. In Section 4 we demonstrate how we can use this special structure to speed up the verification process of the schemes. In Subsection 4.2 we look hereby on partially cyclic UOV schemes, whereas Subsection 4.3 deals with cyclic versions of Rainbow. Section 5 presents the results of our experiments and Section 6 concludes the paper.

2 Multivariate Public Key Cryptography

The basic idea behind multivariate cryptography is to choose a system $\mathcal{F}$ of m quadratic polynomials in n variables which can be easily inverted (central map). After that one chooses two affine invertible maps $\mathcal{S}$ and $\mathcal{T}$ to hide the structure of the central map. The public key of the cryptosystem is the composed quadratic map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ which is difficult to invert. The private key consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert $\mathcal{P}$.
Due to this construction, the security of multivariate cryptography is based on two mathematical problems:

Problem MQ: Solve the system $p^{(1)} = \cdots = p^{(m)} = 0$, where each $p^{(i)}$ is a quadratic polynomial in the n variables $x_1, \ldots, x_n$ with coefficients and variables in $GF(q)$.

The MQ-problem is proven to be NP-hard even for quadratic polynomials over $GF(2)$ [8].

Problem EIP (Extended Isomorphism of Polynomials): Given a class of central maps $\mathcal{C}$ and a map $\mathcal{P}$ expressible as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, where $\mathcal{S}$ and $\mathcal{T}$ are affine maps and $\mathcal{F} \in \mathcal{C}$, find a decomposition of $\mathcal{P}$ of the form $\mathcal{P} = \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}'$, with affine maps $\mathcal{S}'$ and $\mathcal{T}'$ and $\mathcal{F}' \in \mathcal{C}$.

In this paper we concentrate on the case of multivariate signature schemes.

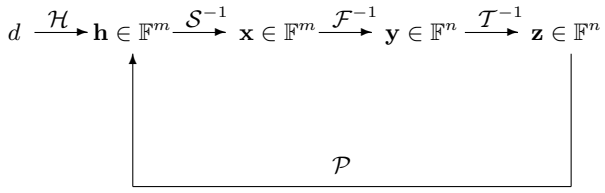

Fig. 1: Signature generation and verification

The standard process for signature generation and verification works as shown in Figure 1.

Signature Generation To sign a document d, we use a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}^m$ to compute the value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}^m$. Then we compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document is $\mathbf{z} \in \mathbb{F}^n$. Here, $\mathcal{F}^{-1}(\mathbf{x})$ means finding one (of the possibly many) pre-image of $\mathbf{x}$ under the central map $\mathcal{F}$.

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

There are several ways to build the central map $\mathcal{F}$ of multivariate schemes. In this paper we concentrate on the so called SingleField constructions. In contrast to BigField schemes like Matsumoto-Imai [11] and MiddleField schemes like ℓiC [6], here all the computations are done in one (relatively small) field. In the following two subsections we describe two well known examples of these schemes in detail.

2.1 The Unbalanced Oil and Vinegar (UOV) Signature Scheme

One way to create an easily invertible multivariate quadratic system is the principle of Oil and Vinegar, which was first proposed by J. Patarin in [13].
Let $\mathbb{F}$ be a finite field. Let o and v be two integers and set $n = o + v$. We set $V = \{1, \ldots, v\}$ and $O = \{v+1, \ldots, n\}$. We call $x_1, \ldots, x_v$ the Vinegar variables and $x_{v+1}, \ldots, x_n$ Oil variables. We define o quadratic polynomials $f^{(k)}(\mathbf{x}) = f^{(k)}(x_1, \ldots, x_n)$ by

$$f^{(k)}(\mathbf{x}) = \sum_{i \in V,\ j \in O} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V,\ i \le j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)} \ (1 \le k \le o). \tag{1}$$

Note that Oil and Vinegar variables are not fully mixed, just like oil and vinegar in a salad dressing.

The map $\mathcal{F} = (f^{(1)}(\mathbf{x}), \ldots, f^{(o)}(\mathbf{x}))$ can be easily inverted. First, we choose the values of the v Vinegar variables $x_1, \ldots, x_v$ at random. Therefore we get a system of o linear equations in the o variables $x_{v+1}, \ldots, x_n$ which can be solved e.g. by Gaussian Elimination. If the system does not have a solution, one has to choose other values of $x_1, \ldots, x_v$ and try again.
The public key of the scheme is given as $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, where $\mathcal{T}$ is an affine map from $\mathbb{F}^n$ to itself. The private key consists of the two maps $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert the public key.

Remark: In opposite to other multivariate schemes the second affine map $\mathcal{S}$ is not needed for the security of UOV. So it can be omitted.

In his original paper [13] Patarin suggested to choose $o = v$ (Balanced Oil and Vinegar (OV)). After this scheme was broken by Kipnis and Shamir in [10], it was recommended in [9] to choose $v > o$ (Unbalanced Oil and Vinegar (UOV)). The UOV signature scheme over GF(256) is commonly believed to be secure for $o \geq 28$ equations [19] and $v = 2 \cdot o$ Vinegar variables. For UOV schemes over GF(31) we set $(o, v) = (33, 66)$.

2.2 The Rainbow Signature Scheme

In [4] J. Ding and D. Schmidt proposed a signature scheme called Rainbow, which is based on the idea of (Unbalanced) Oil and Vinegar [9].

Let $\mathbb{F}$ be a finite field and V be the set $\{1, \ldots, n\}$. Let $v_1, \ldots, v_{u+1}, u \geq 1$ be integers such that $0 < v_1 < v_2 < \cdots < v_u < v_{u+1} = n$ and define the sets of integers $V_i = \{1, \ldots, v_i\}$ for $i = 1, \ldots, u$. We set $o_i = v_{i+1} - v_i$ and $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ $(i = 1, \ldots, u)$. The number of elements in V_i is v_i and we have $|O_i| = o_i$. For $k = v_1 + 1, \ldots, n$ we define multivariate quadratic polynomials in the n variables $x_1, \ldots, x_n$ by

$$f^{(k)}(\mathbf{x}) = \sum_{i \in O_l,\ j \in V_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V_l,\ i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} x_i + \eta^{(k)}, \quad (2)$$

where l is the only integer such that $k \in O_l$. Note that these are Oil and Vinegar polynomials with $x_i,\ i \in V_l$ being the Vinegar variables and $x_j,\ j \in O_l$ being the Oil variables.
The map $\mathcal{F}(\mathbf{x}) = (f^{(v_1+1)}(\mathbf{x}), \ldots, f^{(n)}(\mathbf{x}))$ can be inverted as follows. First, we choose $x_1, \ldots, x_{v_1}$ at random. Hence we get a system of o_1 linear equations (given by the polynomials $f^{(k)}$ $(k \in O_1)$) in the o_1 unknowns $x_{v_1+1}, \ldots, x_{v_2}$, which can be solved by Gaussian Elimination. The so computed values of x_i $(i \in O_1)$ are plugged into the polynomials $f^{(k)}(\mathbf{x})$ $(k > v_2)$ and a system of o_2 linear equations (given by the polynomials $f^{(k)}$ $(k \in O_2)$) in the o_2 unknowns x_i $(i \in O_2)$ is obtained. By repeating this process we can get values for all the variables x_i $(i = 1, \ldots, n)$ [3].

[3] It may happen, that one of the linear systems does not have a solution. If so, one has to choose other values of $x_1, \ldots x_{v_1}$ and try again.

The public key of the scheme is given as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ with two invertible affine maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. The private key consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert te public key.
In the following, we restrict ourselves to Rainbow schemes with two layers (i.e. $u = 2$). For this, $\mathbb{F} = GF(256)$, $(v_1, o_1, o_2) = (17, 13, 13)$ provides 80-bit security under known attacks [15]. For Rainbow schemes over GF(31), we choose $(v_1, o_1, o_2) = (14, 19, 14)$.

3 Improved versions of UOV and Rainbow

In [14] and [16] Petzoldt et al. presented an approach to create UOV- and Rainbow-based schemes with structured public keys, by which they could reduce the public key size of these schemes by up to 83 %. Due to lack of space we give here only a very brief description and refer to [14] and [16] for the details.

The main idea of the approach is to insert a structured matrix B into the Macauley matrix M_P of the public key. In our case the matrix B is chosen partially circulant, i.e. its rows are given by

$$B[i] = \mathcal{R}^{i-1}(\mathbf{b}) \ (i = 1, \ldots, m), \tag{3}$$

where $\mathbf{b}$ is a randomly chosen vector and $\mathcal{R}^i$ denotes the cyclic right shift by i positions.
To insert this matrix B into M_P, the authors used the relation $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ between a UOV public and private key, which translates into the matrix equation

$$M_P = M_F \cdot A \tag{4}$$

with a transformation matrix A whose elements are given as quadratic functions in the coefficients of the affine map $\mathcal{T}$. If this matrix is invertible, one can compute the matrix M_F in such a way that M_P has the form $M_P = (B|C)$ with a partially circulant matrix B and a matrix C without visible structure. Figure 2 shows this key generation process graphical form.

4 The verification process

The central part of the verification process for multivariate signature schemes is the evaluation of the public polynomials. Normally this is done as follows: For a given (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}^n$ one first computes an $\frac{(n+1)\cdot(n+2)}{2}$ vector mon, which contains the values of all monomials of degree ≤ 2, i.e.

$$\text{mon} = (z_1^2, z_1 z_2, \ldots, z_n^2, z_1, \ldots, z_n, 1). \tag{5}$$

Then we have

$$\mathcal{P}(\mathbf{z}) = \begin{pmatrix} M_P[1] \cdot \text{mon}^T \\ \vdots \\ M_P[m] \cdot \text{mon}^T \end{pmatrix}, \tag{6}$$

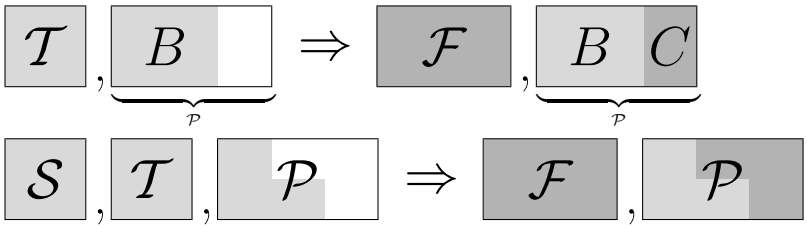

Fig. 2: Alternative key generation for UOV (above) and Rainbow. The light gray parts are chosen by the user, the dark gray parts are computed during the key generation process.

with $M_P[i]$ being the i-th row of the Macauley matrix M_P and $\cdot$ being the standard scalar product.

For schemes with partially cyclic public key, the following strategy seems to be more promising:

4.1 Notations

Let $\mathbf{h} = (h_1, \ldots, h_m)$ be the hash value of the signed message.
The public polynomials can be written as

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(k)} \cdot x_i + p_0^{(k)} \quad (k = 1, \ldots, m). \quad (7)$$

For $k = 1, \ldots, m$ we define upper triangular matrices $MP^{(k)}$ by

$$MP^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} & p_1^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} & p_2^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} & p_3^{(k)} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & p_{nn}^{(k)} & p_n^{(k)} \\ 0 & 0 & \ldots & 0 & 0 & p_0^{(k)} \end{pmatrix}. \quad (8)$$

For a (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n)$ of the message we define the extended signature vector

$$\text{sign} = (z_1, \ldots, z_n, 1). \quad (9)$$

With this notation we can write the verification process in the following form

$$\text{accept the signature } \mathbf{z} \iff \text{sign} \cdot MP^{(k)} \cdot \text{sign}^T = h_k \ \forall k \in \{1, \ldots, m\}. \quad (10)$$

In the following two subsections we consider the question how we can evaluate this equation more efficiently for improved versions of UOV and Rainbow.

4.2 cyclicUOV

In the case of cyclicUOV [14], the matrices $MP^{(k)}$ are of the form shown in Figure 3. We have

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \ \forall i = 1, \ldots, v, \ j = i+1, \ldots, n, \ k = 2, \ldots, o. \tag{11}$$

Therefore we get

$$(\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1,j}^{(k)} \\ MP_{2,j}^{(k)} \\ \vdots \\ MP_{i,j}^{(k)} \end{pmatrix} = (\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1,j-1}^{(k-1)} \\ MP_{2,j-1}^{(k-1)} \\ \vdots \\ MP_{i,j-1}^{(k-1)} \end{pmatrix} \quad \begin{matrix} \forall i = 1, \ldots, v \\ j = i+1, \ldots, n, \\ k = 2, \ldots, o. \end{matrix} \tag{12}$$

The boxes in Figure 3 illustrate this equation. Boxes with continuous lines show the vector $(MP_{1,j-1}^{(k-1)}, \ldots, MP_{i,j-1}^{(k-1)})^T$ on the right hand side of the equation, whereas the boxes with dashed lines represent the vector $(MP_{1,j}^{(k)}, \ldots, MP_{i,j}^{(k)})^T$ on the left hand side. As one can see, the dashed boxes in the matrix $MP^{(k)}$ are exactly the same as the boxes with continuous lines in the matrix $MP^{(k-1)}$ $(k = 2, \ldots, o)$. We can use this fact to speed up the verification process of cyclicUOV by a large factor (see Algorithm 1).

Algorithm 1 works as follows. The first matrix-vector product $\text{sign} \cdot MP^{(1)} \cdot \text{sign}^T$ is computed as for a random polynomial: From step 1 to step 9 we compute the product $\text{sign} \cdot MP^{(1)}$ (the result is written into the vector temp) and step 10 computes the scalar product of temp and sign. Furthermore we compute the vector $a = (a_1, \ldots, a_{n-1})$ which can be used for the computation of $\text{sign} \cdot MP^{(2)}$. In the loop (step 11 to step 24 of the algorithm) we compute the matrix vector products $\text{sign} \cdot MP^{(k)} \cdot \text{sign}^T$ $(k = 2, \ldots, o)$. Step 12 to step 22 computes the vector $\text{temp} = \text{sign} \cdot MP^{(k)}$. We begin with temp_{n+1} and go back to temp_1. In the computation of temp_i $(i = 2, \ldots, n)$ we use the values a_i computed before, since, due to the cyclic structure of the public key, they appear in several of the products $\text{sign} \cdot MP^{(k)}$ (see equation (12)). Furthermore (step 18 and 21) we update the values of the a_i $(i = 1, \ldots, n-1)$ for the use in the next iteration of the loop. Step 23 computes the scalar product of temp and sign. The last three steps (step 25 to 27) use the values h'_l $(l = 1, \ldots o)$ computed in step 10 and 23 to verify the authenticity of the signature.

$$
MP^{(1)} = \begin{pmatrix}
\boxed{s_1} & \boxed{s_2} & \boxed{s_3} & \cdots & \boxed{s_{v-1}} & \boxed{s_v} & \boxed{s_{v+1}} & \cdots & \boxed{s_{n-1}} & \boxed{s_n} & \star \\
0 & \boxed{s_{n+1}} & \boxed{s_{n+2}} & \cdots & \boxed{s_{n+v-2}} & \boxed{s_{n+v-1}} & \boxed{s_{n+v}} & \cdots & \boxed{s_{2n-2}} & \boxed{s_{2n-1}} & \star \\
0 & 0 & \boxed{s_{2n}} & \cdots & \boxed{s_{2n+v-4}} & \boxed{s_{2n+v-3}} & \boxed{s_{2n+v-2}} & \cdots & \boxed{s_{3n-4}} & \boxed{s_{3n-3}} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \cdots & 0 & \boxed{s_{D-2o-1}} & \boxed{s_{D-2o}} & \boxed{s_{D-2o+1}} & \cdots & \boxed{s_{D-o-2}} & \boxed{s_{D-o-1}} & \star \\
0 & & \cdots & & 0 & \boxed{s_{D-o}} & \boxed{s_{D-o+1}} & \cdots & \boxed{s_{D-1}} & s_D & \star \\
0 & & \cdots & & & 0 & \star & \cdots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \cdots & & \cdots & & \cdots & & 0 & \star & \star \\
0 & & \cdots & & \cdots & & \cdots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(2)} = \begin{pmatrix}
\boxed{s_D} & \boxed{s_1} & \boxed{s_2} & \cdots & \boxed{s_{v-2}} & \boxed{s_{v-1}} & \boxed{s_v} & \cdots & \boxed{s_{n-2}} & \boxed{s_{n-1}} & \star \\
0 & \boxed{s_n} & \boxed{s_{n+1}} & \cdots & \boxed{s_{n+v-3}} & \boxed{s_{n+v-2}} & \boxed{s_{n+v-1}} & \cdots & \boxed{s_{2n-3}} & \boxed{s_{2n-2}} & \star \\
0 & 0 & \boxed{s_{2n-1}} & \cdots & \boxed{s_{2n+v-5}} & \boxed{s_{2n+v-4}} & \boxed{s_{2n+v-3}} & \cdots & \boxed{s_{3n-5}} & \boxed{s_{3n-4}} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \cdots & 0 & \boxed{s_{D-2o-2}} & \boxed{s_{D-2o-1}} & \boxed{s_{D-2o}} & \cdots & \boxed{s_{D-o-3}} & \boxed{s_{D-o-2}} & \star \\
0 & & \cdots & & 0 & \boxed{s_{D-o-1}} & \boxed{s_{D-o}} & \cdots & \boxed{s_{D-2}} & \boxed{s_{D-1}} & \star \\
0 & & \cdots & & & 0 & \star & \cdots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \cdots & & \cdots & & \cdots & & 0 & \star & \star \\
0 & & \cdots & & \cdots & & \cdots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(o-1)} = \begin{pmatrix}
\vdots & & & & & & & & & & \\
\boxed{s_{D-o+3}} & \boxed{s_{D-o+4}} & \boxed{s_{D-o+5}} & \cdots & \boxed{s_{o+1}} & \boxed{s_{o+2}} & \boxed{s_{o+3}} & \cdots & \boxed{s_{v+1}} & \boxed{s_{v+2}} & \star \\
0 & \boxed{s_{v+3}} & \boxed{s_{v+4}} & \cdots & \boxed{s_{n+o}} & \boxed{s_{n+o+1}} & \boxed{s_{n+o+2}} & \cdots & \boxed{s_{n+v}} & \boxed{s_{n+v+1}} & \star \\
0 & 0 & \boxed{s_{n+v+2}} & \cdots & \boxed{s_{2n+o-2}} & \boxed{s_{2n+o-1}} & \boxed{s_{2n+o}} & \cdots & \boxed{s_{2n+v-2}} & \boxed{s_{2n+v-1}} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \cdots & 0 & \boxed{s_{D-3o+1}} & \boxed{s_{D-3o+2}} & \boxed{s_{D-3o+3}} & \cdots & \boxed{s_{D-2o}} & \boxed{s_{D-2o+1}} & \star \\
0 & & \cdots & & 0 & \boxed{s_{D-2o+2}} & \boxed{s_{D-2o+3}} & \cdots & \boxed{s_{D-o+1}} & \boxed{s_{D-o+2}} & \star \\
0 & & \cdots & & & 0 & \star & \cdots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \cdots & & \cdots & & \cdots & & 0 & \star & \star \\
0 & & \cdots & & \cdots & & \cdots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(o)} = \begin{pmatrix}
\boxed{s_{D-o+2}} & \boxed{s_{D-o+3}} & \boxed{s_{D-o+4}} & \cdots & \boxed{s_o} & \boxed{s_{o+1}} & \boxed{s_{o+2}} & \cdots & \boxed{s_v} & \boxed{s_{v+1}} & \star \\
0 & \boxed{s_{v+2}} & \boxed{s_{v+3}} & \cdots & \boxed{s_{n+o-1}} & \boxed{s_{n+o}} & \boxed{s_{n+o+1}} & \cdots & \boxed{s_{n+v-1}} & \boxed{s_{n+v}} & \star \\
0 & 0 & \boxed{s_{n+v+1}} & \cdots & \boxed{s_{2n+o-3}} & \boxed{s_{2n+o-2}} & \boxed{s_{2n+o-1}} & \cdots & \boxed{s_{2n+v-3}} & \boxed{s_{2n+v-2}} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \cdots & 0 & \boxed{s_{D-3o}} & \boxed{s_{D-3o+1}} & \boxed{s_{D-3o+2}} & \cdots & \boxed{s_{D-2o-1}} & \boxed{s_{D-2o}} & \star \\
0 & & \cdots & & 0 & \boxed{s_{D-2o+1}} & \boxed{s_{D-2o+2}} & \cdots & \boxed{s_{D-o}} & \boxed{s_{D-o+1}} & \star \\
0 & & \cdots & & & 0 & \star & \cdots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \cdots & & \cdots & & \cdots & & 0 & \star & \star \\
0 & & \cdots & & \cdots & & \cdots & & & 0 & \star
\end{pmatrix}
$$

Fig. 3: Matrices $MP^{(i)}$ for cyclicUOV

Algorithm 1 Verification process for cyclicUOV

for $i = 1$ to $n-1$ **do** ▷ first polynomial
  $a_i \leftarrow \sum_{j=1}^{\min(i,v)} MP_{ji}^{(1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v+1$ to $n-1$ **do**
  $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
end for
$\text{temp}_n \leftarrow \sum_{j=1}^{n} MP_{ji}^{(1)} \cdot \text{sign}_j$
$\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(1)} \cdot \text{sign}_\text{j}$
$h'_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = 2$ to o **do** ▷ polynomials $2, \ldots, o$
  $\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(l)} \cdot \text{sign}_j$
  for $i = n$ to $v+1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(l)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(l)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for
if $h_l = h'_l \ \forall l \in \{1, \ldots, o\}$ **then return** "ACCEPT" ▷ TEST
else return "REJECT"
end if

Computational effort Evaluating the system $\mathcal{P}$ in the standard way, one needs

- $\frac{n\cdot(n+1)}{2}$ field multiplications to compute the vector mon (c.f. equation (5))
- and $o \cdot \frac{(n+1)\cdot(n+2)-2}{2}$ field multiplications to compute the scalar products of equation (6).

Altogether, we need

$$\frac{n+1}{2} \cdot (n \cdot (o+1) + 2 \cdot o) - o \tag{13}$$

field multiplications. Algorithm 1 needs

- in step 2 $\frac{v\cdot(v+1)}{2} + (o-1) \cdot v$ field multiplications,
- in step 6 $\frac{(o-1)\cdot o}{2}$ field multiplications,
- in step 8 n field multiplications,
- in step 9 $n+1$ field multiplications,
- and in step 10 again $n+1$ field multiplications.

Therefore, to compute the value of h_1', the algorithm needs $\frac{(n+1)\cdot(n+4)}{2}$
In the loop (step 11 to 24) Algorithm 1 needs

- in step 12 $n+1$ field multiplications,
- in step 15 $\frac{o\cdot(o+1)}{2}$ field multiplications,
- in step 18 $v-1$ field multiplications,
- in step 21 1 field multiplication,
- and in step 23 $n+1$ field multiplications.

So, for every iteration of the loop the algorithm needs $2 \cdot (n+1) + v + \frac{o\cdot(o+1)}{2}$ field multiplications.
Altogether, we need therefore

$$(o-1) \cdot \left(2 \cdot (n+1) + v + \frac{o \cdot (o+1)}{2}\right) + \frac{(n+1) \cdot (n+4)}{2} \tag{14}$$

field multiplications to evaluate equation (10).

For $\mathbb{F}$ =GF(256), $(o, v) = (28, 56)$ this means a reduction of the number of field multiplications needed during the verification process by 80 % or a factor of 5.0. For a UOV scheme over GF(31), $(o, v) = (33, 66)$, we get a reduction factor of 5.4.

4.3 cyclicRainbow

The verification process for cyclicRainbow is mainly done as for cyclicUOV. However we have to consider the different structure of the polynomials. For cyclicRainbow, the matrices $MP^{(k)}$ look as shown in Figure 4.
So we get for the polynomials $2, \ldots, o_1 + 1$

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \ \forall i = 1, \ldots, v_1, \ j = i+1, \ldots, v_2, \ k = 2, \ldots, o_1 + 1 \tag{15}$$

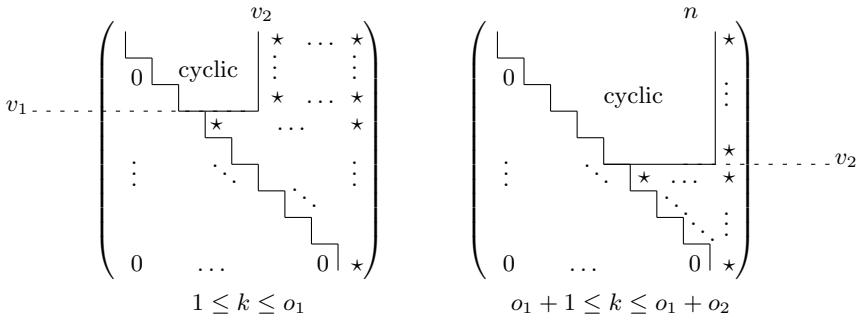

Fig. 4: Matrices $MP^{(k)}$ for cyclicRainbow

or

$$(\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP^{(k)}_{1,j} \\ MP^{(k)}_{2,j} \\ \vdots \\ MP^{(k)}_{i,j} \end{pmatrix} = (\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP^{(k-1)}_{1,j-1} \\ MP^{(k-1)}_{2,j-1} \\ \vdots \\ MP^{(k-1)}_{i,j-1} \end{pmatrix} \quad \begin{array}{c} \forall i = 1, \ldots, v_1, \\ j = i+1, \ldots, v_2, \\ k = 2, \ldots, o_1 + 1. \end{array} \tag{16}$$

For the polynomials $o_1 + 2, \ldots, o_1 + o_2$ we get

$$MP^{(k)}_{ij} = MP^{(k-1)}_{i,j-1} \; \forall i = 1, \ldots, v_2, \; j = i+1, \ldots, n, \; k = o_1+2, \ldots, o_1+o_2 \tag{17}$$

or

$$(\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP^{(k)}_{1,j} \\ MP^{(k)}_{2,j} \\ \vdots \\ MP^{(k)}_{i,j} \end{pmatrix} = (\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP^{(k-1)}_{1,j-1} \\ MP^{(k-1)}_{2,j-1} \\ \vdots \\ MP^{(k-1)}_{i,j-1} \end{pmatrix} \quad \begin{array}{c} \forall i = 1, \ldots, v_2, \\ j = i+1, \ldots, n, \\ k = o_1 + 2, \ldots, o_1 + o_2. \end{array} \tag{18}$$

To cover this fact, we use Algorithm 1 for both groups of polynomials separately (see Algorithm 2 in Appendix A).

Computational cost Our algorithm needs

- $\frac{(n+1)\cdot(n+4)}{2}$ field multiplications to evaluate $p^{(1)}$,
- $o_1 \cdot \left(\frac{(n+1)\cdot(n+4)}{2} - \frac{v_2\cdot(v_2+1)}{2} + \frac{o_1\cdot(o_1+1)}{2} + v_1 \right)$ field multiplications to evaluate the polynomials $p^{(2)} \ldots, p^{(o_1+1)}$ and
- $(o_2 - 1) \cdot \left(2 \cdot (n+1) + v_2 + \frac{o_2\cdot(o_2+1)}{2} \right)$ field multiplications to evaluate the polynomials $p^{(o_1+2)}, \ldots, p^{(o_1+o_2)}$.

For the parameters $(q, v_1, o_1, o_2) = (2^8, 17, 13, 13)$, this means a reduction by 56 % or a factor of 2.3 (with respect to the evaluation with the standard approach, see (13)). For a Rainbow scheme over GF(31), $(v_1, o_1, o_1) = (14, 19, 14)$ the reduction factor is 2.2.

5 Experiments

We checked our theoretical results on a straightforward C implementation of our schemes. Table 1 shows the results. The parameters in this table are chosen for 80 bit security.

Scheme	private key size (kB)	hash length (bit)	signature length (bit)	public key size (kB)	public key red. factor	verification time ms	verification time s. u. f. [1]
UOV$(31, 33, 66)$	102.9	160	528	108.5	-	1.75	-
cyclicUOV$(31, 33, 66)$	102.9	160	528	17.1	6.3	0.34	5.2
UOV$(256, 28, 56)$	95.8	224	672	99.9	-	0.98	-
cyclicUOV$(256, 28, 56)$	95.8	224	672	16.5	6.1	0.20	4.9
Rainbow$(31, 14, 19, 14)$	17.1	160	256	25.3	-	0.44	-
cyclicRainbow$(31, 14, 19, 14)$	17.1	160	256	12.0	2.1	0.21	2.1
Rainbow$(256, 17, 13, 13)$	19.1	208	344	25.1	-	0.26	-
cyclicRainbow$(256, 17, 13, 13)$	19.1	208	344	9.5	2.6	0.13	2.0

[1] speed up factor of the verification time

Table 1: Improved versions of UOV and Rainbow

The differences between the results of our theoretical analysis (see Section 4) and the actual runtime of the verification process is mainly caused by the heavy use of control structures in Algorithms 1 and 2.

6 Conclusion

In this paper we show a way how the structure in the public keys of cyclic versions of UOV and Rainbow can be used to achieve a significant speed up of the verification process. We propose improved algorithms for the verification process of UOV and Rainbow which run up to 5 times faster than the standard verification algorithm. Future research includes:

- *Use of special processor instructions*
  Like in the paper of Chen et al. [3] we plan to use special processor instructions to speed up our implementations.
- *Implementation in hardware*
  We plan to implement our schemes in hardware (e.g. on FPGA and HSM), which should also decrease the verification time.

7 Acknowledgements

We want to thank the anonymous reviewers for their comments which helped to improve the paper. The first author is supported by the Horst Görtz Foundation.

References

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

[2] A. Bogdanov, T. Eisenbarth, A. Rupp, and C. Wolf. Time-area optimized public-key engines: -cryptosystems as replacement for elliptic curves? CHES 2008, LNCS vol. 5154, pp. 45-61. Springer, 2008.

[3] A.I.T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E. L.-H. Kuo, F. Y.-S. Lee, and B.-Y. Yang. SSE implementation of multivariate pkcs on modern x86 cpus. CHES 2009, LNCS vol. 5747, pp. 33–48. Springer, 2009.

[4] Ding J., Schmidt D.: Rainbow, a new multivariate polynomial signature scheme. In Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS vol. 3531, pp. 164–175 Springer, Heidelberg (2005)

[5] Ding, J., Yang, B.-Y., Chen, C.-H. O., Chen, M.-S., and Cheng, C.M.: New Differential-Algebraic Attacks and Reparametrization of Rainbow. In: LNCS 5037, pp.242–257, Springer, Heidelberg (2005)

[6] Ding, J., Wolf, C., Yang, B.-Y.: ℓ-invertible Cycles for Multivariate Quadratic Public Key Cryptography. In: Okamoto, T., Wang, X., (eds.): PKC 2007, LNCS, vol. 4450, pp. 266–281, Springer, Heidelberg (2007)

[7] Faugère, J.C.: A new efficient algorithm for computing Groebner bases (F4). Journal of Pure and Applied Algebra, 139:61–88 (1999)

[8] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, 1979

[9] Kipnis, A., Patarin, L., Goubin, L.: Unbalanced Oil and Vinegar Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS vol. 1592, pp. 206–222 Springer, Heidelberg (1999)

[10] Kipnis, A., Shamir, A.: Cryptanalysis of the Oil and Vinegar Signature scheme. In: Krawzyck, H. (ed.) CRYPTO 1998, LNCS vol. 1462, pp. 257–266 Springer, Heidelberg (1998)

[11] Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for efficient Signature-Verification and Message-Encryption. Advances in Cryptology - EUROCRYPT 1988, LNCS vol. 330, pp. 419–453, Springer, Heidelberg (1988)

[12] Patarin, J.: Hidden Field equations (HFE) and Isomorphisms of Polynomials (IP). In: Proceedings of EUROCRYPT'96, LNCS vol. 1070, pp. 38–48, Springer, Heidelberg (1996)

[13] Patarin, J,: The oil and vinegar signature scheme, presented at the Dagstuhl Workshop on Cryptography (September 97)

[14] Petzoldt, A. Bulygin, S., Buchmann, J.: A Multivariate Signature Scheme with a partially cyclic public key. In Proceedings of SCC 2010, pp. 229 - 235

[15] Petzoldt, A., Bulygin, S., Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme. In: Proceedings of PQCrypto'10, LNCS vol. 6061, pp. 218 -240, Springer, Heidelberg (2010)

[16] Petzoldt, A., Bulygin, S., Buchmann, J.: CyclicRainbow - A Multivariate Signature Scheme with a Partially Cyclic Public Key. In: Proceedings of INDOCRYPT'10, LNCS vol. 6498, pp. 33-48, Springer, Heidelberg (2010)

[17] Petzoldt, A., Bulygin, S., Buchmann, J.: Linear Recurring Sequences for the UOV Key Generation. In: Proceedings of PKC'11, LNCS vol. 6571, pp. 335-350, Springer, Heidelberg (2011)

[18] Shor, P.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput. 26 (5): pp. 1484–1509.

[19] E. Thomae, C. Wolf: Solving underdetermined Systems of Multivariate Quadratic Equations Revisited. PKC 2012, LNCS vol. 7293, pp. 156 - 171. Springer 2012.

A Algorithm for the verification process of cyclicRainbow

Algorithm 2 shows the improved verification process for Rainbow schemes with two layers and partially circulant public key. The algorithm can be extended to Rainbow schemes with more than two layers in a natural way.

Algorithm 2 Verification process for cyclicRainbow

for $i = 1$ to $v_2 - 1$ **do** ▷ First polynomial
  $a_i \leftarrow \sum_{j=1}^{\min(i,v_1)} MP_{ji}^{(1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v_1 + 1$ to $v_2 - 1$ **do**
  $\text{temp}_i \leftarrow a_i + \sum_{j=v_1+1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
end for
for $i = v_2$ to $n + 1$ **do**
  $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
end for
$h'_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = 2$ to o_1 **do** ▷ Polynomials 2 to o_1
  for $i = v_2 + 1$ to $n + 1$ **do**
    $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_2$ to $v_1 + 1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_1$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(l)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(l)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for

Algorithm 2 Verification process for cyclicRainbow (cont.)

$\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(o_1+1)} \cdot \text{sign}_j$ ▷ (o_1+1)-th polynomial
for $i = n$ to $v_2 + 1$ by -1 **do**
  $a_i \leftarrow \sum_{j=1}^{v_2} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i + \sum_{j=v_2+1}^{i} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
end for
for $i = v_2$ to $v_1 + 1$ by -1 **do**
  $a_i \leftarrow a_{i-1} + \sum_{j=v_1+1}^{i} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v_1$ to 2 by -1 **do**
  $a_i \leftarrow a_{i-1} + MP_{ii}^{(o_1+1)} \cdot \text{sign}_i$
  $\text{temp}_i \leftarrow a_i$
end for
$a_1 \leftarrow MP_{11}^{(o_1+1)} \cdot \text{sign}_1$
$\text{temp}_1 \leftarrow a_1$
$h'_{o_1+1} \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = o_1 + 2$ to $o_1 + o_2$ **do** ▷ Polynomials $o_1 + 2$ to $o_1 + o_2$
  $\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(l)} \cdot \text{sign}_j$
  for $i = n$ to $v_2 + 1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v_2+1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_2$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(l)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(l)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for
if $h_l = h'_l \; \forall l \in \{1, \ldots, m\}$ **then return** "ACCEPT" ▷ TEST
else return "REJECT"
end if