FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Multivariate cryptography
Student:	Bc. Jan Rahm
Supervisor:	Ing. Jiří Buček, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Security
Department:	Department of Information Security
Validity:	Until the end of summer semester 2020/21

Instructions

Study the topic of multivariate cryptography as one of the approaches to post-quantum cryptography.
Select a specific algorithm based on multivariate cryptography such as Unbalanced Oil and Vinegar (UOV).
Create an educational implementation of the selected algorithm in Wolfram Mathematica.
Examine the reference implementation of the selected algorithm. Evaluate its time and memory complexity on a PC. Implement the algorithm on a chosen microcontroller such as ARM or ESP32 and evaluate its usability in an embedded environment.
Compare the time and memory complexity of the selected algorithm with a conventional algorithm such as RSA or ECDSA.

References

Will be provided by the supervisor.

prof. Ing. Róbert Lórencz, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 5, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Multivariate cryptography

Bc. Jan Rahm

Department of Information Security
Supervisor: Ing. Jiří Buček, Ph.D.

March 9, 2020

Acknowledgements

I would like to thank Ing. Jiří Buček, Ph.D. for the willingness, consultation and valuable advice he gave me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on March 9, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Rahm, Jan. *Multivariate cryptography.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020. Also available from: ⟨`https://github.com/rahmjan/Diploma_Thesis`⟩.

Abstrakt

Diplomová práce se zabývá vybranými algoritmy multivariační kryptografie, zejména Unbalanced Oil & Vintage a Rainbow. Cílem práce je implementace algoritmů ve Wolfram Mathematica, prozkoumání již existujících řešeních a jejich implementace na mikrokontroleru ESP32. Algoritmy jsou otestovány a změřeny vůči algoritmům RSA a ECDSA.

Klíčová slova Multivariační kryptografie, Unbalanced Oil & Vintage, Rainbow, Wolfram Mathematica, ESP32

Abstract

The diploma thesis deals with selected algorithms of multivariate cryptography, especially Unbalanced Oil & Vintage and Rainbow. The aim of this work is implementation of algorithms in Wolfram Mathematica, investigation of existing solutions and their implementation on ESP32 microcontroller. The algorithms are tested and measured against the RSA and ECDSA algorithms.

Keywords Multivariate cryptography, Unbalanced Oil & Vintage, Rainbow, Wolfram Mathematica, ESP32

Contents

List of Figures

Introduction

Cryptography is one of the most needed part of modern informatics because almost everyone has something they wish to stay private. But today we can see uprise of the quantum computers which are able to decipher the conventional algorithms for cryptology. That is why a new category of post-quantum cryptography was created and one of its candidates is multivariate cryptography.

The objective of this work is to describe principles of multivariate cryptography for educational purpose with creation of simple example in Wolfram Mathematica. The focus is on Unbalanced Oil & Vintage and Rainbow algorithms with examination of reference implementation. Further focusing on possible implementation on ESP32 and possible use in IoT.

The final part belongs to comparison with conventional algorithms which are RSA and ECDSA.

Chapter 1

Basic terms and definitions

The chapter describes concepts and algorithms used in the thesis.

1.1 Basic terms

1.1.1 Polynomial

Polynomial p is function to which applies

$$p(x) = \sum_{i=0}^{n} \alpha_i x^i = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + \alpha_n x^n,$$

where $n \in N_0$ and $\alpha_0, \alpha_1, ..., \alpha_n \in R$. Values $\alpha_0, \alpha_1, ..., \alpha_n$ we calls polynomial coefficients of p.

1.1.2 Degree of a polynomial

The degree of a polynomial is the highest index $i \in N_0$ to which applies that coefficient $\alpha_i \neq 0$. If all coefficients are zero, then the degree of the polynomial is -1.

1.1.3 Post-quantum cryptography

It refers to algorithms that are thought to be secure against an attack by a quantum computer.

But today it is not true for the most used cryptographic algorithms, which are based on mathematical problems of integer factorization, discrete logarithm or elliptic-curve discrete logarithm. These problems can be solved by Shor's algorithm on quantum computer.

1.1.4 Finite field

A finite field is a finite set which is a field. This means that multiplication, addition, subtraction and division (excluding division by zero) are defined and satisfy the rules of arithmetic known as the field axioms.

The simplest examples of finite fields are the fields of prime order: $\mathbb{F}_p$ may be constructed as the integers modulo p.

1.1.5 Translation

In Euclidean geometry, a translation is a geometric transformation that moves every point of a figure or a space by the same distance in a given direction.

1.1.6 Linear map

In mathematics, a linear map is a mapping $V \to W$ between two modules (for example, two vector spaces) that preserves the operations of addition and scalar multiplication.

1.1.7 Affine map

An affine map is the composition of two functions: a translation and a linear map. Ordinary vector algebra uses matrix multiplication to represent linear maps, and vector addition to represent translations. Formally, in the finite-dimensional case, if the linear map is represented as a multiplication by a matrix A and the translation as the addition of a vector $\vec{b}$, an affine map f acting on a vector $\vec{x}$ can be represented as:

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}$$

1.2 Multivariate cryptography

1.2.1 Definition

"Multivariate cryptography (MC) is the generic term for asymmetric cryptographic primitives based on multivariate polynomials over a finite field $\mathbb{F}$."[7]

It means it is system of nonlinear polynomial equations with coefficients over a finite filed $\mathbb{F} = \mathbb{F}_q$ with q elements:

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

$$p^{(2)}(x_1, \ldots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(2)} \cdot x_i + p_0^{(2)}$$

$$\vdots$$

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

If the polynomials are of degree two, they are called multivariate quadratics (MQ). Solving systems of multivariate polynomial equations is proven to be NP hard, so called MQ Problem. That is the reason why MC is often considered to be good candidate for post-quantum cryptography.

1.2.2 MQ Problem

Given m quadratic polynomials $p^{(1)}(x), \ldots, p^{(m)}(x)$ in the n variables $x_1, \ldots, x_n$, find a vector $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ such that $p^{(1)}(\bar{x}) = \ldots = p^{(m)}(\bar{x}) = 0$.

1.2.3 Public key

The public key of MC is system of MC polynomials. To build this system based on MQ Problem, it needs an easily invertible quadratic map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$, so called *central map*. Because it is easily invertible, it needs to be hidden in public key by invertible affine maps: $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$.

The public key of this system is composed map:

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^m$$

and the private key consists of the tree maps $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$, also known as *trapdoor*.

Public key should be hard to invert without the knowledge of the *trapdoor*.

$$\begin{array}{ccc} z \in \mathbb{F}^n & \xrightarrow{\mathcal{P}} & w \in \mathbb{F}^m \\ \mathcal{T}\big\downarrow & & \big\uparrow\mathcal{S} \\ y \in \mathbb{F}^n & \xrightarrow{\mathcal{F}} & x \in \mathbb{F}^m \end{array}$$

Figure 1.1: Workflow of multivariate public key cryptosystems

1.2.4 Encryption

To get a ciphertext w, message $z \in \mathbb{F}^n$ can by easily encrypted by evaluation of the public key $\mathcal{P}$:

$$w = \mathcal{P}(z) \in \mathbb{F}^m$$

For decryption of ciphertext, it needs to be evaluated by private key in tree steps:

$$x = \mathcal{S}^{-1}(w) \in \mathbb{F}^m,\ y = \mathcal{F}^{-1}(x) \in \mathbb{F}^n,\ z = \mathcal{T}^{-1}(y) \in \mathbb{F}^n$$

There is a condition that requires to be $m \geq n$, this way the public key $\mathcal{P}$ will be injective and decryption will output a unique plaintext.

1.2.5 Signature

To generate a signature for a message m, it needs to use a hash function:

$$\mathcal{H} : \{0,1\}^* \to \mathbb{F}^m$$

to compute the hash value:

$$w = \mathcal{H}(m) \in \mathbb{F}^m$$

After this step it can be evaluated by:

$$x = \mathcal{S}^{-1}(w) \in \mathbb{F}^m,\ y = \mathcal{F}^{-1}(x) \in \mathbb{F}^n,\ z = \mathcal{T}^{-1}(y) \in \mathbb{F}^n$$

where z is the signature of message m. As can be seen, it is similar to description of ciphertext.

The verification of signature z is done by computing the hash value:

$$w = \mathcal{H}(m) \in \mathbb{F}^m$$

and by evaluation of public key $\mathcal{P}$:

$$w' = \mathcal{P}(z) \in \mathbb{F}^m$$

If $w' = w$ is true, the signature is valid, otherwise not.

There is also condition that requires to be $m \leq n$, this way the public key $\mathcal{P}$ will be surjective and private key can sign any message.

1.3 UOV

The Unbalanced Oil and Vinegar's name comes from the fact that the variables of the polynomials are grouped into two groups: the vinegar and the oil. These two groups are mixed in the polynomials and the unbalanced attribute refers to the ratio of the variables, where is always more vinegar than oil variables. The signature scheme was proposed by Kipnis and Patarin in 1999.

1.3.1 Definition

Let $\mathbb{F}$ be a finite field, $v, o \in \mathbb{N}$ and $n = v + o$, $V = \{1, \ldots, v\}$, $O = \{v + 1, \ldots, n\}$. The variables $x_1, \ldots, x_v$ are Vinegar variables and $x_{v+1}, \ldots, x_n$ are Oil variables. If $v = o$ the scheme is called balanced Oil & Vinegar (OV), for $v > o$ it is UOV.

The *central map* $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^o$ consist of o quadratic polynomials $f^{(1)}, \ldots, f^{(o)}$:

$$f^{(k)} = \sum_{i=1}^{v}\sum_{j=1}^{v} \alpha_{ij}^{(k)} \cdot x_i x_j + \sum_{i=1}^{v}\sum_{j=v+1}^{n} \beta_{ij}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n} \gamma_i^{(k)} \cdot x_i + \delta^{(k)}$$

where $\alpha_{ij}^{(k)}$, $\beta_{ij}^{(k)}$, $\gamma_i^{(k)}$, $\delta^{(k)} \in \mathbb{F}$ and $1 \leq k \leq o$.

To hide $\mathcal{F}$ in the public key, it is combined with one invertible affine map $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. The public key of the scheme is in the form:

$$\mathcal{P} = \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^o$$

and the private key consist of $\mathcal{F}$ and $\mathcal{T}$. The second affine map $\mathcal{S}$ is not needed for the security of UOV.

1.3.2 Security

For the security of UOV is required $v \geq 2o$ because of the attack of Kipnis and Shamir on balanced OV.[8] Besides of that, the UOV scheme resisted (for suitable parameter sets) cryptanalysis for over 20 years now and is therefore believed to be of high security.

1.4 Rainbow

The Rainbow is a multi layer version of UOV. The layers are not independent from each other but there is a hierarchy which uses the results from the layer above to compute additional variables. The name comes from the link to the layers of a rainbow and the scheme was introduced by Ding and Schmid in 2005.

The main avantage compared to UOV should be in better performance, smaller key sizes and smaller signatures.

1.4.1 Definition

Let $\mathbb{F}$ be a finite field, $0 < v_1 < v_2 < \ldots < v_{u+1} = n$ be a sequence of integers and $V_i = \{1, \ldots, v_i\}$, $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ and $o_i = v_{i+1} - v_i$ $(i = 1, \ldots, u)$.

The *central map* $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ consist of $m = n - v_1$ quadratic polynomials $f^{(v_1+1)}, \ldots, f^{(n)}$:

$$f^{(k)} = \sum_{i,j \in V_l} \alpha_{ij}^{(k)} \cdot x_i x_j + \sum_{i \in V_l, j \in O_l} \beta_{ij}^{(k)} \cdot x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} \cdot x_i + \delta^{(k)}$$

where $l \in \{1, \ldots, u\}$ is the only integer such that $k \in O_l$.

To hide $\mathcal{F}$ in the public key, it is combined with two invertible affine maps $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ and $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$. The public key of the scheme is in the form:

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^m$$

and the private key consist of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$.

Chapter 2

Realization

Mathematica; Specific implementation + difference on IoT; (presentation for teaching)

2.1 Wolfram Mathematica

This section describes examples in Wolfram Mathematica and step by step description of algorithms.

2.1.1 UOV

Here is description of signature scheme of UOV. This example of UOV is in fact example of balanced OV because it simplify the explanation of the algorithm.

First set up the parameters of the example: Let $\mathbb{F} = GF(7)$, $o = v = 3$. The central map $\mathcal{F} = (f^{(1)}, f^{(2)}, f^{(3)})$ is given by:

In[3]:=

```
mod=7;
F1[x1_,x2_,x3_,x4_,x5_,x6_]:=
4x1^2+4x1*x3+5x1*x4+6x1*x5+x1*x6+6x1+4x2^2+x2*x3+6x2*x4
+6x2*x5+5x2*x6+5x2+5x3^2+3x3*x4+5x3*x5+2x3*x6+5x3+6x4+3x5;
F2[x1_,x2_,x3_,x4_,x5_,x6_]:=
3x1*x3+4x1*x4+3x1*x5+4x1*x6+3x1+6x2^2+x2*x3+4x2*x4+4x2*x5
+5x2*x6+6x2+6x3^2+4x3*x4+2x3*x5+x3*x6+3x3+x4+x6+1;
F3[x1_,x2_,x3_,x4_,x5_,x6_]:=
6x1^2+6x1*x3+4x1*x5+2x1*x6+2x2^2+5x2*x3+6x2*x4+5x2*x5+
5x2*x6+6x2+3x3^2+5x3*x4+6x3*x5+x3*x6+3x3+4x4+6x5+5;
```

It will set up the value of *mod* to 7 and initialize functions of the central map. Next is setting up of the affine map $\mathcal{T}$ with matrix A and vector b. These two parts will be later used separately in the example.

In[7]:=
```
A=(   6 5 5 5 5 4
      6 6 4 5 0 6
      2 5 2 1 5 0
      1 1 6 2 2 3
      3 6 2 2 3 0
      0 5 4 6 1 5);
```

In[8]:=
```
b=(   1
      2
      4
      1
      3
      2);
```

In[9]:=
```
T=A.( x1
      x2
      x3
      x4
      x5
      x6)+b;
```

This block computes public key $\mathcal{P}$ by putting values of $\mathcal{T}$ inside of $\mathcal{F}$, it also simplify the expression of $p1, p2, p3$ and finally applies modulo on whole polynomial:

In[10]:=
```
p1 = F1 @@ T[[All]];
p2 = F2 @@ T[[All]];
p3 = F3 @@ T[[All]];
P1[x1_,x2_,x3_,x4_,x5_,x6_] = PolynomialMod[Simplify[p1],mod]
P2[x1_,x2_,x3_,x4_,x5_,x6_] = PolynomialMod[Simplify[p2],mod]
P3[x1_,x2_,x3_,x4_,x5_,x6_] = PolynomialMod[Simplify[p3],mod]
```

The results of $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ are:

Out[13]=
$$\{6+x1+5x2+4x1x2+2x2^2+3x3+x1x3+x2x3+x3^2+6x4+2x1x4+5x2x4+2x3x4 +3x4^2+5x5+3x1x5+6x2x5+4x3x5+3x4x5+4x5^2+4x6+x2x6+3x3x6+2x4x6\}$$

Out[14]=
$$\{5+6x1^2+5x2+4x1x2+5x2^2+4x3+5x1x3+3x2x3+2x3^2+2x4+2x1x4+4x2x4 +2x3x4+5x4^2+3x5+5x1x5+5x2x5+2x3x5+6x5^2+5x2x6+6x4x6+2x5x6+6x6^2\}$$

Out[15]=
$$\{5+5x1+4x1^2+5x2+3x1x2+5x2^2+2x3+2x1x3+x2x3+2x3^2+6x4+3x2x4+2x4^2 +x5+3x1x5+6x2x5+4x3x5+2x5^2+2x6+x1x6+3x2x6+4x3x6+6x4x6+5x5x6+4x6^2\}$$

From this place on it will only focus on computation of signature z for message w. Be aware that in this example is not used hash function for the message because for the example purpose it is not needed.

In[16]:=
```
w={{3},{6},{4}};
y1= 1;
y2= 0;
y3= 6;
```

It sets the message to value $w = (3, 6, 4)$ and also it set values for $y = (y1, y2, y3)$. These values for y are randomly chosen.

In[20]:=
```
f1 = PolynomialMod[F1[y1,y2,y3,y4,y5,y6],mod]
f2 = PolynomialMod[F2[y1,y2,y3,y4,y5,y6],mod]
f3 = PolynomialMod[F3[y1,y2,y3,y4,y5,y6],mod]
```

Here are the results after minimalizing and use of modulo:

Out[20]=
```
f1 = 6+y4+4y5+6y6
f2 = 4+y4+y5+4y6
f3 = 5+6y4+4y5+y6
```

Next two steps solve linear system $f^{(1)} = w_1 = 3, f^{(2)} = w_2 = 6, f^{(3)} = w_3 = 4$, it can also use for the solution the Gaussian elimination:

In[21]:=
```
res = Solve[{f1==w[[1]], f2==w[[2]], f3==w[[3]]},Modulus → mod];
```

In[22]:=
```
y={y1,y2,y3,y4,y5,y6} /. res
```

Out[22]=
```
{{1,0,6,6,3,0}}
```

It will obtain results for $(y4, y5, y6) = (6, 3, 0)$. After combination it is $y = (1, 0, 6, 6, 3, 0)$, so called *pre-image* of w: $y = \mathcal{F}^{-1}(w)$. If the solution for linear system do not exist, choose different values for $(y1, y2, y3)$ and repeat steps before.

Finally use $\mathcal{T}^{-1}$ to compute signature z. For that is needed inversion of matrix A.

In[23]:=
```
A_1=Inverse[A, Modulus→mod]
```

$$A_{-1} = \begin{pmatrix} 2 & 4 & 6 & 2 & 0 & 5 \\ 1 & 3 & 3 & 1 & 6 & 2 \\ 4 & 6 & 6 & 4 & 5 & 4 \\ 2 & 0 & 3 & 4 & 2 & 3 \\ 6 & 0 & 3 & 0 & 0 & 5 \\ 2 & 2 & 2 & 5 & 3 & 3 \end{pmatrix}$$

In[24]:=
```
z = Mod[A_1.(Transpose[y]-b),mod]
```

Out[24]=
```
{{4},{1},{5},{6},{3},{5}}
```

The value of the signature $z = (4, 1, 5, 6, 3, 5)$.

Last part is check if two hashes (in this example two messages w) are the same.

In[25]:=
```
w2=w;
w2={P1 @@ z[[All,1]],P2 @@ z[[All,1]],P3 @@ z[[All,1]]};
(* True? *)
Mod[w2,mod]==w
```

Out[27]=
```
True
```

The file with implementation can by find under name "UOV.nb".

2.1.2 Rainbow

2.2 Reference implementation

2.2.1 UOV

2.2.2 Rainbow

Chapter 3

Testing and discussion

On what was tested (PC and EPS32/ARM); Comparison with RSA,ECDSA; Time and memory complexity; Usability in an embedded environment;

Conclusion

How good I was...

Bibliography

[1] CZYPEK, P.: *Implementing Multivariate Quadratic Public Key Signature Schemes on Embedded Devices.* Ruhr-Universität Bochum, 2012.

[2] PETZOLDT, A.: *Multivariate Cryptography Part 1: Basics* [online]. 2017, [cit. 2020-04-1]. At: `https://2017.pqcrypto.org/school/slides/1-Basics.pdf`

[3] PETZOLDT, A.: *Multivariate Cryptography Part 2: UOV and Rainbow* [online]. 2017, [cit. 2020-04-1]. At: `https://2017.pqcrypto.org/school/slides/2-UOV+Rainbow.pdf`

[4] GEOVANDRO, C.C.F.P.: *Introduction to Multivariate Public Key Cryptography* [online]. 2013, [cit. 2020-04-1]. At: `http://www.ic.unicamp.br/ascrypto2013/slides/ascrypto2013_geovandropereira.pdf`

[5] GOUBIN, L.; PATARIN, J.; YANG, BY.: *Multivariate Cryptography.* In: van Tilborg H.C.A., Jajodia S. *Encyclopedia of Cryptography and Security.* 2011, Springer, Boston, MA

[6] DING, J.; PETZOLDT, A.: *Current State of Multivariate Cryptography.* In: *IEEE Security & Privacy.*, vol. 15, no. 4, pp. 28-36, 2017.

[7] *Multivariate cryptography* [online]. 2020, [cit. 2020-04-1]. At: `https://en.wikipedia.org/wiki/Multivariate_cryptography`

[8] KIPNIS, A.; SHAMIR, A.: *Cryptanalysis of the oil and vinegar signature scheme.* In *CRYPTO 1998*, LNCS vol. 1462, pp. 257–266, Springer, 1998.

Appendix A

Acronyms

IoT Internet of Things

MC Multivariate cryptography

MQ Multivariate quadratics

OV Oil and Vinegar

UOV Unbalanced Oil and Vinegar

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `mathematica` ..... implementation in Mathematica
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format