CNMAC2012
XXXIV CONGRESSO NACIONAL DE
MATEMÁTICA APLICADA E COMPUTACIONAL
17 a 21 de setembro de 2012 – Águas de Lindóia/SP

ISSN 1984-8218

Small private keys for systems of multivariate quadratic equations using symmetric cryptography

Fábio Borges[*], **Albrecht Petzoldt**,
Fachbereich Informatik, TU Darmstadt
Hochschulstraße 10, D-64289, Darmstadt, Germany
E-mail: fabio.borges@cased.de, apetzoldt@cdc.informatik.tu-darmstadt.de,

Renato Portugal
National Laboratory for Scientific Computing - LNCC
25651-075, Petrópolis, RJ, Brazil
E-mail: portugal@lncc.br.

Abstract: *Systems of Multivariate Quadratic Equations ($\mathcal{MQ}$) are important in cryptography due to the resistance against attacks that will arise with the advent of quantum computing. Resistant algorithms against attacks based on quantum computing are called post-quantum cryptography. Unbalanced Oil-Vinegar (UOV) is a well known post-quantum signature scheme based on $\mathcal{MQ}$. This paper presents a variation of the implementation of UOV. The proposed and default schemes were implemented in* `Java` *using* `FlexiProvider` *library, and they were compared with each other. The results present a faster processing time and a reduction in private key size. The proposed implementation was inspired by cryptographic symmetric algorithm RC4 to generate the private key. The size of the private key is independent of the parameters chosen to UOV.*

Key-words: *Multivariate Quadratic Equation, Post-Quantum Cryptography, Unbalanced Oil and Vinegar (UOV), Signature Scheme.*

1 Introduction

Currently all digital signature schemes that ensure Internet security are based on the problem of integer factorization or the discrete logarithm. However, do to Shor's algorithm [9], these problems can be easily solved on a quantum computer of appropriate size. Therefore one needs alternatives to the classical public-key schemes like RSA [8], DH [1] and ECDLP [5, 4], so called Post-Quantum Schemes.

There are several proposed post-quantum algorithms, however $\mathcal{MQ}$ based schemes are the most promising: *Multivariate public key cryptography is one of the main approaches to guarantee on the security of communication in the post-quantum world* [6].

One of the best known multivariate signature schemes is the UOV scheme of Kipnis and Patarin [3]. Since the operations needed by UOV are computationally very simple, it is very fast and can also be used on small devices like RFID chips and smart cards. However, the key sizes of UOV are relatively large.

In this paper we present a variation of the standard UOV implementation by which we can create smaller private keys as well as speed up the processing time, and also present a comparison with the default implementation. On [7] can be found a scheme to reduce the public key, this work presents a scheme to reduce the private key.

[*]I would like to thank to PCI/LNCC for funding me to visit the CASED at end of 2010 and also to Prof. Johannes Buchmann and Dr. Stanislav Bulygin for discussions about $\mathcal{MQ}$.

ISSN 1984-8218

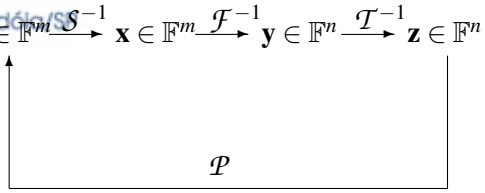

Figure 1: Signature generation and verification.

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 $\mathcal{MQ}$: 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 $GF(q)$.

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. The standard process for signature generation and verification works as follows 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-images 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.

3 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.

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 $O = \{1, \ldots, o\}$. We call $\check{x}_1, \ldots, \check{x}_v$ the Vinegar variables and $x_1, \ldots, x_o$ Oil variables. We define o quadratic polynomials $f^{(k)}(\mathbf{x}) = f^{(k)}(x_1, \ldots, x_o, \check{x}_1, \ldots, \check{x}_v)$ by

$$f^{(k)}(\mathbf{x}) = \sum_{i \in V,\ j \in O} \alpha_{ij}^{(k)} \check{x}_i x_j + \sum_{i,j \in V,\ i \leq j} \beta_{ij}^{(k)} \check{x}_i \check{x}_j \tag{1}$$

Note that Oil and Vinegar variables are not fully mixed.

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 $\check{x}_1, \ldots, \check{x}_v$ at random. Therefore we get a system of o linear equations in the o variables $x_1, \ldots, x_o$ which can be solved e.g. by Gaussian Elimination. If the system does not have a solution, one has to choose other values of $\check{x}_1, \ldots, \check{x}_v$ 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 $\mathcal{T}$.

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 left out.

The UOV signature scheme over $GF(2^8)$ is commonly believed to be secure for $o \geq 26$ equations and $v = 2 \cdot o$ Vinegar variables. For UOV schemes over $GF(31)$ we need at least $o = 33$ equations and $v = 2 \cdot o$ Vinegar variables.

ISSN 1984-8218

4 Implementation Details

In our implementation we use a matrix representation of the polynomials. A homogeneous quadratic polynomial f can be represented as a matrix Q following the relation

$$f(x_1,\ldots,x_o,\check{x}_1,\ldots,\check{x}_v) = \vec{x}Q_i\vec{x}^T$$

where $\vec{x} = (x_1,\ldots,x_o,\check{x}_1,\ldots,\check{x}_v)$. Since $x_ix_j = x_jx_i \forall i,j$, we have diagonal superior matrices.

For instance, let $\mathbb{F}$=GF(4), $o = v = 3$ and α be a generator of $\mathbb{F}$. Let f be the polynomial

$$\begin{aligned} f &= \alpha x_1\check{x}_1 + \alpha x_1\check{x}_2 + x_2\check{x}_1 + x_2\check{x}_3 + \alpha^2 x_3\check{x}_1 \\ &+ x_3\check{x}_2 + \alpha^2 x_3\check{x}_3 + \alpha^2\check{x}_1\check{x}_2 + \check{x}_1\check{x}_3 \\ &+ \check{x}_2\check{x}_3 + \alpha\check{x}_3^2 \end{aligned} \quad . \tag{2}$$

of type (1), then the matrix representation of f is given by

$$Q = \begin{pmatrix} 0 & 0 & 0 & \alpha & \alpha & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & \alpha^2 & 1 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha^2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \alpha \end{pmatrix} = \left(\begin{array}{c|c} 0 & A \\ \hline 0 & B \end{array}\right),$$

where A and B are block matrices of size 3×3. These block matrices are used in our implementation of the scheme. Note that, due to the special structure of the polynomial (2), the upper left submatrix of Q is zero. Algorithm 1 generates a set of matrices for the UOV private key.

Algorithm 1 UOV Private Key

Require: Integers o and v
Ensure: Private Key

1: **for** $i < o$ **do**
2: $\quad Q_i \leftarrow$ random Oil-Vinegar matrix
3: **end for**
4: $L_{n\times n} \leftarrow$ an invertible random matrix
5: **return** Q_i $(i = 0,\ldots,o)$, L

Algorithm 2 generates a public key associated with private key from Algorithm 1. An attacker, observing the public key but not having access to private key, must attempt to recover Q_i from $\bar{Q}_i$. It is assumed that the attacker does have knowledge of the algorithm but does not know L and Q_i.

Algorithm 2 UOV Public Key

Require: Private Key
Ensure: Public Key

1: **for** $i < o$ **do**
2: $\quad \bar{Q}_i \leftarrow L^T Q_i L$
3: **end for**
4: **return** $\bar{Q}_i$ $(i = 0,\ldots,o)$

Algorithm 3 shows how to sign a message that might be a hash. This is the analogous operation with polynomials represented as matrices. More informations about the operations with polynomials can be found in [2].

The recipient may verify the signature using Algorithm 4.

ISSN 1984-8218

Algorithm 3 UOV Signature

Require: Private Key and Message $M = (m_1, \ldots, m_o)$
Ensure: Signature $(s_1, \ldots, s_n)$

1: **repeat**
2: $V = (\check{x}_1, \ldots, \check{x}_v) \leftarrow$ random values
3: **for** $i < o$ **do**
4: $F_i \leftarrow V \times \text{Submatrix}_A(Q_i)$
5: $y_i \leftarrow m_i + V \times \text{Submatrix}_B(Q_i) \times V^T$
6: **end for**
7: $z \leftarrow \text{GaussianElimination}(F, y)$
8: **until** gets a solution z
9: $(s_1, \ldots, s_n) \leftarrow (y_1, \ldots, y_o, z_1, \ldots, z_v) \times L^{-1}$
10: **return** $(s_1, \ldots, s_n)$

Algorithm 4 UOV Verifing the Signature

Require: Public Key, Message M and Signature S
Ensure: True or False

1: **for** $i < o$ **do**
2: $W_i \leftarrow S\bar{Q}_i S^T$
3: **end for**
4: **if** $W = M$ **then**
5: **return** True
6: **else**
7: **return** False
8: **end if**

4.1 Paper Contribution

The main contribution of this paper lies in speeding up Algorithm 1. Instead of using pseudo-random numbers generated by the `FlexiProvider` function `getRandomElement` we use Algorithm 5. The permutations generated by the Algorithm 5 are inspired by the symmetric algorithm RC4 which is widely used on the Internet.

Algorithm 5 receives three parameters, a symmetric key K and states i and j. The states are started as zero and the key K is composed by elements of the field. The key size $|K|$ can be chosen independently of the UOV parameters. Moreover one has to store only K to recover the whole key.

Algorithm 5 Pseudo-random stream of bites based on RC4

Require: Private Key K, States i and j
Ensure: Next pseudo-random element of field

1: $i \leftarrow i + 1 \mod |K|$
2: $j \leftarrow j + K_j \mod |K|$
3: $t \leftarrow K_i$
4: $\text{swap}(K_i, K_j)$
5: $Element \leftarrow (K_i + K_j) \times K_{i+j} \mod |K|$
6: **return** $Element$

Each algorithm was implemented in `Java` using `FlexiProvider` library. The following commands were used: `GF2mField k`, `k.add`, `k.mult`, and `k.getRandomElement`.

Some popular algorithms were also implemented, namely matrix multiplication over GF(2^m) and transpose, Gaussian elimination for the equation system over GF(2^m), and Gauss-Jordan elimination for the inverse matrix over GF(2^m).

ISSN 1984-8218

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Table 1: Operations of multiplication (left) and addition (right) of elements in $\mathbb{F}$.

This approach can be used in other languages like `C`. The use of the function `rand()` is normally not enough. Looking to the prototype: `void srand ( unsigned int seed );` we can see that the bit size of the seed might be a small number. For cryptographic purposes, the length of the seed must be at least 80 bit.

4.2 Validation using a Numerical Example

For instance we consider an unbalanced UOV scheme ($o \neq v$), where all the operations are performed over the finite field $\mathbb{F}$=GF(2^2) = GF(2)$[X]/ < 1+x+x^2 >$. The elements of $\mathbb{F}$ can be represented as the numbers $\{0,1,2,3\}$. The additions and multiplications are done according to Table 1. Fields of characteristics 2 are very interesting because the operations are fast, since they can be performed using `xor` and `shift`.

Let $K = \{2,2,1,3,2,2,3,1,3,2\}$ and choose $o = 3$ and $v = 5$. Thus, the matrices have dimension n^2 where $n = o+v = 8$ and the UOV private key consists out of $\{L, Q_0, Q_1, Q_2\}$. Note that we don't need to store these four matrices, but only K. By applying Algorithm 5 we get

$$L = \begin{pmatrix} 2&0&0&0&2&0&1&0\\ 0&0&0&2&0&0&1&0\\ 1&2&3&2&0&3&0&1\\ 0&2&0&0&1&3&0&2\\ 0&3&1&2&3&2&1&3\\ 3&0&1&1&3&1&2&1\\ 1&3&0&1&1&3&2&1\\ 2&1&1&0&3&2&3&3 \end{pmatrix}, \quad \text{and} \quad Q_0 = \begin{pmatrix} 0&0&0&2&2&1&1&0\\ 0&0&0&2&2&3&3&3\\ 0&0&0&1&2&1&0&1\\ 0&0&0&3&1&3&1&2\\ 0&0&0&0&2&1&0&2\\ 0&0&0&0&0&1&2&3\\ 0&0&0&0&0&0&1&3\\ 0&0&0&0&0&0&0&1 \end{pmatrix},$$

$$Q_1 = \begin{pmatrix} 0&0&0&2&0&3&2&1\\ 0&0&0&2&1&1&2&2\\ 0&0&0&0&3&3&3&1\\ 0&0&0&3&1&3&1&0\\ 0&0&0&0&3&1&1&1\\ 0&0&0&0&0&1&3&1\\ 0&0&0&0&0&0&2&0\\ 0&0&0&0&0&0&0&3 \end{pmatrix}, \quad \text{and} \quad Q_2 = \begin{pmatrix} 0&0&0&1&2&3&2&1\\ 0&0&0&3&3&0&1&1\\ 0&0&0&3&1&2&1&1\\ 0&0&0&3&0&1&1&3\\ 0&0&0&0&2&1&2&1\\ 0&0&0&0&0&1&0&3\\ 0&0&0&0&0&0&2&3\\ 0&0&0&0&0&0&0&3 \end{pmatrix}$$

The public key consists of $\{\bar{Q}_0, \bar{Q}_1, \bar{Q}_2\} = \{L^T Q_0 L, L^T Q_1 L, L^T Q_2 L\}$.

Suppose that we want to send a message $M = (m_0, m_1, m_2) = (2,2,1)$ with a signature $S = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$. To find a valid signature we choose the Vinegar variables at random, say $V = (\check{x}_0, \check{x}_1, \check{x}_2, \check{x}_3, \check{x}_4) = (1,0,2,2,3)$.

$$\begin{pmatrix} 2&0&0\\ 3&2&3\\ 0&2&1 \end{pmatrix} \begin{pmatrix} x_0\\ x_1\\ x_2 \end{pmatrix} = \begin{pmatrix} 0\\ 2\\ 3 \end{pmatrix}.$$

Thus we get $\vec{x} = (0,0,3)$ and our signature is $\sigma = (2,3,2,1,3,1,2,0)$. The authenticity of the signature can be verified by computing

$$\tilde{m}_i = S\bar{Q}_i S^T \ (i = 0, \ldots, 2).$$

ISSN 1984-8218

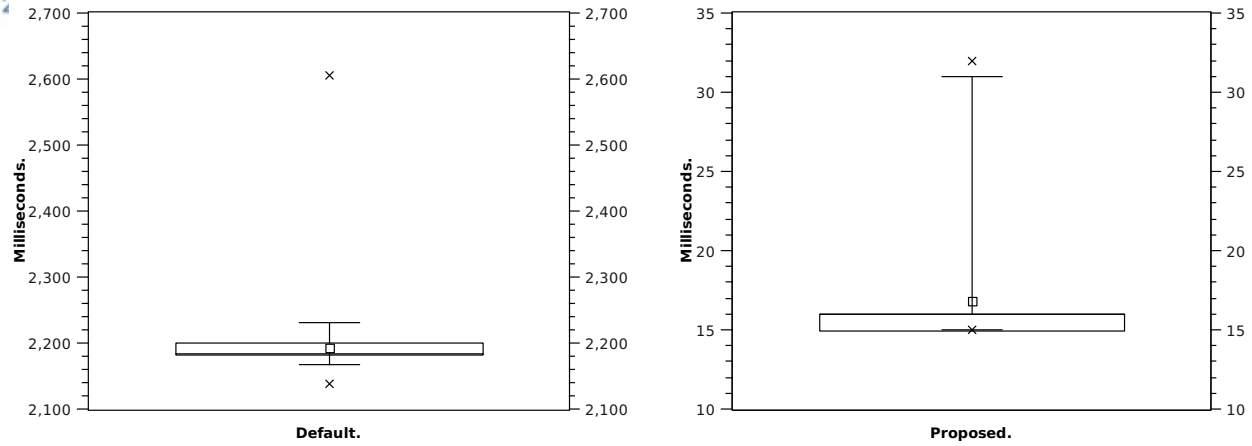

Figure 2: Processing time of implementations showed as box-plot.

Since

$$\tilde{m} = (2,2,1) = m,$$

the signature is accepted.

5 Simulation Details and Results

The scheme was implemented in `Java` version 1.7.0_02-b13 64 bits using `FlexiProvider` packages version 1.7p3 as a library. We run our program on an Intel Core(TM)2 Duo CPU T9400 2.53GHz processor with 4GB of memory. The operating system used was Windows 7 Enterprise 64 bit. The private key of the scheme is chosen using a pseudo-random number generator. In the default version we used the `FlexiProvider` command `getRandomElement`, in the proposed version we expanded a short seed of $|K|$ field elements using Algorithm 5. We used the UOV parameters $o = 26$ and $v = 52$ and the operations were done over the binary finite field $\mathbb{F}$=GF(2^8) = GF(2)$[X]/ < 1 + x + x^3 + x^4 + x^8 >$. With these parameters, we have to generate results 77 142 pseudo-random numbers over GF(2^8) for the private key, hence for each simulation.

To obtain statistical confidence in our results, 1 000 simulation runs were used for each of the two versions. The default implementation resulted in 269 systems of equations without solution, and the proposed implementation resulted in 246 systems of equations of 1 000 simulations. The default and the proposed implementation generated 3 and 4 singular matrices L, respectively.

From Figure 2 it can be seen that the default implementation needs much more time and has bigger standard deviation than the proposed implementation.

Table 2 presents the mean time in milliseconds from the main steps to use UOV.

6 Conclusion

The proposed implementation requests to store an arbitrary smaller key K than in the default implementation. The size of the key stored is independent of other UOV parameters. A user can memorize K and the values of o and v, and can use K to generate the private key.

Simulations verify that the proposed implementation is also faster than the default implementation using the `FlexiProvider` command `getRandomElement`. The machine and software chosen are not appropriate to benchmark, since it is possible to develop faster implementation in `C`. However, the difference between the standard and the proposed implementations remains, if the implementation uses a

ISSN 1984-8218

Implementations	Mean Time		Standard Deviation	
	Default	Proposed	Default	Proposed
Private Key	2,193.02	16.85	25.98	4.25
Public Key	1,945.26	1,965.41	28.28	18.34
Signature	115.68	116.43	7.51	8.02
Verify	35.87	35.77	7.16	7.14

Table 2: Time elapsed on average in milliseconds and standard deviations obtained from implementations.

secure pseudo-random number generator. Other symmetric algorithms can also be used in $\mathcal{MQ}$ schemes in order to increase the security or speed.

References

[1] W. DIFFIE AND M. E. HELLMAN, *New directions in cryptography*, IEEE Trans. Information Theory, IT-22 (1976), pp. 644–654.

[2] J. DING, J. E. GOWER, AND D. SCHMIDT, *Multivariate Public Key Cryptosystems*, vol. 25 of Advances in Information Security, Springer, 2006.

[3] A. KIPNIS, J. PATARIN, AND L. GOUBIN, *Unbalanced oil and vinegar signature schemes*, in EUROCRYPT, J. Stern, ed., vol. 1592 of Lecture Notes in Computer Science, Springer, 1999, pp. 206–222.

[4] N. KOBLITZ, *Elliptic curve cryptosystems*, Mathematics of Computation, 48 (1987), pp. 203–209.

[5] V. S. MILLER, *Use of elliptic curves in cryptography*, in Advances in cryptology—CRYPTO '85 (Santa Barbara, Calif., 1985), vol. 218 of Lecture Notes in Comput. Sci., Springer, Berlin, 1986, pp. 417–426.

[6] A. PETZOLDT, S. BULYGIN, AND J. BUCHMANN, *A multivariate signature scheme with a partially cyclic public key*, in Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography, Jun 2010, pp. 229–235.

[7] A. PETZOLDT, E. THOMAE, S. BULYGIN, AND C. WOLF, *Small public keys and fast verification for multivariate quadratic public key systems*, in Proceedings of the 13th international conference on Cryptographic hardware and embedded systems, CHES'11, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 475–490.

[8] R. L. RIVEST, A. SHAMIR, AND L. ADLEMAN, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM, 21 (1978), pp. 120–126.

[9] P. W. SHOR, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.