Table of Contents

A Multivariate Signature Scheme with an almost cyclic public key

Albrecht Petzoldt[1] and Johannes Buchmann[1]

TU Darmstadt, FB Informatik
Hochschulstrasse 10, 64289 Darmstadt, Germany
{apetzoldt,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. Multivariate public key cryptography is one of the main approaches to guarantee the security of communication in a post quantum world. One of the major drawbacks in this area is the huge size of the public key. In this paper we present a new idea to create a multivariate signature scheme with an almost cyclic public key. The scheme is very similar to the UOV-Scheme of Kipnis and Patarin but reduces the size of the public key by about 83 %.

Keywords: Multivariate Cryptography, UOV Signature Scheme, cyclic public key

1 Introduction

During the last two decades, multivariate cryptography has become a major field of research and many schemes both for encryption and signatures have been proposed. This is mainly due to the need of an alternative to classical public-key cryptosystems like RSA or ECC which can be broken when quantum computers arrive. Multivariate public-key cryptography enables fast en- and decryption as well as signature generation and verification. Since the operations needed are very simple (addition and multiplication over small fields with characteristic 2), multivariate schemes can be implemented on low cost smart cards. However, multivariate cryptography is not yet widely spread, because of the large size of its public keys.

In this paper we present a new idea how to decrease the size of the public key. The principle idea is to compute the coefficients of the private key in such a way, that the corresponding public key gets a nice structure. So it is possible to compute most of the public coefficients out of the coefficients of the first public polynomial, which decreases the storage needed for the public key by a large factor.

In Section 2 we shortly discuss the UOV Signature Scheme which is the basis of our new scheme. Section 3 describes the new scheme in detail. Section 4 contains a short security analysis of the scheme as well as example parameters and Section 5 concludes the paper. A toy example of our scheme can be found in the appendix of this paper.

2 The Oil and Vinegar Signature Scheme

One classical way to build a multivariate signature scheme is the so called (Unbalanced) Oil and Vinegar Signature Scheme ([Pa97], [KP99]).

Let K be a finite field (for example $K = GF(2^8)$). Let o and v be two integers and set $n = o+v$. Patarin suggested to choose $o = v$. After this original scheme was broken by Kipnis and Shamir

in [KS98], it was recommended in [KP99] to choose $v > o$ (Unbalanced Oil and Vinegar (UOV)). In this section we describe the more general approach UOV.
We set $V = \{1, \ldots, v\}$ and $O = \{v+1, \ldots, n\}$. Of the n variables $x_1, \ldots, x_n$ 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

$$f_k(\mathbf{x}) = \sum_{i \in V,\ j \in O} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V,\ i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)} \ (k \in O)$$

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. Such we 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 the system does not have a solution, choose other values of $x_1, \ldots, x_v$ and try again).

In the public key, the central map is hidden by composing it with a linear map $\mathcal{T} : K^n \to K^n$. So the public key of the scheme is given by $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ and is difficult to invert. The private key of the scheme consists of the maps $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert the public key.

Signature Generation To sign a signature m, one computes a hashvalue $h \in K^o$ of m and then recursively $y = \mathcal{F}^{-1}(h)$ and $z = \mathcal{T}^{-1}(y)$. Here, $\mathcal{F}^{-1}(y)$ means to find a preimage of y under the central map $\mathcal{F}$. The Signature of the message m is $z \in K^n$.

Signature Verification To verify the authenticity of a signature, one simply computes $h' = \mathcal{P}(z)$. If h' equals h, the signature is accepted, otherwise rejected.

Note that the $\frac{(n+1)\cdot(n+2)-o\cdot(o+1)}{2}$ coefficients of each of the central polynomials $f_1, \ldots f_o$ can be chosen arbitrarily. In the next Section we look at the question how they can be chosen to create a partially cyclic public key.

3 Description of the Scheme

3.1 Construction of $\mathcal{F}$ and $\mathcal{P}$

If we choose a monomial ordering (e.g. reversed graded lexicographical ordering) we can write the coefficients of the private and public polynomials into two $o \times \frac{(n+1)\cdot(n+2)}{2}$ matrices $F = (f_{ij})$ and $P = (p_{ij})$. In the matrix F, the columns corresponding to quadratic monomials of the Oil$\times$Oil type contain only zeros (so F contains $\frac{o\cdot(o+1)}{2}$ zero columns).
The linear map $\mathcal{T}$ is represented by an $n \times n$ matrix T.

Note: For the simplification of our notation we start counting rows and columns of a matrix by 0. So the top left element of the matrix $A = (a_{ij})$ will be called a_{00}.
Therefore, the first public polynomials coefficient of the term x_1^2 will be denoted by p_{00}, that of the term $x_1 x_2$ by p_{01} ans so on.

In the following we try to create a public map $\mathcal{P}$ such that the highest possible number r of the columns of the corresponding matrix P will be cyclic.

If we use the reversed graded lexicographical ordering, the p_{ij} $\left(0 \leq j < \frac{n\cdot(n+1)}{2}\right)$ are the coefficients of the quadratic monomials. Since $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ for a linear map $\mathcal{T}$, we have

$$p_{ij} = \sum_{k=0}^{\frac{n\cdot(n+1)-o\cdot(o+1)-2}{2}} (f_{ik} \cdot (\text{ some quadratic term in the } t_{ij})) \quad \left(0 \leq j < \frac{n\cdot(n+1)}{2}\right) \qquad (1)$$

So, if we fix the values of the t_{ij}, the p_{ij} ($j < \frac{n\cdot(n+1)}{2}$) will be given as linear combinations of the f_{ij}.

Note that the f_{ij} ($\frac{n\cdot(n+1)-o\cdot(o+1)}{2} \leq j < \frac{n\cdot(n+1)}{2}$) correspond to quadratic monomials of the Oil×Oil type and are 0. That's why you have to sum only up to $\frac{n\cdot(n+1)-o\cdot(o+1)-2}{2}$.

Now we set for a randomly chosen vector $\mathbf{b} = (b_0, \ldots b_{r-1})$

$$p_{ij} = b_{j-i \bmod r} \; \forall j \in \{0, \ldots, r-1\} \tag{2}$$

Together with formula (2), the first r equations of formula (1) lead for each $i \in \{0, \ldots, o-1\}$ to a system of r linear equations in the $\frac{n\cdot(n+1)-o\cdot(o+1)}{2}$ coefficients f_{ij} $\left(0 \leq j < \frac{n\cdot(n+1)-o\cdot(o+1)}{2}\right)$.

By solving these systems of linear equations, we get the non-zero coefficients of the quadratic terms of the central polynomials.
Note, that if you choose r too big, the upper system will not have a solution. However, our experiments show, that the choice $r = \frac{n\cdot(n+1)-o\cdot(o+1)}{2}$ is possible.
The other coefficients of the central map $\mathcal{F}$ are chosen at random.
Since we know both $\mathcal{T}$ and $\mathcal{F}$ now, we can easily compute the remaining coefficients of the public polynomials.

Note that we do not have to store the whole public key any longer. By our construction you only have to store the columns $r, \ldots, \frac{(n+1)\cdot(n+2)-2}{2}$ of P and the vector $\mathbf{b}$. So, the parameter r determines the rate by which the size of the public key is reduced.

3.2 The Scheme

Key Generation
parameters: Underlying field K (e.g. $K = GF(2^8)$),
o (number of Oil variables), v (number of Vinegar variables), r (reduction factor)
The number of variables will be $n = o + v$.

1. Choose a vector $\mathbf{b} = (b_0, \ldots, b_{r-1})$ at random.
2. Choose an $n \times n$ matrix T at random. T must be invertible.
3. Set the entries of the first r columns of P to
$$p_{ij} = b_{j-i \bmod r}.$$
4. Solve for $i = 0, \ldots, o-1$ the linear systems given by the equations (1) (for $j < r$) to get the non-zero coefficients of the quadratic terms of the central map F.
5. Choose the coefficients of the linear and constant terms of the central map at random.
6. Compute the remaining coefficients of the public polynomials by composing $\mathcal{F}$ and $\mathcal{T}$.

The *public key* of the scheme consists of the columns $r, \ldots, \frac{(n+1)\cdot(n+2)-2}{2}$ of P and the vector $\mathbf{b}$. Moreover, it contains a rule how to compute the public map $\mathcal{P}$ out of the key.
So the size of the public key will be $o \cdot \frac{(n+1)\cdot(n+2)-2r}{2} + r$ byte.
The *private key* consists of the matrices F and T. Its size is $o \cdot \frac{(n+1)\cdot(n+2)}{2} + n^2$ byte (as in the case of the UOV Scheme)

Signature Generation and Verification The Signature Generation and Verification works as in the case of the UOV Scheme. To *sign* a message with a hash value $h \in K^o$, we compute recursively $y = \mathcal{F}^{-1}(x)$ and $z = \mathcal{T}^{-1}(y)$. The signature of the message is $z \in K^n$.
To *verify* the authenticity of the signature, one computes $h' = \mathcal{P}(z)$. If $h' = h$ holds, the signature is accepted, otherwise rejected.

4 Security

4.1 Security under direct attacks

We carried out some experiments with MAGMA, which contains an implementation of Faugeres F4 algorithm. To see whether one can exploit the special structure of our public key, we compared the running time of F4 on a random system to that of our scheme with a partially cyclic key. The results are shown in the table below.

# equations	11	12	13	14	15	16
random	1894 s	10980 s	88832 s	527500 s	4540928 s	28345344 s
cyclic	1765 s	10368 s	82688 s	485120 s	4152320 s	26208265 s
	93 %	94 %	93%	92%	91%	92 %

As you can see from the Table, MAGMA works slightly faster on our scheme than on a random system. However, the speed up factor is so small that it isn't a major drawback of our scheme.

4.2 UOV-Reconciliation

During the algebraic part of the UOV-Reconciliation attack some of the structure of our public key gets lost. Because of that we don't believe that the running time of this attack will be decreased by a major factor. However, we haven't proved this yet.

4.3 Parameters

Since we haven't made a detailed security analysis of our scheme yet, the proposed parameters have to be seen with great reservation. Yet we propose (as for UOV): $(o, v) = (24, 48)$. We get $r = \frac{(o+v)\cdot(o+v+1)-o\cdot(o+1)}{2} = 2328$. So, the size of the public key is 11.3 kB. This means a reduction of 83 % to the UOV Scheme (public key size: 64.8 kB)

5 Conclusion and Future Work

We think that the idea presented in this paper might be an interesting approach to reduce the size of the public key. However, there remains a lot of work to be done. Some points we want to adress in the future are

- complete security analysis of our scheme (especially rank attacks)
- effect of a random choice of the cyclic columns (instead of choosing the first r)
- Extension of the idea to other schemes (e.g. Rainbow)
- effect of a cyclic structure on algorithms like XL

Additionally we want to invite anybody to send us his ideas of attacking or strengthening our scheme.

6 Acknowledgements

The first author wants to thank Jintai Ding and Stanislav Bulygin for many helpful hints.

References

[KP99] 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)

[KS98] 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)

[Pa97] Patarin, J,: The oil and vinegar signature scheme, presented at the Dagstuhl Workshop on Cryptography (september 97)

A Toy example

In our example we set $K = GF(17)$ and $(o, v) = (2, 3)$. Then we have $r = \frac{(o+v)\cdot(o+v+1)-o\cdot(o+1)}{2} = 12$. We choose our vector $\mathbf{b}$ and the matrix T to

$$\mathbf{b} = (5, 1, 6, 3, 2, 13, 2, 15, 13, 6, 10, 4), \quad T = \begin{pmatrix} 4 & 9 & 11 & 4 & 15 \\ 13 & 1 & 9 & 9 & 11 \\ 14 & 2 & 12 & 15 & 1 \\ 4 & 4 & 2 & 15 & 9 \\ 13 & 8 & 11 & 15 & 12 \end{pmatrix}$$

Our preliminary public key is (see equation (2))

$$P = \begin{pmatrix} 5 & 1 & 6 & 3 & 2 & 13 & 2 & 15 & 13 & 6 & 10 & 4 & \star & \star & \star & \star & \star & \star & \star & \star & \star \\ 4 & 5 & 1 & 6 & 3 & 2 & 13 & 2 & 15 & 13 & 6 & 10 & \star & \star & \star & \star & \star & \star & \star & \star & \star \end{pmatrix}$$

By the first 12 equations of (1) we get

$$(f_{0,0}, \dots, f_{0,11}) = (12, 13, 10, 12, 2, 13, 8, 9, 8, 1, 11, 13)$$
$$(f_{1,0}, \dots, f_{1,11}) = (10, 7, 14, 5, 14, 4, 15, 11, 0, 4, 0, 8)$$

The next three coefficients of each central polynomial are those of the Oil×Oil-terms and have to be zero.
If we choose the remaining coefficients of the central map at random, we may get

$$F = \begin{pmatrix} 12 & 13 & 10 & 12 & 2 & 13 & 8 & 9 & 8 & 1 & 11 & 13 & 0 & 0 & 0 & 0 & 9 & 13 & 16 & 2 & 7 \\ 10 & 7 & 14 & 5 & 14 & 4 & 15 & 11 & 0 & 4 & 0 & 8 & 0 & 0 & 0 & 9 & 5 & 7 & 0 & 3 & 1 \end{pmatrix}$$

By composing F with T we get the public key of our scheme

$$P = \begin{pmatrix} 5 & 1 & 6 & 3 & 2 & 13 & 2 & 15 & 13 & 6 & 10 & 4 & 15 & 10 & 10 & 15 & 13 & 2 & 2 & 8 & 7 \\ 4 & 5 & 1 & 6 & 3 & 2 & 13 & 2 & 15 & 13 & 6 & 10 & 1 & 10 & 12 & 0 & 5 & 6 & 10 & 12 & 1 \end{pmatrix}$$

respectively

$$\begin{aligned} P = (&5x_1^2 + x_1x_2 + 6x_1x_3 + 3x_1x_4 + 2x_1x_5 + 13x_2^2 + 2x_2x_3 + 15x_2x_4 + 13x_2x_5 + 6x_3^2 + 10x_3x_4 + 4x_3x_5 \\ &+ 15x_4^2 + 10x_4x_5 + 10x_5^2 + 15x_1 + 13x_2 + 2x_3 + 2x_4 + 8x_5 + 7, \\ &4x_1^2 + 5x_1x_2 + x_1x_3 + 6x_1x_4 + 3x_1x_5 + 2x_2^2 + 13x_2x_3 + 2x_2x_4 + 15x_2x_5 + 13x_3^2 + 6x_3x_4 + 10x_3x_5 \\ &+ x_4^2 + 10x_4x_5 + 12x_5^2 + 5x_2 + 6x_3 + 10x_4 + 12x_5 + 1) \end{aligned}$$

From the second polynomial of the public key you have to store only the last 9 coefficients.