Bachelor's Thesis Computing Science

Radboud University Nijmegen

An investigation of the nested subset differential attack on the lifted unbalanced oil and vinegar signature scheme

Author:
Felix Mölder
s1022118

First supervisor/assessor:
Ph.D. Simona Samardjiska
simonas@cs.ru.nl

Second assessor:
Prof. Peter Schwabe
p.schwabe@cs.ru.nl

January 6, 2023

Abstract

The nested subset differential attack reduced the complexity of solving a large quadratic polynomial system in $\mathbb{F}_{2^r}$ to solving two smaller quadratic polynomial systems in $\mathbb{F}_2$. To solve these systems, exhaustive search was considered to be the best method. This thesis investigates the impact of the crossbred algorithm of Joux and Vitse on these reduced systems. Since multivariate cryptography plays a major role in the area of post-quantum cryptography, the impact of the crossbred algorithm to nested subset differential attack can confirm a new method (next to exhaustive search) to solve boolean polynomial systems and moreover, can show a new standard method to attack multivariate cryptosystems in general by reducing the systems to boolean systems and solve those systems with the crossbred algorithm. In this thesis, we mathematically analyzed the complexity of the nested subset differential attack that uses the crossbred algorithm to solve the reduced systems. Best parameters for both, theoretical examination and practical application were found and the complexity of the application of the crossbred algorithm to these reduced systems were calculated and compared. The results show that the complexity was reduced for all three parameter sets of LUOV compared to the classical attack. In the best case, a reduction of the complexity by 2^{14} was investigated.

Contents

Chapter 1

Introduction

Since Peter Shor published his paper [18] in 1994 that shows a quantum computer algorithm which breaks modern cryptosystems in polynomial time, the cryptographic world is in search for a new quantum safe standard. The National Institute of Standards and Technology (NIST) has started a new competition to find this new standard. One of these candidates is LUOV, a multivariate signature scheme and its safety is based on the NP-hardness of the MQ-problem. In round 2 of the NIST competition, an attack was found that significantly reduced the size of the polynomial equation systems. However, these systems are still quadratic and the best known method for these specific systems is nevertheless exhaustive search. Joux and Vitse described in 2018 an algorithm, the so-called crossbred algorithm, that outstands known techniques to solve boolean polynomial systems. In this thesis, we describe an analysis of the complexity to examine the impact of this algorithm and compare it to the complexity of the known attack. A fundamental understanding of LUOV and the known attacks on it will be given in Chapter 3 and Chapter 4. Several state-of-the-art methods for solving boolean quadratic systems will be compared in Chapter 5 and the complexity of the crossbred algorithm applied to the attack will be investigated in Chapter 6 to come to a final conclusion in Chapter 7.

Chapter 2

Preliminaries

2.1 Modern cryptography

There are two cryptographic methods used nowadays, namely symmetric and asymmetric cryptography.

2.1.1 Symmetric cryptography

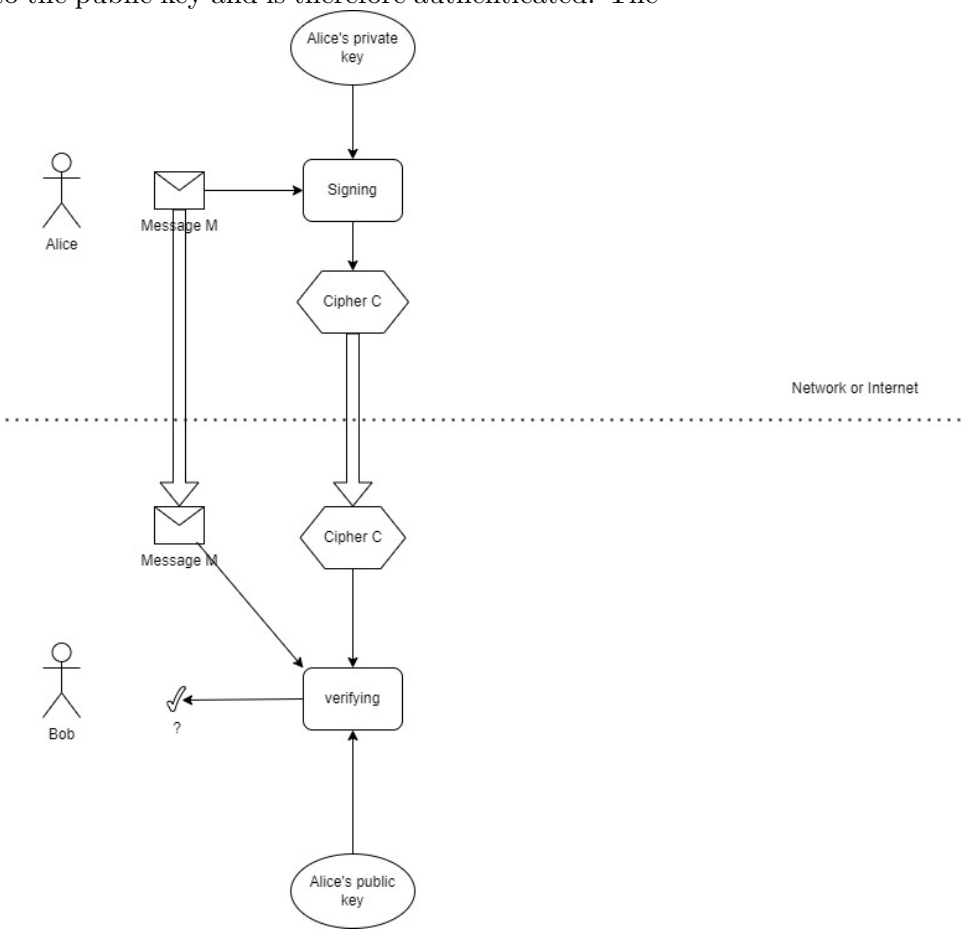
In this cryptographic method, both parties have the same key which should stay secret. This means, that Party A can encrypt the message with the secret key and send it over to Party B. Any party in between can not read the encrypted message without this secret key. However, party B can easily decrypt the message by applying the scheme together with this secret key. Therefore, the key is the same and is used for both, encryption and decryption. The NIST declared AES as the standard for symmetric cryptography in 2000.

2.1.2 Asymmetric cryptography

The security in symmetric cryptography is based on the secrecy of the key. With the rise of computer networks, the necessity to establish trustful remote connections between parties became crucial. This led to the question, how to establish a key, such that the key stays secret. This problem can be solved with asymmetric cryptography. In asymmetric cryptography, each party has a key pair consisting of a private key (Prk), which should be kept secret and a public key (Pk), which should be known to others. The public key can only be derived by the private key, but not the other way around (one-way-functions). To communicate, party A encrypt the message with the public key of party B. Party A can then send the encrypted message to party B and can be sure that only B can decrypt the message with his private key. Because it is hard to derive the private key from the public key, it is hard for someone to decrypt the message without the private key.

2.1.3 Digital Signature

With a digital signature, one can verify to others that a public key belongs to you. Assume Alice and Bob are two entities. Then Alice can prove to Bob that she has the private key belonging to the public key by signing a message using her own private key and send it to Bob together with the original message. Finally, Bob can use the public key to verify the signature and check, whether signature equals the message. If so, Bob knows that Alice has the private key related to the public key and is therefore authenticated. The



scheme work as follows:

Forging a signature

To forge a signature for a given public key $\mathscr{P}$, we want to construct a message $\boldsymbol{y}$ and signature $\boldsymbol{x}$, such that $\mathscr{P}(\boldsymbol{x}) = \boldsymbol{y}$ without knowing the private key. Therefore, we can create a signature/message pair that matches without knowing the private key. Forging of signatures might be used to let the other party think, that you are someone else. For example in the man-in-

the-middle attack.

2.2 Post-quantum cryptography (PQC)

In 1994, Peter Shor published an algorithm [18], that breaks modern cryptosystems in polynomial time using quantum computer effects. Although commercially usable quantum computers are still a long way off, intelligence agencies as well as big tech companies such as IBM or Google are interested to create a working quantum computer. Under these circumstances, the cryptographic community decided to be prepared and started to work out modern and new cryptographic schemes that are resistent against both, classical computer attacks as well as quantum computers. This new area of research is called Post-quantum cryptography (PQC).

2.3 MQ Problem

The multivariate quadratic polynomial problem, or MQ problem for short, is a mathematical problem that was proven to be NP-hard [12]. The MQ problem resides on the difficulty to find a solution for a system of multivariate quadratic equations in finite fields. This means, given a system of m quadratic polynomials with n unknown variables

$$\begin{array}{rcl} f_1(x_1,\ldots,x_n) & = & \sum_{1\leq i\leq j\leq n} \alpha_{i,j}^{(1)} x_i x_j + \sum_{1\leq i\leq n} \beta_i^{(1)} x_i + \gamma^{(1)} \\ f_2(x_1,\ldots,x_n) & = & \sum_{1\leq i\leq j\leq n} \alpha_{i,j}^{(2)} x_i x_j + \sum_{1\leq i\leq n} \beta_i^{(2)} x_i + \gamma^{(2)} \\ \vdots & = & \vdots \\ f_m(x_1,\ldots,x_n) & = & \sum_{1\leq i\leq j\leq n} \alpha_{i,j}^{(m)} x_i x_j + \sum_{1\leq i\leq n} \beta_i^{(m)} x_i + \gamma^{(m)} \end{array} \tag{2.1}$$

and values for $f_1,\ldots,f_m$, denoted $y_1,\ldots,y_m$, where $x_1,\ldots,x_n$ are n unknown variables and $\alpha_{i,j}^{(k)}$, $\beta_i^{(k)}$ and $\gamma^{(k)}$ are coefficients of some finite field $\mathbb{F}_q$, finding values for $x_1,\ldots,x_n$ such that $f_i(x_1,\ldots,x_n) = y_i$ holds for all $i \in [1,m]$ can not be done in polynomial time.

2.4 Multivariate public key cryptography (MPKC)

In post-quantum cryptography, several research areas exist. They are mainly distinguished by the mathematical problem on which their security is based. One of them is the so-called multivariate public key cryptography (MPKC), where the security is based on the NP-hardness of solving directly or indirectly the MQ problem. Therefore, MPKC schemes are asymmetric and based on multivariate polynomials over a finite field $\mathbb{F}$ (see 2.5).

2.5 Finite fields

In abstract algebra, different algebraic structures were established.

2.5.1 Groups

A group $(S\ ,\ *)$ is a set S with a binary operation $*$ such that the following properties hold [16]:

G1. $\forall x, y \in S \Rightarrow x * y \in S$ (Closure property)

G2. $\forall x, y, z \in S:\ x * (y * z) = (x * y) * z$ (associativity)

G3. $\exists\ e \in S:\ \forall x \in S : e * x = x * e = x$ (Existence of a neutral element)

G4. $\forall x \in S:\ \exists\ x^{-1} \in S:\ x * x^{-1} = x^{-1} * x = e$ (Existence of an inverse element)

If additionally $\forall x, y \in S:\ x * y = y * x$ (commutativity) holds, $(\ S\ ,\ *\ )$ is called an abelian group.
An example for an abelian group would be $(\mathbb{Z}, +)$, the set of integers together with the addition operation.

2.5.2 Rings

A ring $(S\ ,\ *\ ,\ +)$ is a set S, together with two binary operations ($*$ and $+$) such that the following properties hold [16]:

R1. $(S\ ,\ +)$ forms an abelian group

R2. $\forall x, y, z \in S:\ x * (y * z) = (x * y) * z$ (associativity of $*$)

R3. $\forall x, y, z \in S:\ (x + y) * z = xz + yz$ and $z * (x + y) = zx + zy$ (distributivity)

If additionally $\forall x, y \in S:\ x * y = y * x$ holds, $(S\ ,\ *\ ,\ +)$ is a commutative ring.
An example of a commutative ring would be $(\mathbb{Z}\ ,\ \cdot\ ,\ +)$, the set of integers together with the multiplication and addition operation.

2.5.3 Fields

A field $(S\ ,\ *\ ,\ +)$ is a set S, together with two binary operations ($*$ and $+$) such that the following properties hold:

K1. $(S\ ,\ +)$ forms an abelian group

K2. $S^* := (S \setminus \{e\}\ ,\ *)$ forms an abelian group, where e describes the neutral element in $(S\ ,\ +)$

K3. $\forall x, y, z \in S : \quad (x + y) * z = xz + yz$ and $z * (x + y) = zx + zy$ (distributivity)

Furthermore, if S has finite many elements and K1 - K3 holds, then $\mathbb{F} := (S\ ,\ *\ ,\ +)$ is called a finite Field. The smallest finite field for example is $\mathbb{F}_2 := (\{0, 1\},\ +,\ \cdot\ )$.

2.6 Extension of finite fields and lifting

Let $\mathbb{F}_p$ be a finite field as defined in 2.5.3 where p is a prime number. Then due to isomorphism there is for every positive integer n and every prime number p exactly one finite field of order p^n and is called the extension field of $\mathbb{F}_p$ or Galois field, denoted by $\mathbb{F}_{p^n}$. If $\mathbb{F}_p$ has p elements, then the finite degree n extension $\mathbb{F}_{p^n}$ has exactly p^n elements [11]. In the case of LUOV, the finite base field (field of the public key) is $\mathbb{F}_2$. Signatures and messages are created and verified over the finite extension field of degree r, namely $\mathbb{F}_{2^r}$. The isomorphism between these two fields preserves the structure of the fields (homomorphism) and this mapping is bijective (isomorphism). Mapping one polynomial with coefficients in $\mathbb{F}_2$ to a polynomial with coefficients in $\mathbb{F}_{2^r}$ in the extension field via this isomorphism is called the 'lifting' of the polynomial.

2.7 Macaulay matrix and the monomial order

Let $F = (f_1, \ldots, f_m)$ be a system of polynomials with n variables, denoted $x_1, \ldots, x_n$. The macaulay matrix of degree D of F is a matrix, that contains all monomial multiples of the elements of F such that the degree of any such product is smaller or equal to D. The columns are indexed by the monomials and the rows are indexed by the multiples. The entry $Mac_D^{i,j}(F)$ of this matrix is the coefficient of the term j in the product of i. As an example in $\mathbb{F}_2$ we consider

$$F = \{x_1x_3 + x_2,\ x_2 + x_3 + 1,\ x_1x_2 + x_2^2\} \tag{2.2}$$

Then for $D = 2$ all possible monomials in 3 variables of degree 2 or less are

$$m = (x_1^2,\ x_2^2,\ x_3^2,\ x_1x_2,\ x_1x_3,\ x_2x_3,\ x_1,\ x_2,\ x_3,\ 1) \tag{2.3}$$

and m forms the columns in the macaulay matrix. The multiples of F and monomials can have at most degree 2. Because f_0 and f_2 already have degree 2, they can only be multiplied by 1. Since f_1 has degree 1, we can multiply f_1 by $1, x_1, x_2$ and x_3 which gives

$$f_R = (f_0\ f_1,\ f_2,\ x_1f_1,\ x_2f_1,\ x_3f_1) \tag{2.4}$$

and f_R forms the rows of the macaulay matrix. Thus, the degree 2 macaulay matrix of F, namely $Mac_2(F)$, is given by

$$\begin{array}{r} \\ f_0 \\ f_1 \\ X_3 f_1 \\ X_2 f_1 \\ X_1 f_1 \\ f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} X_1^2 & X_1X_2 & X_2^2 & X_1X_3 & X_2X_3 & X_3^2 & X_1 & X_2 & X_3 & 1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{2.5}$$

The order in which the monomials are indexing the columns is called the monomial order. Lazard has shown in [15], that for any monomial order $\succ$, the rows of the reduced echelon form of the degree D macaulay matrix, whose columns are ordered by $\succ$, contains the coefficients of a Gröbner basis of the ideal I generated by F. Although any monomial order will lead to a Gröbner basis, the graded reverse lexicographical order (grevlex) seems to be the most efficient for computations and is also used in the crossbred algorithm [**?**]. The total degree of a term in a polynomial is the sum of all degrees of the unknowns. As an example, we consider $t = x_1^2 x_2 x_3^3$. Then the total degree of t is $|t| = 2 + 1 + 3 = 6$. In the grevlex order, monomials with the highest total degree are considered to be the largest. As the order is reversed, the monomials are ordered from highest to smallest total degree, so for example $x_1^2 x_2 \succ x_1 x_2 \succ x_1$. If two monomials have the same total degree, they are ordered by the lexicographical order. This means that two terms $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $x_1^{\beta_1} \cdots x_n^{\beta_n}$ with the same total degree, so $\alpha_1 + \cdots + \alpha_n = \beta_1 + \cdots + \beta_n$, are ordered such that $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \succ x_1^{\beta_1} \cdots x_n^{\beta_n}$ if $\alpha_n < \beta_n$. If $\alpha_n = \beta_n$, α_{n-1} and β_{n-1} are compared and so on. Therefore, the grevlex order of m in 2.3 is

$$x_1^2 \succ x_1x_2 \succ x_2^2 \succ x_1x_3 \succ x_2x_3 \succ x_3^2 \succ x_1 \succ x_2 \succ x_3 \succ 1. \tag{2.6}$$

2.8 s-truncation

For $1 \leq s \leq r-1$, the s-truncation (see [8]) of an element

$$a = \sum_{i=0}^{r-1} a_i t^i \tag{2.7}$$

is defined as

$$\overline{a}^s = \sum_{i=0}^{s} a_i t^i. \tag{2.8}$$

The s-truncation of a polynomial

$$f(\overline{X}) = \sum_{i=1}^{n} \sum_{j=i}^{n} a_{i,j} \overline{x_i x_j} + \sum_{i=1}^{n} b_i \overline{x_i} + c \tag{2.9}$$

is defined term by term (see (2.8)) and yields to

$$\overline{f}^s(\overline{X}) = \sum_{i=1}^{n} \sum_{j=i}^{n} \overline{a_{i,j}}^s \overline{x_i x_j} + \sum_{i=1}^{n} \overline{b_i}^s \overline{x_i} + \overline{c}^s. \tag{2.10}$$

Finally, the s-truncation of a system of polynomials

$$\mathscr{G}(\overline{X}) = (g_1(\overline{X}), g_2(\overline{X}), \ldots, g_m(\overline{X})) \tag{2.11}$$

is defined such that each polynomial is s-truncated (see (2.10)) and yields to

$$\overline{\mathscr{G}}^s(\overline{X}) = (\overline{g_1}^s(\overline{X}), \overline{g_2}^s(\overline{X}), \ldots, \overline{g_m}^s(\overline{X})). \tag{2.12}$$

Chapter 3

The oil and vinegar signature scheme family

3.1 Unbalanced oil and vinegar scheme

The unbalanced oil and vinegar scheme (UOV) was developed in 1999 by Kipnis and Patarin [13] as an adjusted version of the broken original oil and vinegar scheme (OV). In contrast to OV, the number of oil variables is not equal to the number of vinegar variables to avoid known attacks using the method of invariant subspaces [2]. In the following the signature scheme is explained.

3.1.1 Public and private key generation

The unknown variables $x_1, \ldots, x_n$ are separated in two sets, the oil variables O and the vinegar variables V such that $|O| + |V| = n$. In the unbalanced version the number of vinegar variables $v := |V|$ is a multiple of the number of oil variables $o := |O|$, usally around $v = 2o$ to $v = 3o$. Furthermore, it holds that $o = m$, so the number of oil variables is equal to the number of equations of the system. Throughout this thesis, we will stick with the notation m to denote the number of equations. Now let $\mathbb{F}_q$ be a small finite field with q elements. Then the central map $\mathscr{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with the equations $f_1, \ldots, f_m$ of the form:

$$f_k(\mathbf{x}) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{i,j,k} x_i x_j + \sum_{i=1}^{n} \beta_{i,k} x_i + \gamma_k \tag{3.1}$$

describes the multivariate quadratic system of m equations and n variables whose coefficients $\alpha_{i,j,k}$, $\beta_{i,k}$ and γ_k are elements of the finite field $\mathbb{F}_q$. Here $x_1, \ldots, x_v$ are the vinegar variables and $x_{v+1}, \ldots, x_n$ are the oil variables. Because the oil variables are never multiplied to one another, the system, if all vinegar variables are given, becomes linear and therefore solvable in

feasible time. To hide the structure of oil and vinegar variables in the system, $\mathscr{F}$ is composed with an invertible affine map $\mathscr{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ to form the public key $\mathscr{P} = \mathscr{F} \circ \mathscr{T}$. Then the private key is a pair consisting of $\mathscr{F}$ and $\mathscr{T}$ [14]. We define the following elements for UOV:

$\boldsymbol{y} \in \mathbb{F}_q^m :=$ message that will be signed

$\boldsymbol{x} \in \mathbb{F}_q^n :=$ signature that will be verified

$\mathscr{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m :=$ central map that forms the polynomial system

$\mathscr{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n :=$ invertible map to hide the structure of the variables

$\mathscr{P} = \mathscr{F} \circ \mathscr{T} :=$ public key to verify the signature

$(\mathscr{F}, \mathscr{T}) :=$ private key to sign the message

3.1.2 Create the signature

First, a preimage, say $\boldsymbol{z} \in \mathbb{F}_q^n$, for the message $\boldsymbol{y}$ under the central map $\mathscr{F}$ must be found. To find such a preimage, the Vinegar variables of $\boldsymbol{z} := (z_1, \ldots, z_v)$ are chosen randomly from $\mathbb{F}_q$. Because there are no quadratic oil values in $\mathscr{F}$, the substitution of vinegar values into this central map results into a linear system which can be solved to obtain the solution $\boldsymbol{z}$ for the message $\boldsymbol{y}$ under $\mathscr{F}$. To hide the structure of the variables in $\mathscr{F}$, the actual signature $\boldsymbol{x}$ will be obtained by computing the preimage of $\boldsymbol{z} \in \mathbb{F}_q^n$ under the invertible linear map $\mathscr{T}$, such that

$$\boldsymbol{x} = \mathscr{T}^{-1}(\boldsymbol{z}) = \mathscr{T}^{-1}(\mathscr{F}^{-1}(\boldsymbol{y})) = \mathscr{P}^{-1}(\boldsymbol{y}) \tag{3.2}$$

which gives the signature $\boldsymbol{x}$ for message $\boldsymbol{y}$ and both will be, together with the public key $\mathscr{P}$ send to the opposite.

3.1.3 Verify the signature

To verify a signature, the opposite receives the public key $\mathscr{P}$, the signature $\boldsymbol{x}$ and the message that was signed $\boldsymbol{y}$ and takes $\mathscr{P}$ which describes a system of m polynomial equations with n unknowns and assign the signature $\boldsymbol{x}$ to it, such that $\mathscr{P}(\boldsymbol{x})$. The result of this computation, say $\boldsymbol{y'} \in \mathbb{F}_q^m$, is compared to $\boldsymbol{y}$. If and only if $y_i = y_i'$ for all $i \in [0, m]$, the signature is verified.

3.2 Lifted unbalanced oil and vinegar scheme

The biggest disadvantage of UOV is the size of the public key. While the size of the public key for 128 bit security in UOV in $\mathbb{F}_{16}$ is around 585.0 kB [17], the size of the public key of a lattice-based signature scheme, called Falcon-512 needs only 897 bytes [10]. This makes UOV less preferable compared

to other signature schemes. Therefore, many variants of UOV focus mainly on the reduction of the key and signature sizes. One of these variants is the lifted unbalanced oil and vinegar signature scheme (LUOV) and was first proposed by Beullens 2017 in [21]. While the public key system in UOV has coefficients from a small finite field like $\mathbb{F}_{16}$, the key pair in LUOV is generated over the binary field $\mathbb{F}_2$, which is then lifted to a field extension of the form $\mathbb{F}_{2^r}$, where r represents the degree of the extension of $\mathbb{F}_2$. These lifted keys are used to sign and verify messages. The central map is thus defined as $\mathscr{F} : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^m$ and the variables are hided by composing $\mathscr{F}$ with $\mathscr{T} : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^n$ to generate the public key $\mathscr{P} = \mathscr{F} \circ \mathscr{T}$. The components of the central map are defined similar to the ones in UOV (see 3.1) with the difference that the coefficients $\alpha_{i,j,k}$, $\beta_{i,k}$ and γ_k are no longer chosen from a small finite field, but from $\mathbb{F}_2$, which means each coefficient can only be 0 or 1. Therefore, the size of the public key decreases significantly, even when it remains larger than the public keys of other post-quantum signature schemes [21]. In the following, creating and verifying signatures in LUOV is explained. We define the elements for LUOV as follows:

$\boldsymbol{y} \in \mathbb{F}_{2^r}^m :=$ message that will be signed

$\boldsymbol{x} \in \mathbb{F}_{2^r}^n :=$ signature that will be verified

$\mathscr{F} : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^m :=$ central map that forms the polynomial system

$\mathscr{T} : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^n :=$ invertible map to hide the structure of the variables

$\mathscr{P} = \mathscr{F} \circ \mathscr{T} :=$ public key to verify the signature

$(\mathscr{F}, \mathscr{T}) :=$ private key to sign the message

3.2.1 Create and verify a signature

Similar to UOV, the message $\boldsymbol{y}$ is signed by finding a preimage $\boldsymbol{z} \in \mathbb{F}_{2^r}^n$ under the central map $\mathscr{F}$. Again the vinegar variables are chosen randomly from $\mathbb{F}_{2^r}$ and can be solved to receive $\boldsymbol{z}$. Finally, the structure will be hidden by using the invertible linear map $\mathscr{T}$, such that

$$\boldsymbol{x} = \mathscr{T}^{-1}(\boldsymbol{z}) = \mathscr{T}^{-1}(\mathscr{F}^{-1}(\boldsymbol{y})) = \mathscr{P}^{-1}(\boldsymbol{y}) \tag{3.3}$$

where $\boldsymbol{x} \in \mathbb{F}_{2^r}^n$ is the searched signature for the massage $\boldsymbol{y}$. Likewise, the signature $\boldsymbol{x}$ is verified similar to UOV by evaluate $\mathscr{P}(\boldsymbol{x}) = \boldsymbol{y'}$. If $\boldsymbol{y'}$ is equal to $\boldsymbol{y}$, then the signature is accepted. If not, then the signature should be rejected [21].

Chapter 4

The differential attack family

4.1 The subfield differential attack

The subfield differential attack (SDA) uses the structure of finite field extensions. Finite field theory shows that $\mathbb{F}_2$ is not the only subfield embedded in the field extension $\mathbb{F}_{2^r}$. Hence, if d is any positive divisor of r with $d \cdot s = r$, we can construct an isomorphism

$$\mathbb{F}_{2^d}/(f(t)) \cong \mathbb{F}_{2^r} \tag{4.1}$$

where $f(t)$ is an irreducible polynomial of degree $s = \frac{r}{d}$ and $\mathbb{F}_{2^d}$ is a smaller subfield embedded in the field extension $\mathbb{F}_{2^r}$ [16].
To use this isomorphism we consider a differential. Consider $\mathscr{P} : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^m$ as a quadratic multivariate polynomial system with coefficients from $\mathbb{F}_2$ and $\boldsymbol{y}$ a message arbitrary chosen from $\mathbb{F}_{2^r}^m$. We define
$\overline{\mathscr{P}} : \mathbb{F}_{2^d}^n \to \mathbb{F}_{2^r}^m$ by $\overline{\mathscr{P}}(\overline{\boldsymbol{x}}) = \mathscr{P}(\boldsymbol{x'} + \overline{\boldsymbol{x}})$ where $\boldsymbol{x'} \in \mathbb{F}_{2^r}^n$ is a randomly chosen point and $\overline{\boldsymbol{x}} \in \mathbb{F}_{2^d}^n$ an indeterminate point. If we randomly chose $\boldsymbol{x'}$, the kth component of $\overline{\mathscr{P}}$ is of the form:

$$\tilde{f}_k(\boldsymbol{x'} + \overline{\boldsymbol{x}}) = \sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j,k}(x'_i + \overline{x}_i)(x'_j + \overline{x}_j) + \sum_{i=1}^{n} \beta_{i,k}(x'_i + \overline{x}_i) + \gamma_k \tag{4.2}$$

and expanding the brackets and separate the quadratic term yields to

$$\begin{aligned} \tilde{f}_k(\boldsymbol{x'} + \overline{\boldsymbol{x}}) = & \sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j,k}(x'_i x'_j + x'_i \overline{x}_i + x'_j \overline{x}_j) + \sum_{i=1}^{n} \beta_{i,k}(x'_i + \overline{x}_i) \\ & + \gamma_k + \sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j,k} \overline{x}_i \overline{x}_j. \end{aligned} \tag{4.3}$$

Because $\alpha_{i,j,k} \in \mathbb{F}_2$, the coefficients of the quadratic terms $(\overline{x}_i \overline{x}_j)$ will be all in $\mathbb{F}_2$. On the other side, all x'_i are in $\mathbb{F}_{2^r}$ and therefore the coefficients of

the linear $\overline{x}_i$ terms will have all powers of t up until $s-1$. Based on this fact, we can group the terms by their powers of t such that the kth equation can be written as:

$$\tilde{f}_k(\boldsymbol{x}' + \overline{\boldsymbol{x}}) = Q_k(\overline{x}_1, \ldots, \overline{x}_n) + \sum_{i=1}^{s-1} L_{i,k}(\overline{x}_1, \ldots, \overline{x}_n)t^i \tag{4.4}$$

where $Q_k(\overline{\boldsymbol{x}}) \in \mathbb{F}_{2^d}[\overline{x}_1, \ldots, \overline{x}_n]$ are the quadratic polynomials and $L_{i,k}(\overline{\boldsymbol{x}}) \in \mathbb{F}_{2^d}[\overline{x}_1, \ldots, \overline{x}_n]$ are linear polynomials grouped by the power of t. To forge a signature $\boldsymbol{x} \in \mathbb{F}_{2^r}^n$, we decompose an arbitrary message $\boldsymbol{y} \in \mathbb{F}_{2^r}^m$ into a sum of vectors such that:

$$\boldsymbol{y} = Y_0 + Y_1 t + \cdots + Y_{s-1}t^{s-1} \tag{4.5}$$

where for each i, $Y_i = (y_{i,1}, \ldots, y_{i,m}) \in \mathbb{F}_{2^d}^m$, i.e.:

$$\boldsymbol{y} = \begin{bmatrix} y_{0,1} \\ \vdots \\ y_{0,m} \end{bmatrix} + \begin{bmatrix} y_{1,1} \\ \vdots \\ y_{1,m} \end{bmatrix} \cdot t + \cdots + \begin{bmatrix} y_{s-1,1} \\ \vdots \\ y_{s-1,m} \end{bmatrix} \cdot t^{s-1} \tag{4.6}$$

Combining the equations 4.4 and 4.6, we first need to find the solution space S of the system of $(s-1) \cdot m$ linear equations

$$A = \{L_{i,j}(\overline{\boldsymbol{x}}) = y_{i,j} : 1 \leq i \leq s-1, 1 \leq j \leq m\} \tag{4.7}$$

We see that A describes a linear map with $A : \mathbb{F}_{2^d}^n \to \mathbb{F}_{2^d}^{((s-1)\cdot m)}$. A solution of this linear equation system is any $\boldsymbol{x}$ that satisfies

$$A \cdot \boldsymbol{x} = \boldsymbol{0} \tag{4.8}$$

Therefore, the solution space S is the nullspace (or kernel) of A. Therefore, the dimension of the solution space S is the dimension of the kernel of A, hence $\dim(S) = \dim(\ker(A))$. Furthermore, the dimension of the domain of A is $\dim(\mathbb{F}_{2^d}^n) = n$ and since A is actually a random system of linear equations, there is a high probability [7] to be of full rank: $\min\{(s-1)m,\ n\}$. Now by the rank-nullity theorem [3], which states that for any $m \times n$ matrix M, it holds that rank(M) + nullspace(M) = $\dim(domain(M)$, we know that:

$$\begin{aligned} \dim(domain(A)) &= rank(A) + \dim(\ker(A)) \\ n &= \min\{(s-1)\cdot m,\ n\} + \dim(S) \\ \dim(S) &= n - \min\{(s-1)\cdot m,\ n\} \\ &= \max\{n - (s-1)\cdot m,\ 0\} \end{aligned}$$

Now, since we know the dimension of S and finding a solution to a linear system is easy, we see that our problem reduces to solving the set of m quadratic equations

$$B = \{Q_i(\overline{\boldsymbol{x}}) = y_{0,i} :\ 1 \leq i \leq m,\ \overline{\boldsymbol{x}} \in S\} \tag{4.9}$$

over S. If S is now of large enough dimension (depending on the choice of d, n and m), the solution $\overline{\boldsymbol{x}}$ for B shows that the signature we searched is $\boldsymbol{x}' + \overline{\boldsymbol{x}}$ as

$$\mathscr{P}(\boldsymbol{x}' + \overline{\boldsymbol{x}}) = \overline{\mathscr{P}}(\overline{\boldsymbol{x}}) = \boldsymbol{y} \tag{4.10}$$

and thus, the signature-message-pair $(\boldsymbol{x}' + \overline{\boldsymbol{x}},\ \boldsymbol{y})$ will be a forged pair that will be verified without knowing the private key. Since the complexity is reduced from m equations in n variables over the field $\mathbb{F}_{2^r}$ to m equations in $n - (s-1) \cdot m$ variables over the subfield $\mathbb{F}_{2^d}$, SDA broke LUOV [14].

4.2 The nested subset differential attack

In response to the subfield differential attack, the developers of LUOV adjusted the parameters, such that the degree of the extension (r) is a prime number. This ensures that the only subfield existing, will be the prime field $\mathbb{F}_{2^1} = \mathbb{F}_2$ due to the fact that 1 and the prime number itself are the only divisors of r. This led to the following new parameters [20]:

Name	Security level	(r, m, v, n)
LUOV-7-57-197	I	(7, 57, 197, 254)
LUOV-7-83-283	III	(7, 83, 283, 366)
LUOV-7-110-374	V	(7, 110, 374, 484)
LUOV-47-42-182	I	(47, 42, 182, 224)
LUOV-61-60-261	III	(61, 60, 261, 321)
LUOV-79-76-341	V	(79, 76, 341, 417)

In fact, the probability that for a signature in the domain exists is $e^{-\frac{|A|}{|B|}} = 1 - e^{-((dn)-(rm))}$, with A the domain and B the range of the polynomial system $\mathscr{P}$ [14]. Thus, applying SDA to the new parameters gives the following probabilities:

Name	(r, m, v, n)	Probability
LUOV-7-57-197	(7, 57, 197, 254)	$1 - e^{(-2^{-145})}$
LUOV-7-83-283	(7, 83, 283, 366)	$1 - e^{(-2^{-215})}$
LUOV-7-110-374	(7, 110, 374, 484)	$1 - e^{(-2^{-286})}$
LUOV-47-42-182	(47, 42, 182, 224)	$1 - e^{(-2^{-1750})}$
LUOV-61-60-261	(61, 60, 261, 321)	$1 - e^{(-2^{-3339})}$
LUOV-79-76-341	(79, 76, 341, 417)	$1 - e^{(-2^{-5587})}$

Since all probabilities are close to 0, there is no signature for a particular message and SDA cannot be applied to these new parameters. In [8], *Ding et al.* showed a modification of the SDA, called the nested subset differential

attack (NSDA) that makes the forge of a signature possible for the first three new parameter sets, where $r = 7$. Let $P : \mathbb{F}_{2^r}^n \to \mathbb{F}_{2^r}^m$ be the public key with $r = 7$ and assume we want to forge a signature $\boldsymbol{x} \in \mathbb{F}_{2^r}^n$ for a message $\boldsymbol{y} \in \mathbb{F}_{2^r}^m$. We will denote by $\overline{\boldsymbol{x}} = (\overline{x_1}, \ldots, \overline{x_n}) \in \mathbb{F}_2^n$ an indeterminate point. We decompose $\boldsymbol{y} = Y_0 + Y_1 t + \cdots + Y_{r-1} t^{r-1}$ where $Y_i = (y_{i,1}, \ldots, y_{i,m}) \in \mathbb{F}_2^m$ for all i, thus

$$\boldsymbol{y} = \begin{bmatrix} y_{0,1} \\ \vdots \\ y_{0,m} \end{bmatrix} + \begin{bmatrix} y_{1,1} \\ \vdots \\ y_{1,m} \end{bmatrix} \cdot t + \cdots + \begin{bmatrix} y_{r-1,1} \\ \vdots \\ y_{r-1,m} \end{bmatrix} \cdot t^{r-1} \tag{4.11}$$

Finally, we denote the set of all polynomials in $\mathbb{F}_2[t]/(g(t))$ which are truncated to the third power by

$$E := \{\overline{a}^3 : a \in \mathbb{F}_{2^r}\} \tag{4.12}$$

The probabilities that there will be a signature for Y if we consider E^n with a size of 2^{4n} possible elements, as the domain and recalculate the probabilities. This probabilities can be found in [8] and are all close to 1, so it is very likely that we only need to consider signatures from E^n in this attack. We can construct a signature step by step using differentials instead of looking over all elements of E^n at once. For each step, we add an additional power of t in our signature $\overline{\mathscr{P}}^h(A_0 + A_1 t + \cdots + \overline{\boldsymbol{x}} t^h)$ with $0 \leq h \leq 3$, consider the solution spaces and use the result in the next step. This is possible due to the special construction of lifted polynomials given by the following lemma

Lemma 2. *Let*

$$\tilde{f}(X) = \sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j} x_i x_j + \sum_{i=1}^{n} \beta_i x_i + \gamma \tag{4.13}$$

be a lifted polynomial and $A_0, A_1, \ldots, A_{l-1} \in \mathbb{F}_2^n$ *with*

$$A_i = (a_{i,1}, \ldots, a_{i,n}). \tag{4.14}$$

Set $\boldsymbol{A} = A_0 + A_1 t + A_2 t^2 + \cdots + A_{l-1} t^{l-1}$. *We have that for* $\tilde{f}(\boldsymbol{A} + X t^l)$ *all the quadratic terms are coefficients of* t^{2l}, *the linear terms are coefficients of* $t^l, t^{l+1}, \ldots, t^{2l-1}$, *and the coefficients of* t^h *depends only on* $\alpha_{i,j}, \beta_i$, *and* A_k *for* $k \leq h$ *and* X *for* $h \geq l$.

The proof of this lemma is given in [8]. To keep this attack efficient, we ensure to always solve no more than m quadratic equations over $\mathbb{F}_2$ with at least as many variables as equations and we achieve this in four steps using Lemma 2.

Step 1:

If we define $\overline{\mathscr{P}}^s(\overline{\boldsymbol{x}})$ to be the s-truncation of $\mathscr{P}(\overline{\boldsymbol{x}})$ then we see that

$$\overline{\mathscr{P}}^0(\overline{\boldsymbol{x}}) = \begin{cases} Q_{0,1}(\overline{\boldsymbol{x}}) \\ Q_{0,2}(\overline{\boldsymbol{x}}) \\ \quad\vdots \\ Q_{0,m}(\overline{\boldsymbol{x}}) \end{cases} \tag{4.15}$$

Here, each $Q_{0,i}(\boldsymbol{x})$ is a quadratic polynomial over $\mathbb{F}_2$ (boolean). We solve $\overline{\mathscr{P}}^0(\overline{\boldsymbol{x}}) = Y_0$, a system of m equations in n variables, using a direct attack method like exhaustive search and call the found solution A_0.

Step 2:

We can use our solution of Step 1 to construct

$$\overline{\mathscr{P}}^1(A_0 + \overline{\boldsymbol{x}}t) = \begin{cases} y_{0,1} + L_{1,1}(\overline{\boldsymbol{x}})t \\ y_{0,2} + L_{1,2}(\overline{\boldsymbol{x}})t \\ \quad\vdots \\ y_{0,m} + L_{1,m}(\overline{\boldsymbol{x}})t \end{cases} \tag{4.16}$$

where each $L_{1,i}(\overline{\boldsymbol{x}})$ is, due to Lemma 2, a boolean linear polynomial. We can solve this system by finding a solution, say A_1, for the system of linear equations

$$\{L_{1,i}(\overline{\boldsymbol{x}}) = y_{1,i} : 1 \leq i \leq m\} \tag{4.17}$$

and then have $\overline{\mathscr{P}}^1(A_0 + A_1 t) = Y_0 + Y_1 t$.

Step 3:

We use A_0 and A_1 to examine $\overline{\mathscr{P}}^2(A_0 + A_1 t + \overline{\boldsymbol{x}}t^2)$. The s-truncation makes this a system of polynomials of degree 2 in t and Lemma 2 of [8] states that the coefficients of the t^2 terms will be linear, the coefficients of the constant terms will only depend on A_0 and the coefficients of the t will depend only on A_0 and A_1, so we get

$$\overline{\mathscr{P}}^2(A_0 + A_1 t + \overline{\boldsymbol{x}}t^2) = \begin{cases} y_{0,1} + y_{1,1}t + L_{2,1}(\overline{\boldsymbol{x}})t^2 \\ y_{0,2} + y_{1,2}t + L_{2,2}(\overline{\boldsymbol{x}})t^2 \\ \quad\vdots \\ y_{0,m} + y_{1,m}t + L_{2,m}(\overline{\boldsymbol{x}})t^2 \end{cases} \tag{4.18}$$

with each $L_{2,i}(\overline{\boldsymbol{x}})$ a linear boolean polynomial. Therefore if we find a solution, say A_2, to the system of linear equations

$$\{L_{2,i}(\overline{\boldsymbol{x}}) = y_{2,i} : 1 \leq i \leq m\} \tag{4.19}$$

then we have $\overline{\mathscr{P}}^2(A_0 + A_1 t + A_2 t^2) = Y_0 + Y_1 t + Y_2 t^2$.

Step 4:

In the final step we drop the need for s-truncation and examine $\mathscr{P}^3(A_0 + A_1 t + A_2 t^2 + \overline{\boldsymbol{x}} t^3)$. Since there is no truncation, the system of polynomials will have degree $r-1$ in t. Thus, the first 3 new parameter sets of LUOV ($r = 7$) will have degree 6 in t, the highest degree for polynomials in $\mathbb{F}_2[t]/(g(t))$. Due to Lemma 2, we know that only the coefficients of the t^6 terms are quadratic and the coefficients of the t^3, t^4 and t^5 terms are linear in $\overline{\boldsymbol{x}}$. Furthermore, the coefficients of the constant terms only depend on A_0, the coefficients of t only on A_0 and A_1 and the coefficients of t^2 on A_0, A_1 and A_2. By construction of A_0, A_1 and A_2 we have

$$\overline{\mathscr{P}}^3(A_0+A_1t+A_2t^2+\overline{\boldsymbol{x}}t^3) = \begin{cases} y_{0,1} + y_{1,1}t + y_{2,1}t^2 + L_{3,1}(\overline{\boldsymbol{x}})t^3 + L_{4,1}(\overline{\boldsymbol{x}})t^4 \\ \quad +L_{5,1}(\overline{\boldsymbol{x}})t^5 + Q_{6,1}(\overline{\boldsymbol{x}})t^6 \\ y_{0,2} + y_{1,2}t + y_{2,2}t^2 + L_{3,2}(\overline{\boldsymbol{x}})t^3 + L_{4,2}(\overline{\boldsymbol{x}})t^4 \\ \quad +L_{5,2}(\overline{\boldsymbol{x}})t^5 + Q_{6,2}(\overline{\boldsymbol{x}})t^6 \\ \quad \vdots \\ y_{0,m} + y_{1,m}t + y_{2,m}t^2 + L_{3,m}(\overline{\boldsymbol{x}})t^3 + L_{4,m}(\overline{\boldsymbol{x}})t^4 \\ \quad +L_{5,m}(\overline{\boldsymbol{x}})t^5 + Q_{6,m}(\overline{\boldsymbol{x}})t^6 \end{cases}$$

We can now proceed similar to the last step in SDA. We first find the solution space S for the system of linear equations

$$A = \{L_{i,j}(\overline{\boldsymbol{x}}) = y_{i,j} : 3 \leq i \leq 5, 1 \leq j \leq m\} \tag{4.20}$$

and because A has a high probability to be full rank $3m$, the dimension of S will be $n - 3m$. Thus, the system of quadratic equations

$$B = \{Q_{6,j}(\overline{\boldsymbol{x}}) = y_{6,j} : 1 \leq j \leq m, \overline{\boldsymbol{x}} \in S\} \tag{4.21}$$

has a high probability to have a solution and therefore finding a solution to B, say A_3 we have

$$\mathscr{P}(A_0 + A_1 t + A_2 t^2 + A_3 t^3) = \boldsymbol{y}. \tag{4.22}$$

Hence, $\boldsymbol{x} = A_0 + A_1 t + A_2 t^2 + A_3 t^3$ is a forged signature for the message $\boldsymbol{y}$. This attack is only possible for small values of r, otherwise the number of

linear equations to be solved besides the final quadratic system also increases so much, that it is highly unlikely that a final solution exist.
The complexity of NSDA is largely reduced to solve the two quadratic systems of m equations over $\mathbb{F}_2$, because the overhead from solving the linear systems can be ignored as the size is never much larger than the quadratic systems and are much more efficient to solve. Due to the small field size and the limited number of variables, *Ding et al.* used exhaustive search to solve these two quadratic systems. For determined systems ($n = m$), [6] estimates that the number of bit operations for finding all the solutions would be $\log_2(n)2^{n+2}$. Since only one solution is necessary, values can be randomly assigned until the system is either determined or lightly underdetermined ($n > m$). *Ding et al.* decided to guess all but $m + 2$ of the variables to assure a solution on the first try. Thus, the complexity is reduces to solve a system of m equations with $m + 2$ variables in $\mathbb{F}_2$ and yields to 2 times $\log_2(m + 2)2^{m+4}$ [8].

4.3 Effects and security consequences

The NIST presumed 6 different security level. Three levels (I,III and V) are equivalent to the symmetric standard AES and three levels (II, IV and VI) are equivalent to the latest hash standard SHA3. For each of them, the NIST gave the number of classical gate counts for the optimal key recoveries (collision attacks) on AES (SHA3). The security of these levels are:

Security level	Equivalence	security in $\log_2$
I	AES-128	143
III	AES-192	207
V	AES-256	272
II	SHA3-256	146
IV	SHA3-384	210
VI	SHA3-512	274

Applying SDA on the old parameters of LUOV leads to the following complexities given in $\log_2$:

Parameter set	NIST security level	security (SDA)
LUOV-8-58-237	II	107
LUOV-8-82-323	IV	146
LUOV-8-107-371	VI	184
LUOV-48-43-222	II	135
LUOV-64-61-302	IV	202
LUOV-80-76-363	VI	244

Applying NSDA on the new parameters of LUOV lead to the following complexities given in $\log_2$:

Parameter set	NIST security level	security (NSDA)
LUOV-7-57-197	I	61
LUOV-7-83-283	III	89
LUOV-7-110-374	V	116

It can be seen that due to the SDA none of the security levels required for the NIST competition can be hold. Furthermore, due to the new parameters and the NSDA, not one of the first three parameter sets satisfies the lowest NIST security level I. This makes LUOV no longer acceptable for the NIST standard competition and both attacks are the main reason for the elimination of LUOV after round 2.

Chapter 5

Solving polynomial systems in $\mathbb{F}_2$

The efficiency of solving quadratic polynomial systems with coefficients in $\mathbb{F}_2$ highly depends on the number of equations m and the number of unknowns n. For extremely overdetermined ($m > n(n+1)/2$) and extremely underdetermined ($n > m(m+1)$) random polynomial equation systems, methods are known to find a solution in polynomial time [13]. Between these two extremes, the determined case ($n = m$) is the hardest, since for $m > n$ the additional information of the extra equations will simplify the problem and for $m < n$ we can always assign random values to $n-m$ unknowns and make the system determined.

5.1 Known attacks and previous methods

5.1.1 Exhaustive search

This attack can be applied to any cryptosystem and in general its complexity forms the upper bound of the complexity, because the attacker tries any possible key until the correct one is found. The length of the key and the size of the set of possible keys are crucial parameters to determine its complexity. For the public key $\mathscr{P}$ describing a random boolean quadratic polynomial system with n unknown variables and m equations takes therefore $m \cdot 2^n$ evaluations. Thus, $m \cdot 2^n$ forms the upper bound of the complexity of the system. Using Gray codes and other techniques, this complexity can be reduced to $\mathcal{O}(\log_2(n)2^n)$ operations [6].

5.1.2 Algebraic methods

The idea of algebraic methods is to consider the polynomial equation systems by looking at the ideals they generate and solve them by finding a good representation of the corresponding ideal. Assume $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a

family of elements in a multivariate polynomial ring $K[X_1, \ldots, X_n]$, then $\mathcal{F}$ generates the ideal I which contains the following set of polynomials [1]:

$$I = \left\{ \sum_{i=1}^{m} p_i f_i \;\middle|\; (p1, \ldots, p_m) \in K[X_1, \ldots, X_n]^m \right\} \tag{5.1}$$

With the Gröbner basis of this ideal, solutions can be easily computed by successively eliminating the variables, which means computing solutions of univariate polynomials and use the results to reduce the number of unknowns in other polynomials [5]. Lazard has shown in [15] that we can compute the Gröbner basis of I, by constructing the Macaulay matrix, whose columns are sorted according to any monomial order $\succ$. Then, the rows the reduced row echelon form of this matrix contains the coefficients of a Gröbner basis of I. Various algorithms such as F_4, F_5 or XL use this algebraic method to solve systems of polynomial equations and in general, solving an overdetermined system of quadratic polynomials in $\mathbb{F}_2$ has the following complexity:

$$\mathcal{O}\left(\binom{n}{D_{reg}}^{\omega} \right) \tag{5.2}$$

where D_{reg} is the degree of regularity [4] and ω is the exponent of matrix multiplication. The smallest value known is $\omega = 2.373$ but in practice $\omega = 2.807$ by using the Strassen algorithm.

FXL/BooleanSolve

The FXL/BooleanSolve algorithms combine exhaustive search with algebraic methods on Macaulay matrices. In the first step, the algorithm uses exhaustive search by picking randomly an $\boldsymbol{a} = (a_{k+1}, \ldots, a_n) \in \mathbb{F}_2^{n-k}$ and specializes partly the polynomials $f_{1,a}, \ldots, f_{m,a}$ of the boolean system such that the resulting boolean system has k unknown variables and m equations. In the second step, the Macaulay matrix of this system in degree D_{reg} is used to check if this system admits a solution. If it does, the algorithm uses exhaustive search on $x_1, \ldots, x_k$ to find the solution. If not, the first step is repeated with another $\boldsymbol{a}$.

5.2 The crossbred algorithm of Joux/Vitse

The new crossbred algorithm is based on this principle, but unlike the FXL/-BooleanSolve algorithms, it performs the specialization step on $n - k$ variables after using the macaulay matrix and the Gröbner basis to decide if a system admits a solution. Hence, we construct a degree D macaulay matrix, sort the monomials (columns) in grevlex order and compute the row echelon form. Therefore, we can generate degree D equations with k variables eliminated. Finally, we try to solve the resulting system with exhaustive search on $n - k$ variables. We define the necessary parameters as follows [9]:

$F :=$ a system of m equations over n variables in $\mathbb{F}_2$

$D :=$ the degree of the macaulay matrix of F

$d :=$ the degree of the equations we want our specialized system to have, after the last $n-k$ variables are specialized.

$k :=$ the number of variables we want our specialized system to have

$M^{(k)}_{D,d}(\mathcal{F}) :=$ the submatrix of $Mac^{(k)}_{D,d}(\mathcal{F})$ whose columns correspond

The are two phases, the preparation phase and the algorithm phase. The preparation phase goes as following:

1. Construct the macaulay matrix of degree D of the polynomial system F, where the columns are sorted in grevlex order, denoted $Mac_D(F)$.
2. Let $Mac^k_{D,d}(F)$ be the submatrix of $Mac_D(F)$, where each row $u_{i,j}f_i$ represents a polynomial with $deg_k(u_{i,j}) \geq d-1$.
3. Let $M^k_{D,d}(F)$ be a submatrix of $Mac^k_{D,d}$, where each column i represents the monomial M_i with $deg_k(M_i) > d$.

After all necessary matrices have been constructed, the algorithm phase goes as following:

1. Construct the kernel of $M^k_{D,d}$ and multiply it by $Mac^k_{D,d}(F)$. The result forms a system of polynomial equations, P, where each has a total degree $\leq D$ and at most d in $x_1, \ldots, x_k$.
2. $\forall a = \{a_{k+1}, \ldots, a_n\} \in \mathbb{F}_2^{n-k}$:
   a) Partially evaluate the last $n-k$ variables of F in a, so $\forall i$: $f_i(X_1, \ldots, X_k, a_{k+1}, \ldots, a_n)$ and denote the resulting system F^*.
   b) Construct the macaulay matrix of degree d of F, denoted $Mac_d(F^*)$.
   c) Similar to a), evaluate the last $n-k$ variables of P in a and let P^* represent this new system as a coefficient matrix.
   d) Append $Mac_d(F^*)$ to P^* and let PM^* represent the resulting system.
   e) Check if PM^* is solvable in $x_1, \ldots, x_k$ and if so, extract the variables $x_1, \ldots, x_k$ and test the solution.

The complexity of this algorithm is determined step 1 and step 2 of the algorithm phase. In step 1 the main computation is necessary to compute the kernel vectors of $M^k_{D,d}$, a sparse matrix. This complexity is [9]

$$C_{Ker} = \mathcal{O}(n_{cols}) + \mathcal{O}(n^2_{cols} \log n_{cols} \log\log n_{cols}) \approx \mathcal{O}(n^2_{cols}) \tag{5.3}$$

where $n_{col} :=$ the number of columns of the matrix. This number of columns of $M^k_{D,d}(F)$, say $N^k_{D,d}$, corresponds to the number of monomials labeling its columns in $\mathbb{F}_2$ is given by [1]:

$$N^k_{D,d} = \sum_{d_k=d+1}^{D} \sum_{d'=0}^{D-d_k} \binom{k}{d_k} \binom{n-k}{d'} \tag{5.4}$$

In the second step, we guess for $n-k$ variables ($\mathcal{O}(2^{n-k})$). For each of these guesses, we use linearization to find a solution of PM^* with the complexity [9]

$$\mathcal{O}\left(\left(\sum_{i=0}^{d} \binom{k}{i}\right)^{\omega}\right) \tag{5.5}$$

Therefore, the overall complexity of the crossbred algorithm in $\mathbb{F}_2$ is given by [9]:

$$C_{cross} = \mathcal{O}\left(\left(\sum_{d_k=d+1}^{D} \sum_{d'=0}^{D-d_k} \binom{k}{d_k} \binom{n-k}{d'}\right)^2\right) + 2^{n-k} \cdot \mathcal{O}\left(\left(\sum_{i=0}^{d} \binom{k}{i}\right)^{\omega}\right) \tag{5.6}$$

Chapter 6

Research

6.1 Comparison of different algorithms

As we have seen in Chapter 4, the NSDA ensures that we only need to solve two systems of quadratic equations of smaller size in $\mathbb{F}_2$. As these systems are underdetermined and the method of Thomae and Wolf [19] can not be applied here, we guess all but $m+2$ variables to ensure that a solution can be found in the first try and lead to the following system sizes:

Parameter set	(m, n) before NSDA	(m, n) after NSDA
LUOV-7-57-197	$(57, 254)$	$(57, 59)$
LUOV-7-83-283	$(83, 366)$	$(83, 85)$
LUOV-7-110-374	$(110, 484)$	$(110, 112)$

Therefore, we need to solve nearly determined systems in $\mathbb{F}_2$. To solve these systems several algorithmic methods have been shown in Chapter 5. Ding et al. decided in [8] to use exhaustive search, because of the small field size. This is the case because [6] expected that for determined systems, pre-existing algebraic methods can not beat fast exhaustive search for n lower than 200. However, Joux & Vitse demonstrated in [1] that the crossover point between fast exhaustive search and the crossbred algorithm is $n = 37$ even for small fields like $\mathbb{F}_2$. Moreover, the FXL/BooleanSolve algorithm first guesses $n-k$ variables ($\mathcal{O}(2^{n-k})$) and for every guess, linear algebra on the macaulay matrix is performed, which makes this repeating step costly. The crossbred algorithm circumvents this problem by first constructing the macaulay matrix of degree D in grevlex order and then generate new equations that are linear in $X_1, \ldots, X_k$ such that if then $X_{k+1}, \ldots, X_n$ are specified (as in FXL), the leftover system is linear and therefore easier solvable. Also in [1] it was shown that for determined systems, the crossbred algorithm outstands the FXL/BooleanSolve algorithm for any number of variables although the complexity exponents move closer together as n becomes larger.

This is why it is worthwhile to examine the crossbred algorithm as a new approach to solve the reduced systems of NSDA.

6.2 Determine the parameters D, d and k

As described in Chapter 5, the performance of the crossbred algorithm highly depends on the choice of the three parameters D, d and k. Finding the best or even admissible values for D, d and k is far from trivial. To determine the admissibility of these three parameters, Joux and Vitse derived a bivariate generating function. They first defined

$$S_{D,d}^{k} = \frac{(1+X)^{n-k}}{(1-X)(1-Y)} \left(\frac{(1+XY)^k}{(1+X^2Y^2)^m} - \frac{(1+X)^k}{(1+X^2)^m} \right) \tag{6.1}$$

where the coefficient of $X^D Y^d$ of $S_{D,d}^{K}$ represents the number of new independent polynomials after the reduction of $Mac_{D,d}^{k}$, which is equivalent to the left kernel of $M_{D,d}^{k}$ (see Step 1 of the algorithm phase). However, some of these new independent polynomials reduce to 0 after evaluation and therefore do not yield any new information. The number of those polynomials is generated by [9]

$$\frac{(1+Y)^k}{(1-X)(1-Y)(1+Y^2)^m} \tag{6.2}$$

This means, if we compute

$$S_{D,d}^{k} - \frac{(1+Y)^k}{(1-X)(1-Y)(1+Y^2)^m} \tag{6.3}$$

then the coefficient of $X^D Y^d$ describes the amount of new independent polynomials that do not reduce to 0 and thus, if the coefficient is non-negative the values for D, d and k are admissible. To determine the parameters for the reduced NSDA systems, we wrote a script A.1 that iterates through all possible values for these three parameters and and checks wether the combination is admissible. If this is the case, the complexity for these parameters is computed using the formula given in [9]. The best complexity in $\log_2$ is printed together with the best parameters for D, d and k.

6.3 Compute complexity for NSDA systems

In [8], the first three new parameter sets of LUOV were reduced:

Parameter set	Original system over $\mathbb{F}_{2^7}$	New system over $\mathbb{F}_2$
LUOV-7-57-197	$m = 57, n = 254$	$m = 57, n = 59$
LUOV-7-83-283	$m = 83, n = 366$	$m = 83, n = 85$
LUOV-7-110-374	$m = 110, n = 484$	$m = 110, n = 112$

By using the script in A.1, we obtain the following parameters for the reduced systems and the complexity:

Parameter set	(m, n)	D	d	k	$\log_2$ complexity
LUOV-7-57-197	$(57, 59)$	5	1	14	55
LUOV-7-83-283	$(83, 85)$	8	1	20	77
LUOV-7-110-374	$(110, 112)$	9	1	22	102

These complexities are theoretically computed and the complexity is the best possible based on the best values for $2 \leq D \leq \sqrt{n}$, $1 \leq d < D$ and $1 \leq k \leq n$. However, since D describes the degree of the macaulay matrix, $D \geq 5$ makes the saving of the macaulay matrix not feasible anymore on normal computer. To see this, we consider the first parameter set of the table above. This gives $m = 57$, $n = 59$, $D = 5$, $d = 1$ and $k = 14$. Now we can use the formula 5.4 to compute the number of columns of the macaulay matrix. This yields $N_{5,1}^{14} = 1810718$ and looking at Table 1 in [1] shows that the number of rows will be around the same. Therefore, the macaulay matrix will have $1810718^2 = 3.278699675524 \times 10^{12}$ entries. Assuming that each entry needs to be represented by one bit, the matrix will take around 410 GB. The same computation for the last row of table 1 in [1] ($N_{4,1}^{23} = 277288, R_{4,1} = 278166$), will take around 10 GB. As 10 GB in memory is already heavy to work with, 410 GB are not possible except for a small group of specified computers that cost a lot. Thus, we decided to provide the theoretical values above and furthermore rerun the script with $D = \{3, 4\}$ and provide values for parameters and an estimation of the complexity that might be used in a practical environment. This rerun yields

Parameter set	(m, n)	D	d	k	$\log_2$ complexity
LUOV-7-57-197	$(57, 59)$	4	1	13	56
LUOV-7-83-283	$(83, 85)$	3	1	14	81
LUOV-7-110-374	$(110, 112)$	4	1	17	106

To summarize, the following table shows the required complexity of the NIST Post-quantum cryptography competition, the claimed complexity of the LUOV developers for direct attacks [20], the complexity of the NSDA [8], the complexity of our hybrid attack (NSDA+C for NSDA + crossbred algorithm) and the complexity of NSDA+C with $D = \{3, 4\}$ all given in $\log_2$:

Parameter set	NIST	LUOV	NSDA	NSDA+C	$D = \{3, 4\}$
LUOV-7-57-197	I (143)	143	61	55	56
LUOV-7-83-283	III (207)	208	89	77	81
LUOV-7-110-374	V (272)	274	116	102	106

6.4 Comparison to exhaustive search used in NSDA

Comparing the complexity of the regular NSDA with the complexity of the NSDA+C algorithm, we found that the theoretical application of the crossbred algorithm in the NSDA lowers the complexity for all three parameter sets of LUOV. However, we see that even for feasible values of D $(3, 4)$, the complexity of all three parameters is lower than for the regular NSDA. Furthermore, we see that the more variables and equations a parameter set has, the higher is the efficiency benefit compared to exhaustive search. For the first parameter set we see a benefit of 2^6 in theory (practically 2^5) while for the third parameter set the benefit is 2^{14} (2^{10}).

6.5 Impacts on UOV

As stated in [8], NSDA can not be applied to non-lifted schemes. Therefore, the quadratic system will have coefficients in a finite field, which is to small to find pre-images in. Hence, the system cannot be transformed in a boolean system (coefficients in $\mathbb{F}_2$) and thus, the crossbred algorithm can not be applied. This indicates that the crossbred algorithm has no impact on the security of UOV.

6.6 Scientific impacts and security consequences

The usage of LUOV is, due to the impact of NSDA, already known to be insecure and was eliminated after the second round of the Post-quantum cryptography standardization process. However, the consequences of this research go beyond the LUOV cryptosystem. Given the small field of the coefficients ($\mathbb{F}_2$), it was thought that exhaustive search is the only feasible option to solve a system of quadratic equations. This research shows that the crossbred algorithm by Joux and Vitse is similar efficient and with the right parameters can outstand exhaustive search. In addition to that, the crossbred algorithm shows an algebraic approach to solve boolean systems of quadratic equations in general. Therefore, the attack can be used to reduce a system of quadratic equations to a boolean system and apply the crossbred algorithm to create signature frauds and break cryptosystems in the multivariate quadratic cryptography section. Thus, the crossbred algorithm can be used as an intermediate step of an attack and this research shown that the efficiency of such hybrid attacks can exceed known ones.

Chapter 7

Conclusions

This research investigated the potential of the crossbred algorithm applied to the systems obtained in the nested subset differential attack. To show this, we examined the old and new parameters of LUOV together with the SDA and NSDA. We considered the reduced systems in NSDA and compared several methods to solve boolean quadratic polynomial systems. We decided to examine the crossbred algorithm of Joux/Vitse and used the bivariate generating series [1] and the complexity estimation [9] to create a script (A.1) that leads to the best parameters used in the crossbred algorithm and the complexity. This work has shown that for both, theoretical and practical parameters, our hybrid attack NSDA+C is effective enough to outstand the complexity of the general NSDA. However, the attack is limited to lifted multivariate cryptosystems and non-lifted schemes such as UOV are not affected.

7.1 Future work

This work has done complexity analysis. Therefore, it is worthwhile to change the experimental setup in [8] by implementing the crossbred algorithm and examine wether and if so, how far the 210 minute record can be reduced. Additionally, *Joux & Vitse* addressed a hybrid version of the crossbred algorithm. Future work should examine this hybrid approach and extend this complexity analysis.

Bibliography

[1] Vanessa Joux Antoine and Vitse. A crossbred algorithm for solving boolean polynomial systems. pages 3–21. Springer International Publishing, 2018.

[2] Adi Kipnis Aviad and Shamir. Cryptanalysis of the oil and vinegar signature scheme. pages 257–266. Springer Berlin Heidelberg, 1998.

[3] Sheldon Jay Axler. *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer, New York, 1997.

[4] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and B-Y. Yang. Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems.

[5] Luk Bettale and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Crypt*, 2:1–22, 12 2008.

[6] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 203–218, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[7] C. Cooper. On the distribution of rank of a random matrix over a finite field. In *Proceedings of the Ninth International Conference on on Random Structures and Algorithms*, page 197–212, USA, 2000. John Wiley & Sons, Inc.

[8] Jintai Ding, Joshua Deaton, Vishakha, and Bo-Yin Yang. The nested subset differential attack: A practical direct attack against luov which forges a signature within 210 minutes. Springer-Verlag, 2021.

[9] João Duarte. On the complexity of the crossbred algorithm, 7 2020.

[10] Pierre-Alain Fouque, J. Hoffstein, Paul Kirchner, Vadim Lyubashevsky, T. Pornin, T. Prest, Thomas Ricosset, Gregor Seiler, William Whyte,

and Z. Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. 2019.

[11] John B. Fraleigh. A first course in abstract algebra. 1967.

[12] Juris Hartmanis. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). *SIAM Review*, 24(1):90–91, 1982.

[13] Jacques, Goubin Louis Kipnis Aviad, and Patarin. Unbalanced oil and vinegar signature schemes. pages 206–222. Springer Berlin Heidelberg, 1999.

[14] Joshua, Schmidt Kurt, Vishakha, Zhang Zheng Ding Jintai, and Deaton. Cryptanalysis of the lifted unbalanced oil vinegar signature scheme. pages 279–298. Springer International Publishing, 2020.

[15] D. Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *Computer Algebra*, pages 146–156, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.

[16] Rudolf Lidl and Harald Niederreiter. *Finite fields.*, volume 20. Cambridge: Cambridge Univ. Press, 2nd ed. edition, 1996. Chapter 2.

[17] Albrecht Petzoldt. *Selecting and Reducing Key Sizes for Multivariate Cryptography.* PhD thesis, Technische Universität, Darmstadt, Juli 2013.

[18] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[19] Enrico Thomae and Christopher Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012*, pages 156–171, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] Beullens W., Preneel B., Szepieniec A., and Vercauteren F.:. Luov signature scheme proposal for nist pqc project (round 2 version). 2019.

[21] Bart Beullens Ward and Preneel. Field lifting for smaller uov public keys. pages 227–246. Springer International Publishing, 2017.

Appendix A

Appendix

A.1 Code to decide the best parameters

```
import sys

def adCheck (n, m, D, d, k):
    R.<X,Y> = PowerSeriesRing(ZZ); R
    I = (((1+X)^(n-k))/((1-X)*(1-Y)))*((((1+X*Y)^k)/((1+X^2*Y^2)^(m)))-(((1+X)^k)      /((1+X^2)^(m))))-(((1+Y)^k)/((1-X)*(1-Y)*(1 + Y^2)^(m)));
    dic = I.coefficients();
    if dic.get(X^D*Y^d) < 0:
        return false
    return true


def compComplexity(n,D,d,k):
    #Compute the Complexity
    qnk = 2^(n-k)
    #Compute the complexity of solving P U Mac
    omega = 2.807
    var('i')
    solPMac = (sum(binomial(k,i), i, 0,d))^omega
    #Compute the complexity of the kernel
    var('dk')
    var('dp')
    s = sum(sum((binomial(k,dk) * binomial(n-k,dp)), dp, 0, D-dk), dk, d+1, D)
    #Add the complexities together
    fin = s^2 + qnk * solPMac
    #And compute the log2 and floor the result
    return floor(RDF(log(fin,2)))

def main(n,m):
    bestD = 20
    bestd = 20
    bestk = 20
    bestComp = 200
```

```
    for k in range(1,n+1):
        for D in range(1,floor(sqrt(n))):
            for d in range(1, D):
                if adCheck(n, m, D, d, k):
                    if bestComp > compComplexity(n,D,d,k):
                        bestComp = compComplexity(n,D,d,k)
                        bestD = D
                        bestd = d
                        bestk = k
    print("Finished! :", bestD, bestd, bestk, bestComp)

if __name__=="__main__" :
    if len(sys.argv) != 3 :
        print("To less arguments! Use 'sage parameters.sage <n> <m>'")
    else:
        main(int(sys.argv[1]), int(sys.argv[2]))
```