The Singularity Attack to the Multivariate Signature Scheme HIMQ-3

Jintai Ding, Zheng Zhang, Joshua Deaton and Vishakha

Department of Mathematical Science
University of Cincinnati

Abstract. In 2017 Kyung-Ah Shim et al proposed a multivariate signature scheme called Himq-3 which is a submission to National Institute of Standards and Technology (NIST) standardization process of post-quantum cryptosystems[12]. The Himq-3 signature scheme can be classified into oil vinegar signature scheme family. It has a multilayer structure but it uses a cycle system to invert the central map. The signing process of Himq-3 is very fast, and it has small signatures. In this paper we present a cryptanalysis of Himq-3. We show that inherent to the signing process is a leakage of information of the private key. Using this information one can forge a signature.

Keywords: Multivariate public key cryptosystem · Cryptanalysis · Oil Vinegar Signature Scheme.

1 Introduction

1.1 Background

The ability to authenticate digital messages has always been an important building block for any free, secure, and digital society. In 1976, Whitfield Diffie and Martin Hellman did a major contribution to construct a mathematical framework, known as digital signature scheme, in this direction. Realizing its practical importance, Rivest, Shamir and Adleman proposed their idea called RSA in 1978 whose security relies on the difficulty of the discrete logarithm problem.

The Digital Signature Algorithm (DSA), RSA Digital Signature Algorithm, and The Elliptic Curve Digital Signature Algorithm were the only signature schemes that were allowed under the guidelines of the National Institute of Standards and Technology (NIST)'s up to 2013. However, a mayor drawback to these signature schemes is that Peter Shor [13] proved that they were weak to a sufficiently powerful quantum computer. So quantum computers seem to be a threat to the Cryptosystems because the Shor's algorithm was able to perform the prime factorization of an integer in polynomial time on a quantum computer. Therefore, once these computers are able to handle the factorization of the large integers of quantum bits, the RSA system will be of no use. This indicates a significant need to prepare the current communication system for a post quantum world. For it is no easy nor quick undergoing to transition our current infrastructure into a post quantum one, a consequential effort will have to be done in order to develop, standardize, and establish new post quantum signature schemes.

Since the work of Diffie and Hellman, mathematicians have found many other groups of cryptosystems that do not rely on Number Theory based problems. Multivariate PublicKey Cryptosystems (MPKC) are one of the groups that have potential to resist quantum attack. The security of a MPKC depends on the difficulty of solving a system of multivariate polynomials over a finite field. A breakthrough in MPKC was proposed by Matsumoto and Imai [8] in 1988. Instead of looking for a invertible map between k^n, they looked at the bigger filed K, degree n extension of k, where an inverse map can be constructed. Unfortunately, this scheme was broken by Patarin [10] by using the linearization equation attack. However, inspired by his attack, Patarin [11] proposed the oil vinegar signature scheme. The idea of oil vinegar signature scheme is that certain solvable quadratic equations can be generated if random values are assigned to some variables. The oil vinegar can be classified into three groups: Balanced oil vinegar [11] (Patarin 1997), Unbalanced oil vinegar[6] (Kipnis et al. 1999) and rainbow [3], a multilayer signature scheme with unbalanced oil vinegar at each layer (Ding and Schmidt 2005). Despite the fact that the balanced oil vinegar scheme was broken by Kipnis and Shamir [7] using the method of invariant subspace, both Rainbow and Unbalanced Oil and Vinegar scheme continue to offer promise for post quantum cryptography as can be seen by the fact that both Rainbow and LUOV (lifted unbalanced oil vinegar) [2] have passed into the second round for the new NIST standardization project. Himq-3 can be viewed as another development of oil vinegar scheme, as it is a multilayer signature scheme where the solution of each layer becomes the vinegar variables for the next layer.

1.2 Post Quantum Cryptography Standardization[9]

Due to the rapid development of quantum computers, NIST believes that it is prudent to begin developing standards for post quantum cryptography. Moreover, it is reasonable to plan ahead because a transition to post quantum cryptography will not be simple. A significant effort will be required in order to develop, standardize, and deploy new post quantum cryptosystems. The call for proposals started in Dec 2016. NIST expects to perform multiple rounds of evaluation, over a period of three to five years. The goal of this process is to select a number of acceptable candidate cryptosystems for standardization. These new standards will be used as quantum resistant counterparts to existing standards. The evaluation will be based on the following three criteria: Security, Cost, and Algorithm and Implementation Characteristics. By the end of 2017, 23 signature schemes and 59 encryption/KEM schemes were submitted, of which 69 participated in the first round, Himq-3 is a round 1 submission on the list.

1.3 Our Contributions

We will present an attack to HIMQ-3 by using a new method called the singularity attack. The method comes from a special property of a particular type of variables that play an important role in inverting the central map, and such property can not be hidden even under change of basis. We will show that this flaw will make it possible for us to obtain a part of a private key, then we can turn the public key into the form in which forgery can be done.
First, we will recall general construction of a MPKC signature scheme, then we will describe a system called L-cycle, which makes it possible for the central map of Himq-3 to be invertible. Next, we will introduce Himq-3 signature scheme. Moreover, we will point out the Achilles' heel in the design and explain why it will reveal information about the private key. Before we introduce our attack, we will estimate how many signatures are needed to perform the attack. The attack will be described into two parts. First, we will show that given enough signatures, part of private key can be obtained due to that flaw in the design. Next, using that part of a private key, we show that one can separate the variables and layers easily using the knowledge of matrix theory. A complexity analysis will be provided in section 3.13 to show that Himq-3 does not meet the NIST requirement. Last but not least, we will give some experimental results.

2 HIMQ-3 signature scheme

2.1 Preliminary

General Construction of Bipolar MPKC signature scheme We first describe the general construction of a Bipolar MPKC signature scheme. Let $\mathbb{F}_q$ be a finite field of order q. The main idea for the construction of MPKC signature schemes is to construct a polynomial map $\mathscr{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, called the central map, defined by $\mathscr{F} = (\mathscr{F}^{(1)}, \cdots, \mathscr{F}^{(m)})$ of m equations in n variables such that it is easy to find pre-images

for a given vector. To hide the ability to find pre-images and thus construct a public key from $\mathscr{F}$, one uses two invertible affine maps $\mathscr{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$, and $\mathscr{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$. The public key is the composition $\mathscr{P} = \mathscr{S} \circ \mathscr{F} \circ \mathscr{T}$. The private keys are the invertible affine maps $\mathscr{S}$ and $\mathscr{T}$. The signing process for a document is as follows:

$$\mathbb{F}_q^m \xrightarrow{\mathscr{S}^{-1}} \mathbb{F}_q^m \xrightarrow{\mathscr{F}^{-1}} \mathbb{F}_q^n \xrightarrow{\mathscr{T}^{-1}} \mathbb{F}_q^n$$

The verification process is just backwards

$$\mathbb{F}_q^m \xleftarrow{\mathscr{P}} \mathbb{F}_q^n$$

L-invertible cycle system [4] The Himq-3 scheme contains a system of quadratic equations called L-invertible cycle system. This system makes it possible to invert the central map.
Suppose $\mathbb{F}_q$ is a field of characteristic 2 and l is an odd positive integer. The cycle products system $\mathscr{Q}$ is defined by:

$$\mathscr{Q} : \alpha_1 x_1 x_2 = \beta_1, \alpha_2 x_2 x_3 = \beta_2, \cdots, \alpha_l x_l x_1 = \beta_l,$$

where α_i and β_i are nonzero elements in $\mathbb{F}_q$.
To find a solution to $\mathscr{Q}$, first write the cycle products in the form

$$x_1 x_2 = \gamma_1, \cdots, x_l x_1 = \gamma_l,$$

where $\gamma_i = \beta_i / \alpha_i$. Let $A = \gamma_1 \gamma_2 \cdots \gamma_l$ and $B = \gamma_2 \gamma_4 \cdots \gamma_{l-1}$. It is easy to see that $x_1 = \sqrt{A}/B$, $x_i = \gamma_{i-1}/x_{i-1}$ for $i = 2, \cdots, l-1$, and $x_l = \gamma_l / x_1$.

Note: None of the variables x_i can be equal to zero in the L-invertible cycle system, otherwise, the system does not have a solution. Furthermore, in the signing process, this property can not be hidden under change of basis. Thus, this is a weakness that Himq-3 scheme cannot get rid off; an attacker can use this to forge a signature.

2.2 Description of the HIMQ-3 scheme[12]

The Himq-3 scheme can be described as a combination of rainbow scheme[3], unbalanced oil vinegar scheme [6], and L-invertible cycle system[4]. First of all, it shares the layer structure with the rainbow scheme, and the signing process of both schemes is very fast. Second, each layer contains vinegar variables and oil variables, which are mixed by the change of basis matrix. The difference is that Himq-3 contains three types of oil variables that are contained in different layers. Third, Himq-3 uses the L-invertible cycle system to make its central map invertible. We will now describe the particulars of the HIMQ-3 central map.

Let us denote $\mathbb{F}_q$ to be the finite field of order $q = 2^k$. Let v, o_1, o_2, o_3 be positive integers where o_1 and o_2 are odd. Further, let $v_1 = v + o_1$, $v_2 = v + o_1 + o_2$, $m = o_1 + o_2 + o_3$ and $n = v + o_1 + o_2 + o_3$. Let $\mathbf{X} = (x_1, \cdots, x_n)$. Let $\mathscr{F} = (\mathscr{F}^{(1)}, \cdots, \mathscr{F}^{(m)})$ be the central map defined by three layers:

$$\begin{cases} \mathscr{F}^{(1)}(\mathbf{X}) = \Phi_1(\mathbf{X}) + \delta_1 x_{v+1} x_{v+2} \\ \mathscr{F}^{(2)}(\mathbf{X}) = \Phi_2(\mathbf{X}) + \delta_2 x_{v+2} x_{v+3} \\ \vdots \\ \mathscr{F}^{(o_1)}(\mathbf{X}) = \Phi_{o_1}(\mathbf{X}) + \delta_{o_1} x_{v+o_1} x_{v+1} \end{cases}$$

$$\begin{cases} \mathscr{F}^{(o_1+1)}(\mathbf{X}) = \Psi_1(\mathbf{X}) + \delta_{o_1+1} x_{v_1+1} x_{v_1+2} \\ \mathscr{F}^{(o_1+2)}(\mathbf{X}) = \Psi_2(\mathbf{X}) + \delta_{o_1+2} x_{v_1+2} x_{v_1+3} \\ \vdots \\ \mathscr{F}^{(o_1+o_2)}(\mathbf{X}) = \Psi_{o_2}(\mathbf{X}) + \delta_{o_1+o_2} x_{v_1+o_2} x_{v_1+1} \end{cases}$$

$$\begin{cases} \mathscr{F}^{(o_1+o_2+1)}(\mathbf{X}) = \sum\limits_{v+1 \le i \le j \le v_1} \beta_{i,j}^{(1)} x_i x_j + \Theta_1(\mathbf{X}) + \Theta'_1(\mathbf{X}) + \epsilon_1 x_{o_1+o_2+1} \\ \vdots \\ \mathscr{F}^{(o_1+o_2+o_3)}(\mathbf{X}) = \sum\limits_{v+1 \le i \le j \le v_1} \beta_{i,j}^{(o_3)} x_i x_j + \Theta_{o_3}(\mathbf{X}) + \Theta'_{o_3}(\mathbf{X}) + \epsilon_{o_3} x_{o_1+o_2+o_3} \end{cases}$$

For $i = 1, \cdots, o_1$, we call $\mathscr{F}^{(i)}$ to be the polynomials of the first layer. The term $\Phi_i(\mathbf{X})$ is a quadratic polynomial in the variables $(x_1, \cdots, x_v)$ defined by

$$\Phi_i(\mathbf{X}) = \sum_{j=1}^{v} \alpha_{i,j} x_j x_{1+(i+j-1) (\mathrm{mod}\ v)}$$

where $\alpha_{i,j}$ is a nonzero element in $\mathbb{F}_q$.
The polynomials $\mathscr{F}^{(i)}$ for $i = o_1+1, \cdots, o_1+o_2$ form the second layer. The term $\Psi_i(\mathbf{X})$ is a quadratic polynomial in the variables $(x_1, \cdots, x_{v+o_1})$ defined by

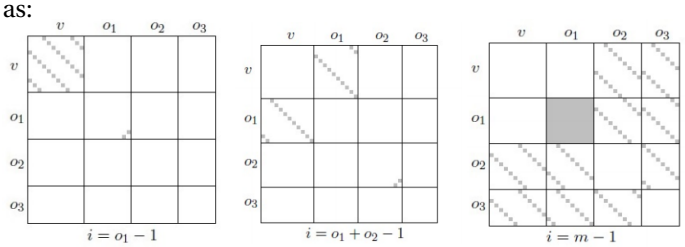
$$\Psi_i(\mathbf{X}) = \sum_{j=1}^{v} \alpha'_{i,j} x_j x_{v+(i+j-1) (\mathrm{mod}\ o_1)}$$

where $\alpha'_{i,j}$ is a nonzero element in $\mathbb{F}_q$.
This leaves the third layer. The $\beta_{i,j}^{(k)}$ are elements in the finite field $\mathbb{F}_q$. Θ_i and Θ'_i in the third layer are quadratic equations in variables $(x_1, \cdots, x_n)$ defined by

$$\Theta_i(\mathbf{X}) = \sum_{j=1}^{v_1} \gamma_{i,j} x_j x_{v_1+(i+j-1) (\mathrm{mod}\ o_2)},\ \Theta'_i(\mathbf{X}) = \sum_{j=1}^{v_2} \gamma'_{i,j} x_j x_{v_2+(i+j-1) (\mathrm{mod}\ o_3)}$$

where $\gamma_{i,j}$ and $\gamma'_{i,j}$ are nonzero elements in $\mathscr{F}_q$. We notice also that there are linear terms in the third layer defined by some ϵ_i in $\mathbb{F}_q$, and that the variables $x_{v_2}, \cdots, x_n$ are never multiplied together like oil variables in a oil vinegar scheme.
The design rational of the individual $\Phi_i, \Psi_i, \Theta_i, \Theta'_i$ is the increase the rank of the associated symmetric matrices of the polynomial they are in to the maximum amount of rank for the variables they involve, and they do this using the least amount of terms. As the field has characteristic 2, the Φ_i, being quadratics of the first v variables, will have associated matrices of rank v. The Ψ_i will have rank $2o_1$. Combining the terms of the $\beta_i, \Theta_i, \Theta'_i$ will lead to a matrix of full rank.

We notice also that there are linear terms in the third layer defined by some ϵ_i in $\mathbb{F}_q$, and that the variables $x_{v_2}, \cdots, x_n$ are never multiplied together like oil variables in a oil-vinegar scheme. The symmetric matrices of each layers can be roughly viewed as:



2.3 Inverting the central map

Given a $M = (M_1, \cdots, M_m)$ in $\mathbb{F}_q^m$, we want to compute $\mathscr{F}^{-1}(M) = \mathbf{s}$.

1 Randomly generate $\mathbf{s_v} \in \mathbb{F}_q^v$, and plug $\mathbf{s_v}$ into the first layer obtaining the cycle product

$$\begin{cases} \delta_1 x_{v+1} x_{v+2} = M_1 - \Phi_1(\mathbf{s_v}) \\ \vdots \\ \delta_{o_1} x_{v+o_1} x_{v+1} = M_{o_1} - \Phi_{o_1}(\mathbf{s_v}) \end{cases}$$

2 If $M_i - \Phi_i(\mathbf{s_v}) \neq 0$ for all i, then solve by the process described before. Name this $\mathbf{s_{v_1}} \in \mathbb{F}_q^{v_1}$. Otherwise, return to step 1.
3 Plug $\mathbf{s_{v_1}}$ into the second layer creating another cycle product

$$\begin{cases} \delta_{o_1+1} x_{v_1+1} x_{v_1+2} = M_{o_1+1} - \Psi_1(\mathbf{s_{v_1}}) \\ \vdots \\ \delta_{o_1+o_2} x_{v_1+o_2} x_{v_1+1} = M_{o_1+o_2} - \Psi_{o_1}(\mathbf{s_{v_1}}) \end{cases}$$

If $M_{o_1+i} - \Psi_i(\mathbf{s_v}) \neq 0$ for all i, call the solution $\mathbf{s_{v_2}} \in \mathbb{F}_q^{v_2}$. Else, return to step 1.
4 Plug $\mathbf{s_{v_2}}$ into the third layer. It will thus have only linear terms. Use Gaussian Elimination to see if there is a solution. If so, then the solution is $\mathbf{s}$. Otherwise, return to step 1.

3 The Singularity Attack

3.1 General idea of our attack

The key observation is that the cycle variables cannot be equal to zero when evaluated at a honestly generated signature. In addition, this fact does not change under the change of basis $\mathscr{T}$. Since the scheme is constructed over a finite field of 2^k elements, it is a basic knowledge that if we raise any nonzero element a in the field

to the power of 2^k-1, then $a^{2^k-1}=1$. For this reason, if we evaluate the cycle variables at the signatures under the change of basis,and then raise the power, we will obtain some equations. Thus, if we have access to enough signatures, we will obtain enough equations. Once we solve these equations, we will get part of the private key up to scalar multiplication. This process consumes the most memory in our attack. The next step is to use this known part of secrete key to separate the layers and variables. This is not hard to accomplish, only elementary linear algebra is needed. In other words, we will perform linear transformations on the public key to turn it in the form that we can forge signatures. Hence, one can see that our attack is theoretically straightforward. There are several variants of HIMQ-3 as presented by Kyung-Ah Shim et al. For simplicity's sake we will present the basis variant, but our attack will work against Himq-F as well.

3.2 Notations and definitions

We call elements in $\{x_1,\cdots,x_v\}$ the v variables, in $\{x_{v+1},\cdots,x_{v+o_1}\}$ the o_1 variables, in $\{x_{v_1+1},\cdots,x_{v_1+o_2}\}$ the o_2 variables, and in $\{x_{v_2+1},\cdots,x_n\}$ the o_3 variables. Therefore, the cycle variables are o_1 and o_2 variables. We use the notation V to denote the set of v variables. Likewise, O_1,O_2 and O_3 for o_1,o_2 and o_3 variables respectively. In the attack, we can not always get the original variables, but we can only get the space spanned by these variables. For ease of notation, V, O_1, O_2, and O_3 will also mean the set of basis for the vector space spanned by v, o_1, o_2, and o_3 variables respectively. Let us define f_i to be the first layer polynomial, g_i to be the second layer polynomial, h_i to be the third layer polynomial. Furthermore, f_i' is a first layer polynomial after change of basis. g_i' is a second layer polynomial after change of basis, h_i' is a third layer polynomial after change of basis.

3.3 Finding parts of $\mathscr{T}$

Suppose that a private key $(\mathscr{F},\mathscr{T},\mathscr{S})$ has been generated with its corresponding public key $\mathscr{P}=\mathscr{S}\circ\mathscr{F}\circ\mathscr{T}$. We may describe the affine map $\mathscr{T}$ by an invertible matrix $(a_{ij})_{1\le i,j\le n}$ and a vector $b=(b_1,\cdots,b_n)$ so that for any $(x_1,\cdots,x_n)\in\mathbb{F}_q^n$ we have that

$$\mathscr{T}((x_1,\cdots,x_n))=\begin{bmatrix}a_{11}&a_{12}&\dots&a_{1n}\\a_{21}&a_{22}&\dots&a_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{n1}&a_{n2}&\dots&a_{nn}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}+\begin{bmatrix}b_1\\b_2\\\vdots\\b_n\end{bmatrix}=\begin{bmatrix}\sum_{i=1}^n a_{1i}x_i+b_1\\\sum_{i=1}^n a_{2i}x_i+b_2\\\vdots\\\sum_{i=1}^n a_{ni}x_i+b_n\end{bmatrix}$$

The first goal of the attack is to find how this map $\mathscr{T}$ changes the variables used in the cycle products up to a multiplication by a non-zero constant. That is, for $v+1\le j\le v+o_1+o_2$, we want to find $\gamma_j\left(\sum_{i=1}^n a_{ji}x_i+b_j\right)$ for some $\gamma_j\in\mathbb{F}_q^*$. This can be done as if we denote a signature $\sigma=(\sigma_1,\cdots,\sigma_n)$, then for $v+1\le j\le v+o_1+o_2$

$$\sum_{i=1}^n a_{ji}\sigma_i+b_j\neq 0$$

because a cycle variable cannot be zero when evaluated at a signature by the signing process described above. As $\mathbb{F}_q$ is a field, multiplying by a non-zero constant still yields another non-zero constant. This allows us to say that for any $\gamma \in \mathbb{F}_q^*$ that for any signature σ

$$1 = \left(\sum_{i=1}^{n} \gamma_j a_{ji}\sigma_i + \gamma_j b_j\right)^{2^k-1} = \prod_{h=1}^{k}\left(\sum_{i=1}^{n} \gamma_j a_{ji}\sigma_i + \gamma_j b_j\right)^{2^{k-h}}$$

As we are working in characteristic two we have that

$$\prod_{h=1}^{k}\left(\sum_{i=1}^{n} \gamma_j a_{ji}\sigma_i + \gamma_j b_j\right)^{2^{k-h}} = \prod_{h=1}^{k}\left(\sum_{i=1}^{n} (\gamma_j a_{ji}\sigma_i)^{2^{k-h}} + (\gamma_j b_j)^{2^{k-h}}\right).$$

First we will solve the case when $b_j \neq 0$. We can thus set $\gamma_j = b_j^{-1}$ to obtain

$$\prod_{h=1}^{k}\left(\sum_{i=1}^{n} (b_j^{-1} a_{ji}\sigma_i)^{2^{k-h}} + 1\right) = 1.$$

For the sake of notation, let $\tilde{a}_{ji} = b_j^{-1} a_{ij}$. Thus, we see by performing the above product that

$$\tilde{a}_{j1}^{2^k-1}\sigma_1^{2^k-1} + \tilde{a}_{j1}^{2^k-2}\tilde{a}_{j2}\sigma_1^{2^k-2}\sigma_2 + \cdots + \tilde{a}_{jn}\sigma_n + 1 = 1.$$

We can treat the individual products of the $\tilde{a}_{ij}$'s as individual variables to get a homogeneous linear equation with $(n+1)^k - 1$ terms. We get another homogeneous linear equation if we use a different signature. Hence by collecting around $(n+1)^k - 1$ signatures we can form a matrix in the following way.

Construction of the matrix For $v+1 \leq j \leq v+o_1+o_2$, we list the products of $\tilde{a}_{ij}$ in the order: $\tilde{a}_{j1}^{2^k-1}, \tilde{a}_{j1}^{2^k-2}\tilde{a}_{j2}, \cdots, \tilde{a}_{jn}$ (Here we use lexicographic order on $(l_1, -l_1', l_2, -l_2', ...)$ for products $\tilde{a}_{jl_1}^{l_1'}\tilde{a}_{jl_2}^{l_2'}...$). Moreover, for each signature $\sigma = (\sigma_1, \cdots, \sigma_n)$, the corresponding coefficients are: $\sigma_1^{2^k-1}, \sigma_1^{2^k-2}\sigma_2, \cdots, \sigma_n$. The matrix is simply constructed by having these corresponding coefficients as a row for each signature we use. Therefore the size of this matrix is $(n+1)^k-1$ by $(n+1)^k-1$ if we use $(n+1)^k-1$ signatures. Hence, we obtain a homogeneous linear system: $\mathbf{Ax} = \mathbf{0}$, where $\mathbf{A}$ is the matrix whose rows are $(\sigma_1^{2^k-1}, \sigma_1^{2^k-2}\sigma_2, \cdots, \sigma_n)$ for each signature used, and $\mathbf{x} = (\tilde{a}_{j1}^{2^k-1}, \tilde{a}_{j1}^{2^k-2}\tilde{a}_{j2}, \cdots, \tilde{a}_{jn})^T$.

Remark 1. Assume that $b_j \neq 0$, for $v+1 \leq j \leq v+o_1+o_2$, $\tilde{\mathbf{a}}_j = (\tilde{a}_{j1}^{2^k-1}, \tilde{a}_{j1}^{2^k-2}\tilde{a}_{j2}, \cdots, \tilde{a}_{jn})^T$ is contained in the kernel of $\mathbf{A}$. Moreover, it is obvious that they are linearly independent. It follows that $\text{Rank}(\mathbf{A}) \leq (n+1)^k - 1 - (o_1+o_2)$. In fact, according to our experiments (see chapter 4), with very high probability, $\text{Rank}(\mathbf{A}) = (n+1)^k - 1 - (o_1+o_2)$.

Solving the $\tilde{a}_{ji}$'s We do Gaussian Elimination on this matrix $\mathbf{A}$, and turn the linear system into $\mathbf{A}'\mathbf{x} = \mathbf{0}$. Now we try to solve for the $\tilde{a}_{ji}$'s. We start at the bottom of $\mathbf{A}'$. If

A has rank $(n+1)^k - 1 - (o_1 + o_2)$, then in the last nonzero row of $\mathbf{A}'$, most entries will equal to zero and the nonzero entries will only appear in the last o_1+o_2+1 columns in variables $\tilde{a}_{jn}^{o_1+o_2+1}, \tilde{a}_{jn}^{o_1+o_2}, \tilde{a}_{jn}^{o_1+o_2-1}, \cdots, \tilde{a}_{jn}$. Hence, converting this back into a polynomial means we have a univariate polynomial equation which we can thus solve. One can see that if $2^k - 1 \geq o_1 + o_2 + 1$, we will obtain a univeriate polynomial. This allows us to get our possibilities for $\tilde{a}_{jn}$ (as the above equation will be true for any of the $\tilde{a}_{ji}$'s, $v+1 \leq j \leq v+o_1+o_2$, we will return all of these values). We then move up the matrix to the first time that $\tilde{a}_{j(n-1)}$ appears only with powers of itself and $\tilde{a}_{jn}$. As we already know what $\tilde{a}_{jn}$ can be, this is also a univariate polynomial equation. For each of our possible solutions to $\tilde{a}_{jn}$, we plug in and get the possible solutions to $\tilde{a}_{j(n-1)}$. Continue this process until we collect all the $\tilde{a}_{ji}$ for which $b_j \neq 0$. On the other hand, to avoid the inequality $2^k - 1 \geq o_1 + o_2 + 1$, the size of the field is then forced to be small, which reduces the complexity of other attacks such as direct attack, min/high rank attack (see Section 2.2, 2.3 and 2.4 in [12]). The process is essentially the same as for the case $b_j = 0$ except that we then guess the last available $\tilde{a}_{ji}$ to be non-zero hence enabling us to set $\gamma_j = \tilde{a}_{ji}^{-1}$ for that particular $\tilde{a}_{ji}$. Repeat until all of the $\tilde{a}_{ji}$ are found, which generally is after the first few guesses. Note that the collection of $\tilde{a_{ji}}$ that we found can recover the cycle variables. A toy example is provided in the appendix.

3.4 Separating second layer in the public key

Let us recall how the polynomials of the central map are defined for the three different layers. A first layer polynomial f_i is defined by

$$f_i = \Phi_i(\mathbf{X}) + \delta_i x_{v+i} x_{v+i+1}$$

where Φ_i has only the first v variables times one of themselves. A second layer polynomial g_i is defined by

$$g_i = \Psi_i(\mathbf{X}) + \delta_i x_{v_1+i} x_{v_1+i+1}$$

where Ψ_i has terms with a cycle variable as a factor. A third layer polynomial h_i is more complicated but must contain terms of one of the first v variables being multiplied by one of the last o_3 variables. Thus, if we set the cycle product variables to be zero, the second layer polynomials will vanish but not those from the first and third. This can be accomplished by forming the quotient ring $\mathbb{F}_q[X]/ < O_1 \cup O_2 >$.

The public key can be denoted by

$$\mathscr{S} \circ \mathscr{F} \circ \mathscr{T} = P = \begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{1,i} h_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{n,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{n,i} h_i{}' \end{cases}$$

If we place the public key into the quotient ring $\mathbb{F}_q[X]/ < O_1 \cup O_2 >$, we see that the second layer polynomials vanish, leaving the first and third layer polynomials

alive. The public key will be in the form:

$$\begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} \tilde{f}_i + \sum_{i=1}^{o_3} \gamma_{1,i} \tilde{h}_i \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} \tilde{f}_i + \sum_{i=1}^{o_3} \gamma_{n,i} \tilde{h}_i \end{cases}$$

where $\tilde{f}_i, \tilde{g}_i, \tilde{h}_i$ represent f_i', g_i', h_i' in the quotient ring. If we take the coefficients of each term in these m polynomials and build a matrix of these coefficients with an order, then doing Gaussian Elimination would leave the bottom o_2 equations as zero and thus representing linear combinations of the second layer polynomials in this quotient ring. Record the transformation that performs the Gaussian Elimination, and apply it to the public key, we have the second layer separated from the other layers.

$$P_1 = \begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} {f_i}' + \sum_{i=1}^{o_2} \beta_{1,i} {g_i}' + \sum_{i=1}^{o_3} \gamma_{1,i} {h_i}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} {f_i}' + \sum_{i=1}^{o_2} \beta_{n,i} {g_i}' + \sum_{i=1}^{o_3} \gamma_{n,i} {h_i}' \\ \sum_{i=1}^{o_2} \beta_{1,i} {g_i}' \\ \vdots \\ \sum_{i=1}^{o_2} \beta_{1,i} {g_i}' \end{cases}$$

Note that the last o_2 polynomials are not the original second layer polynomials, they are actually the linear combinations of the original second layer polynomials. But we still call these linear combinations the "second layer".

3.5 Separating a combination of the first and second layers from the rest

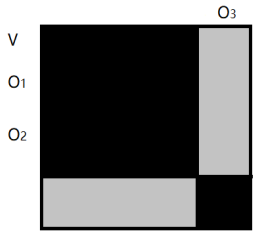
One can observe that o_3 variables only appear in the third layer, and are multiplied by v, o_1, o_2 variables. In addition, first layer and second layer does not contain o_3 variables. Hence, we may use O_3 to get rid of third layer. In fact, O_3 is not hard to obtain. It can be seen that O_3 is always contained in the common kernel of second layer. Taking intersecting the null-spaces of the symmetric matrices of second layer, one can find O_3. On the other hand, $V \cup O_1 \cup O_2$ can be obtained by collecting the images of the symmetric matrices of second layer. However, we may not always get it completely. We will provide an estimation of the probability of getting the space with experimental results in section 3.14.

Having O_3 and $V \cup O_1 \cup O_2$, we construct a matrix defined by:

$$\begin{bmatrix} O_3 \\ V \cup O_1 \cup O_2 \end{bmatrix}.$$

We will act a change of basis by this matrix on the symmetric matrices of the first $o_1 + o_3$ polynomials in P_1. Of course, we do not want to bother the second layer that

is already separated out. Using the change of basis, the symmetric matrices will be in the following form



The point of using change of basis is to force o_3 variables time o_3 variables to be in their own separate submatrix (bottom right corner part). Meanwhile, the v, o_1, o_2 variables time v, o_1, o_2 variables will also be in their own separate submatrix (up left corner). The top right corner and bottom left corner will thus represent v, o_1, o_2 variable times a o_3 variable. Hence, polynomials from first and second layer should have zeros there, while polynomials from third layer should have nonzeros there. Hence, we perform a Gaussian elimination only on up right corner part. In the resulting matrix, the zeros in the bottom will represent the combinations of the first and second layer, and the nonzeros in the top will represent the third layer. Record the transformation of the Gaussian elimination and apply it to the symmetric matrices of the first $o_1 + o_3$ polynomials in P_1, we can get rid of the third layer.

$$P_2 = \begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{1,i} h_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{n,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{n,i} h_i{}' \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \vdots \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \end{cases}$$

3.6 Separating first layer from first+second layer

Our goal now is to remove the polynomials from the original second layer from our new second layer made of linear combinations of first and second layer polynomials

and to get these polynomials to have a low rank so that they will almost be in the form for forging signatures. First we will modify the third layer of P_2 such that the associated symmetric matrices are of lowest rank. We may do this by forming linear combinations of the matrices, and if the ranks drop, replacing the old polynomial with the new one. We may get these matrices to rank $2o_1+2$. We now focus on the polynomials in the second layer of P_2. Due to how the $\mathscr{S}$ might have mixed the polynomials together, it might be impossible to only add polynomials from the second layer of P_2 to themselves to reduce to rank fully. We might remove part of a original first layer but then add again a second layer leaving the rank unchanged. To defeat this we examine each second layer polynomial in turn, trying to lower rank alternating between linear combinations from the same second layer and the third. Even if the rank is not immediately fully reduced, we move to the next polynomial and repeat the process. After several passes through all the polynomials, their rank will be reduced to $v+1$. We do note that even though these polynomials have matrices of lowest rank, it does not mean that they are in the original forms as having one or two pairs of cycle products leaves the rank the same.

$$P_3 = \begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{1,i} h_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{n,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{n,i} h_i{}' \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \vdots \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \end{cases}$$

3.7 Switch the order

We may simply switch the order, and the matrix that does this job can be easily found.

$$P_4 = \begin{cases} \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \vdots \\ \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' \\ \sum_{i=1}^{o_1} \alpha_{1,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{1,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{1,i} h_i{}' \\ \vdots \\ \sum_{i=1}^{o_1} \alpha_{n,i} f_i{}' + \sum_{i=1}^{o_2} \beta_{n,i} g_i{}' + \sum_{i=1}^{o_3} \gamma_{n,i} h_i{}' \end{cases}$$

3.8 Finding V, O_1, O_2, O_3 individually

We will now try to undo the effect of $\mathcal{T}$. Before we invert the change of variables, we have to find the space V, O_1, O_2, O_3 individually. These spaces are mixed by the private key, hence it is difficult to find these individual spaces directly. However, the known part of $\mathcal{T}$ already tells us the $O_1 \cup O_2$ space. So, our goal now becomes separating these variables.

3.9 Separate o_1 variables from o_2 variables

Note that in the second layer of central map, o_1 variables are multiplied by v variables, and o_2 variables are multiplied by o_2 variables. Using this observation, one can distinguish o_1 and o_2 variables from the set $O_1 \cup O_2$ obtained in 3.3. Firstly, in the second layer (obtained in 3.4) one can set all elements in $O_1 \cup O_2$ equal to zero except one. If the element is an o_1 variable, then the $v \times o_1$ block (in a symmetric matrix in section 2) will be alive. On the other hand, if the element is an o_2 variable, then the $v \times o_1$ block will be killed. Hence, by following this procedure for all the elements of $O_1 \cup O_2$, we can tell which one belongs to O_1 and which one belongs to O_2. In other words, we can separate O_1 and O_2 from $O_1 \cup O_2$.

Finding $V \cup O_1$ This can be found by combining the image space of the matrices from the first layer of P_4 as they only contain variables from $V \cup O_1$.

Separate V from $V \cup O_1$ Note that $V \cup O_1$ is contained in the image space of each matrix of the first layer. We find V by intersecting $V \cup O_1$ with the image spaces of the matrices until the dimension of the space is v.

3.10 Invert the change of basis

We now have all the information required to create a change of basis E which will undo $\mathcal{T}$'s effect of hiding the cycle structure in the public key. The exact shape of both the first v variables and the last o_3 variables do not matter as they do not appear in a cycle product. Their values are found by either guessing or Gaussian Elimination. As long as these variables are mapped to a linear combination of themselves we will have no problem inverting the central map as done in the original scheme. Hence, having just V and O_3 is enough. However, the variables for the cycle products must each be mapped to another cycle product variable. That is, we must know exactly how $\mathcal{T}$ changed these variables, and also its affine part cannot be ignored. Fortunately, we have already found this upto a scalar multiple when we found $O_1 \cup O_2$. The change of basis we require can be constructed as follows: Let

$$E = \begin{bmatrix} V \\ O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

and the affine part in the order we choose as c_i for $i = 1, \cdots, o_2$, and finally let

$$\mathbf{x}^T = \left[x_1, \cdots, x_v, x_{v+1} + c_1 \cdots, x_{v+o_1+o_2} + c_{o_2}, x_{v+o_1+o_2+1}, \cdots, x_{v+o_1+o_2+o_3}\right]$$

The inverse of change of basis can be done by computing $P_5 = E^{-1}P_4$, and P_5 is obtained.

3.11 Modify first and second layer into proper form

After change of basis, we can see that the matrices of the first and second layer are nearly in the form of what we want. However we have to clean the cycle variable part so that each polynomial in the first and second layer contains only one cycle product. This can easily be done by Gaussian Elimination. For ease of coding, it is helpful to get the cycle products exactly as they were originally. Examining which are multiplied together and in what order allows easy reindexing. The order of polynomials may also be off, but it is not difficult to re-order these polynomials by applying a permutation on them so that the cycle product $x_{v+i}x_{v+i+1}$ is only contained in the i-th polynomial. We denote the modified public key as P_6.

3.12 Forgery

Now we are ready to forge the signature. Let us denote the document $m := (M_1, \cdots, M_m)$. The forgery process is almost the same process as signing. We randomly assign values to v variables in P_6, then we will get a cycle system in the first layer, which can be solved by using exact the same method in the signing process. Next, we plug in the values of v and o_1 to the second layer, and solve the cycle products. Finally plug in values of v, o_1, o_2 in the third layer and solve the linear system. Therefore, we have a solution

$$\mathbf{s}_1 = (\bar{s}_1, \cdots, \bar{s}_v, \bar{s}_{v+1}, \cdots, \bar{s}_{v+o_1}, \bar{s}_{v+o_1+1}, \cdots, \bar{s}_{v+o_1+o_2}, \bar{s}_{v+o_1+o_2+1}, \cdots, \bar{s}_{v+o_1+o_2+o_3})$$

Now we apply the inverse of the permutation in section 3.10 to the solution $\mathbf{s}_1$. By doing this, we turn back the order of $\mathbf{s}_1$ so that it matches the change of basis matrix. Moreover, we have to add the constants $(c_1, \cdots, c_{o_2})$ of $\mathcal{T}$ to the cycle variable part. Let us call the modified solution $\mathbf{s}'$. Applying the change of basis matrix E, we obtained the forged signature $\mathbf{s} = E\mathbf{s}'$. One can use $\mathbf{s}$ to forge signature for the document m.

3.13 Complexity

In our attack, the most complicated step is to do Gaussian elimination over a linear system of dimension $(n+1)^k - 1$. The complexity of solving such linear system is $((n+1)^k - 1)^\omega$, where ω called the complexity exponent of linear algebra [1]. The best published estimates to date gives $\omega \approx 2.3727$ [15][5]. A practical algorithm that is frequently used for implementation is Strassen-Winograd's algorithm [14] with $\omega \approx 2.8047$. For $v = 31$, $o_1 = o_2 = 15, o_3 = 14$ and $k = 8$, the parameters for 128-bit security parameter proposed in [12], we need approximately 2^{50} signatures. We estimate the complexity to be from 2^{119} using [15] and 2^{140} using [14].

3.14 Experimental Result

In section 3.5, we may not always get the space $V \cup O_1 \cup O_2$. We give an analysis of the probability of this case occurs.
We know that there are o_1 column vectors in the $v \times o_1$ part of each symmetric matrix in second layer. So we have $o_1 o_2$ such vectors. Assume that these $o_1 o_2$ vectors do not span the entire V space. So we can take $v-1$ vectors and look at the span of these $v-1$ vectors. Therefore, the probability of the next vector being in the span of these $v-1$ vector is $1/q$. There are $o_1 o_2 - v - 1$ vectors to check, hence, the probability of failing to fill the entire space is $1/q^{o_1 o_2 - v - 1}$.
We ran our attack with Magma over two sets of parameters. The first set is:

$$v = 7, o_1 = 3, o_2 = 3, o_3 = 2, n := 15, m := 8, q = 2^3,$$

where in 1000 attempts we can always get some part of $\mathscr{T}$ but 163 times we cannot get the $V \cup O_1 \cup O_2$ space.

The second set is:

$$v = 31, o_1 = 15, o_2 = 15, o_3 = 14, n = 75, m = 44, q = 2^8.$$

We cannot run these actual parameters on our computer, but we can analyze the probability of getting the $V \cup O_1 \cup O_2$ space correctly. In 1000 attempts, we get the space every time.

4 Appendix: Toy example

We provide a toy example to clarify the step **3.2**. In this example, we choose $k = 3$, thus our field is the Galois field of 2^3 elements. The Galois field will be represented by $\{0, 1, w, w^2, \cdots, w^6\}$, where w is a generator in the multiplicative group of the Galois field. Let $n = 2$. For the sake of clarity. We use a linear map instead of a affine map. Our linear map $\mathscr{T}$ is randomly chosen to be the matrix $\begin{bmatrix} w^2 & w^2 \\ w^3 & w \end{bmatrix}$.
Suppose we obtain a set of signatures (x_1, x_2):

$$(w, w^5), (w^5, w), (w^2, 1), (w^6, w^5), (0, w^2), (w^5, w^3), (1, w^6), (0, w^5),$$
$$(0, w^2), (1, 0), (w^5, w^6), (0, w), (w^5, w^3), (1, w), (w^5, 0), (w^6, 1), (w^6, w^3),$$
$$(w, w^4), (w^2, w^5), (w^3, w), (1, w^6), (w, 1), (w^2, w), (w^2, w), (w^4, w), (w^4, 1), (w^4, w^2).$$

We first construct a generic polynomial $g = a_1 x_1 + a_2 x_2$. We assume that this polynomial is never equal to zero. Hence, in this Galois field, $g^{2^3-1} = (a_1 x_1 + a_2 x_2)^{2^3-1} = 1$. By elementary field theory, we can rewrite this equation as

$$(a_1 x_1 + a_2 x_2)^{2^3-1} = (a_1 x_1 + a_2 x_2)^{2^3-1} (a_1 x_1 + a_2 x_2)^{2^3-2} (a_1 x_1 + a_2 x_2)^{2^3-3} = 1$$

Since this is a field of characteristic 2, the equations turns out to be

$$((a_1 x_1)^{2^3-1} + (a_2 x_2)^{2^3-1})((a_1 x_1)^{2^3-2} + (a_2 x_2)^{2^3-2})((a_1 x_1)^{2^3-3} + (a_2 x_2)^{2^3-3}) = 1$$

Multiply the product out, we have

$$a_1^7 x_1^7 + a_1^6 a_2 x_1^6 x_2 + a_1^5 a_2^2 x_1^5 x_2^2 + a_1^4 a_2^3 x_1^4 x_2^3 + a_1^3 a_2^4 x_1^3 x_2^4 + a_1^2 a_2^5 x_1^2 x_2^5 + a_1 a_2^6 x_1 x_2^6 + a_2^7 x_2^7 + 1 = 0$$

We view the products of a_i as variables, and x_i as coefficients. In other words, we have the coefficients in the order:

$$x_1^7, x_1^6 x_2, x_1^5 x_2^2, x_1^4 x_2^3, x_1^3 x_2^4, x_1^2 x_2^5, x_1 x_2^6, x_2^7, 1$$

and monomials in the order:

$$a_1^7, a_1^6 a_2, a_1^5 a_2^2, a_1^4 a_2^3, a_1^3 a_2^4, a_1^2 a_2^5, a_1 a_2^6, a_2^7, 1$$

If we evaluate these coefficients at the signatures, we get $(n+1)^k$ vectors which will be the rows of the following matrix:

$$\begin{bmatrix}
1 & w^4 & w & w^5 & w^2 & w^6 & w^3 & 1 & 1 \\
1 & w^3 & w^6 & w^2 & w^5 & w & w^4 & 1 & 1 \\
1 & w^5 & w^3 & w & w^6 & w^4 & w^2 & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & w^5 & w^3 & w & w^6 & w^4 & w^2 & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & w^5 & w^3 & w & w^6 & w^4 & w^2 & 1 & 1 \\
1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & 1 & 1 \\
1 & w^4 & w & w^5 & w^2 & w^6 & w^3 & 1 & 1 \\
1 & w^3 & w^6 & w^2 & w^5 & w & w^4 & 1 & 1 \\
1 & w^3 & w^6 & w^2 & w^5 & w & w^4 & 1 & 1 \\
1 & w^5 & w^3 & w & w^6 & w^4 & w^2 & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
1 & w^6 & w^5 & w^4 & w^3 & w^2 & w & 1 & 1 \\
1 & w^4 & w & w^5 & w^2 & w^6 & w^3 & 1 & 1 \\
1 & w^3 & w^6 & w^2 & w^5 & w & w^4 & 1 & 1 \\
1 & w^5 & w^3 & w & w^6 & w^4 & w^2 & 1 & 1
\end{bmatrix}$$

We apply echelon form on this matrix and then remove the zero rows. The new matrix is:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & w^5 & 0 & w^4 \\
0 & 0 & 1 & 0 & 0 & 0 & w^2 & 0 & w^6 \\
0 & 0 & 0 & 1 & 0 & 0 & w^4 & 0 & w^5 \\
0 & 0 & 0 & 0 & 1 & 0 & w^3 & 0 & w \\
0 & 0 & 0 & 0 & 0 & 1 & w^6 & 0 & w^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}$$

Our next goal is to turn this matrix back to polynomials. Recall the order of the monomials, we get 7 multivariate polynomials:

$$\begin{aligned}
&a_1^7 + 1\\
&a_1^6 a_2 + w^5 a_1 a_2^6 + w^4\\
&a_1^5 a_2^2 + w^2 a_1 a_2^6 + w^6\\
&a_1^4 a_2^3 + w^4 a_1 a_2^6 + w^5\\
&a_1^3 a_2^4 + w^3 a_1 a_2^6 + w\\
&a_1^2 a_2^5 + w^6 a_1 a_2^6 + w^2\\
&a_2^7 + 1
\end{aligned}$$

The first and last polynomials do not help, they are trivial. Remember that we are not looking for the original values for a_i, we only need solutions for a_i up to unit multiple. Therefore, we can set $a_1 = 1$, and if we pick the second polynomial, we then get a univariate polynomial $w^5 a_2^6 + a_2 + w^4$. The roots are $a_2 = 1$ and $a_2 = w^5$.

Let us check our solution with the linear map $\mathscr{T} = \begin{bmatrix} w^2 & w^2 \\ w^3 & w \end{bmatrix}$. It is clear that $a_1 = 1$ and $a_2 = 1$ are unit multiples of $a_1 = w^2$ and $a_2 = w^2$. Now if we check the second row, The original values are:

$$\begin{aligned}
a_1 &= w^3\\
a_2 &= w
\end{aligned}$$

If we multiply the inverse of w^3 by w, we get w^{-2} which is exactly equal to w^5 in the Galois field of 2^3 elements.

Bibliography

[1] Martin R Albrecht, Gregory V Bard, and Clément Pernet. Efficient dense gaussian elimination over the finite field with two elements. *arXiv preprint arXiv:1111.6549*, 2011.

[2] Ward Beullens, Alan Szepieniec, Frederik Vercauteren, and Bart Preneel. Luov: Signature scheme proposal for nist pqc project. 2017.

[3] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.

[4] Jintai Ding, Christopher Wolf, and Bo-Yin Yang. l-invertible cycles for multivariate quadratic. *Public Key Cryptography–PKC 2007*, page 266.

[5] Jean-Guillaume Dumas and Clément Pernet. Computational linear algebra over finite fields. *arXiv preprint arXiv:1204.3735*, 2012.

[6] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 206–222. Springer, 1999.

[7] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In *Annual International Cryptology Conference*, pages 257–266. Springer, 1998.

[8] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 419–453. Springer, 1988.

[9] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2017.

[10] Jacques Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In *Annual International Cryptology Conference*, pages 248–261. Springer, 1995.

[11] Jacques Patarin. The oil and vinegar algorithm for signatures. In *Dagstuhl Workshop on Cryptography, 1997*, 1997.

[12] Park Shim and Kim. Himq-3: A high speed signature scheme based on multivariate quadratic equations. 2017.

[13] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[14] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[15] V Vassilevska Williams. Breaking the coppersmith-winograd barrier. *E-mail address: jml@ math. tamu. edu*, 2011.