Shifting our knowledge of MQ-Sign security

Lars Ran[1] and Monika Trimoska[2]

[1] Radboud Universiteit, Nijmegen, The Netherlands
[2] Eindhoven University of Technology, The Netherlands
`lran@cs.ru.nl, m.trimoska@tue.nl`

Abstract. Unbalanced Oil and Vinegar (UOV) is one of the oldest, simplest, and most studied ad-hoc multivariate signature schemes. UOV signature schemes are attractive because they have very small signatures and fast verification. On the downside, they have large public and secret keys. As a result, variations of the traditional UOV scheme are usually developed with the goal to reduce the key sizes. Seven variants of UOV were submitted to the additional call for digital signatures by NIST, prior to which, a variant named MQ-Sign was submitted to the (South) Korean post-quantum cryptography competition (KpqC). MQ-Sign is currently competing in the second round of KpqC with two variants. One of the variants corresponds to the classic description of UOV with certain implementation and parameter choices. In the other variant, called MQ-Sign-LR, a part of the central map is constructed from row shifts of a single matrix. This design makes for smaller secret keys, and in the case where the equivalent keys optimization is used, it also leads to smaller public keys. However, we show in this work that the polynomial systems arising from an algebraic attack have a specific structure that can be exploited. Specifically, we are able to find preimages for d-periodic targets under the public map with a probability of 63% for all security levels. The complexity of finding these preimages, as well as the fraction of d-periodic target increases with d and hence provides a trade-off. We show that for all security levels one can choose $d = \frac{v}{2}$, for v the number of vinegar variables, and reduce the security claim. Our experiments show practical running times for lower d ranging from 6 seconds to 14 hours.

1 Introduction

The quest for cryptographic schemes that are secure even against attackers equipped with a large-scale quantum computer is in full tilt, and a substantial portion of the current focus is on developing secure digital signature schemes. Even though digital signatures are not concerned by the store-now-decrypt-later attack, it is still of great urgency to find practical solutions that replace the non-post-quantum signature schemes that are currently deployed. For certain

This research has been supported by the Dutch government through the NWO grant OCNW.M.21.193 (ALPaQCa).

applications, for instance issuing signatures for documents that have a long validity period, replacing the signatures once a quantum computer is available can not bring back confidence in the validity of the signature. Furthermore, many embedded devices are designed to have a long shelf-life and simply replacing the cryptosystem in the future is not a viable option for these use-cases either. This effort is further motivated by the various worldwide standardization initiatives. The additional call for digital signatures by NIST recently came to the start of its second round, where 14 candidates were selected to advance through the selection process [16]. Another initiative that aims at encouraging the development and advancing the field of post-quantum cryptography is the Korean post-quantum Cryptography (KpqC) project [21]. Promoting innovative research and broadening the field being the foremost goal of the KpqC project is also symbolized in the acronym, where the lowercase letters 'p' and 'q' put together resemble the Chinese character for 'door'. Most prominently, the KpqC project is responsible for organizing the KpqC competition that started in 2022, calling for submissions of post-quantum Key Encapsulation Mechanisms (KEMs) and post-quantum digital signatures. The competition is currently moving towards the end of round 2 and the work in this paper analyzes one of the candidates that is competing for the win. An analysis on all round 1 candidates can be found in [7].

Multivariate digital signatures based on the trapdoor construction are an appealing candidate solution because they commonly yield small signatures and fast verification running times. On the other hand, they come with huge public and secret keys. Hence, optimizations of signature schemes that follow this paradigm focus primarily on reducing these sizes. One way to accomplish this is public key compression, a method where the design of the system offers the possibility of having a part of the public map generated from a *seed*. Employing such a method is not always possible, and depends on the concrete construction of the cryptosystem. Another approach is to incorporate an additional structure to the secret/public quadratic maps that allows us to generate them from a smaller input. This second approach also needs a novel design, but moreover, it requires extensive security analysis to ensure that the additional structure does not render the problem of inverting the public maps easier, which would implicitly put in jeopardy the security of the cryptosystem. The latter approach is explored in MQ-Sign, a digital signature scheme that was submitted to the KpqC competition. The initial idea rested on *sparseness*, i.e. having only few nonzero entries in the central map. This design was attacked first in [2] and shortly after in [13], which lead to the idea being abandoned and replaced by a different approach in the round 2 proposal of MQ-Sign. In round 2, a new variant of MQ-Sign, called MQ-Sign-LR, was introduced. This variant uses linear forms to generate a part of the central map, which makes it significantly more compact. In this work, we take a closer look at the underlying structure and the security of this new variant.

Our contributions. In this work we analyse the security claim of the newly proposed MQ-Sign variant called MQ-Sign-LR. As a first contribution, we give an alternative description of the cyclic systems appearing in the public key of this variant by using row-shift matrices. Then, armed with this reformulation, we observe that we can make quadratic maps coincide by using periodic input. This leads to our discovery of *weak targets*, namely periodic targets, of which we can find preimages faster. Furthermore, we show that we are able to practically find preimages for a substantial subset of them, with running times ranging from 6 seconds to 14 hours. The fact that these weak targets exists and are practically invertible for all security levels of MQ-Sign-LR is already a security concern.

Our second contribution is turning these findings into a universal forgery attack that challenges the security claims for all security levels. More precisely, we show that the class of weak targets is big enough for an attacker to enumerate salts until a weak target is found.

Our third contribution is isolating a hardness assumption, the Shifted MQ problem, $SMQ(n, q)$, underlying this attack, and providing a novel method for solving it. The proposed algorithm is a variant of FXL, whereby care is taken in which variables, or actually linear constraints in this case, are fixed. It turns out that, fixing one linear constraint can lead to two quadratic equations devolving into linear equations, hence simplifying the system considerably. We show that this can be done once per distinct nth root of unity in $\mathbb{F}_q$.

Finally, we back up all our claims by running and documenting an extensive array of experiments.

2 Background

Let $\mathbb{F}_q$ be the finite field of q elements. We use bold letters to denote vectors, e.g. $\mathbf{x}, \mathbf{t}$, and matrices, e.g. $\mathbf{P}, \mathbf{F}$. We write (column) vectors as $\mathbf{a} = (a_1, \ldots, a_n)$ and the entries of a vector $\mathbf{a}$ are denoted by a_i, or, if $\mathbf{a}$ itself has a subscript, e.g. $\mathbf{a}_b$, by $(a_b)_i$. When we use indices in superscript they will always be enclosed in parentheses, for example $\mathbf{P}^{(i)}$.

2.1 Quadratic forms

Let $p(x_1, \ldots, x_n) = \sum_{1 \leqslant i \leqslant j \leqslant n} \gamma_{ij} x_i x_j$ be a quadratic form over $\mathbb{F}_q$. Then, for fields of odd characteristic, p can be represented by a symmetric matrix $\mathbf{P}$ where $\mathbf{P}_{ij} = \gamma_{ij}/2$ for $i \neq j$, and $\mathbf{P}_{ii} = \gamma_{ii}$. There is a one-to-one correspondence between quadratic forms and symmetric matrices, since for $\mathbf{x} = (x_1, \ldots, x_n)$ it holds that

$$p(x_1, \ldots, x_n) = \mathbf{x}^\top \mathbf{P} \mathbf{x}.$$

With this representation, all operations on quadratic forms naturally transform into operations on matrices since the one-to-one correspondence between quadratic forms and symmetric matrices is actually an isomorphism. Over fields $\mathbb{F}_q$ of even characteristic, this relation does not hold, since for a symmetric matrix $\mathbf{P}$ we have $(\mathbf{P}_{ij} + \mathbf{P}_{ji}) x_i x_j = 2\mathbf{P}_{ij} x_i x_j = 0$. Thus, in even characteristic we associate to p an upper-triangular matrix with coefficients $\mathbf{P}_{ij} = \gamma_{ij}$ for $i \leqslant j$.

With this representation, some nice computational properties of the symmetric matrix representation break down, but in practical applications this can be overcome and the upper-triangular representation is used when working with fields of even characteristic. The operation Upper is used to transform any matrix representing a quadratic form to upper-triangular form.

2.2 The UOV construction

A trapdoor construction is a combination of a function and a trapdoor whereby computing the function is easy for everyone, but inverting it is hard for parties that do not have knowledge on the trapdoor. The oil and vinegar construction, first proposed in [17], is a leading example of such a trapdoor construction. Unfortunately, the initial proposal of this construction was broken one year later in [15]. However, one more year later, Kipnis, Patarin, and Goubin in [14] proposed the Unbalanced Oil and Vinegar construction that is still considered secure to this day.

This construction can be specified as a sequence of m quadratic maps

$$\mathcal{F} = (f^{(1)}, \ldots, f^{(m)}) \in \mathbb{F}_q[x_1, \ldots, x_n]$$

in n variables. However, these maps are of a specific shape, in the sense that not all quadratic monomials are present. This shape can be described by splitting the variables into vinegar variables $(x_1, \ldots, x_v)$ and oil variables $(x_{v+1}, \ldots, x_{v+o})$ where $o = m = n - v$. Then the quadratic maps are given as

$$f^{(k)} = \sum_{\substack{1 \le i \le v \\ i \le j \le n}} \alpha_{ij}^{(k)} x_i x_j + \sum_i \beta_i^{(k)} x_i + \gamma^{(k)}. \tag{1}$$

Knowing this structure, it is easy to find preimages for any target $\mathbf{t} \in \mathbb{F}_q^m$ by fixing all vinegar variables and solving the remaining linear system. Therefore, we apply an invertible linear map $\mathcal{S}$ to define the public key $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$. This linear transformation hides the special structure of the central map and the security of UOV relies on the assumption that the public map obtained in such a way is indistinguishable from a random quadratic map.

UOV follows the standard signature generation and verification process of a trapdoor-based multivariate signature scheme. In the following, we summarize the two algorithms. For simplicity, we omit the use of the linear transformation of the output, as this is not used in UOV or MQ-Sign.

Signature Generation. To generate a signature for a message M, the signer uses a hash function $H : \{0,1\}^\star \to \mathbb{F}_q^m$ and a salt r to compute the hash value $\mathbf{t} = H(M||r) \in \mathbb{F}_q^m$, and computes recursively $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{t}) \in \mathbb{F}_q^n$, and $\mathbf{z} = \mathcal{S}^{-1}(\mathbf{y})$. The signature of the message M is $\mathbf{z} \in \mathbb{F}_q^n$. Here, $\mathcal{F}^{-1}(\mathbf{t})$ means finding one (of possibly many) preimages of $\mathbf{t}$ under the central map $\mathcal{F}$.

Verification. To check if $\mathbf{z} \in \mathbb{F}_q^n$ is indeed a valid signature for a message M, one computes $\mathbf{t} = H(M||r)$ and $\mathbf{t}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}_q^m$. If $\mathbf{t}' = \mathbf{t}$ holds, the signature is accepted, otherwise it is rejected.

For cryptographic schemes in the UOV family, the linear and constant parts of Equation (1) are usually dropped and we work only with quadratic forms. The shape of the quadratic forms that form the central map in upper-triangular form is

$$\mathbf{F}^{(k)} = \begin{pmatrix} \mathbf{F}_1^{(k)} & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{2}$$

where the $\mathbf{F}_1^{(k)}$ are upper-triangular. We use this block matrix notation to distinguish between the so-called vinegar-vinegar part and vinegar-oil part of the secret key. The matrices $\mathbf{F}_1^{(k)}$ hold the coefficients of monomials comprised of two vinegar variables, whereas the matrices $\mathbf{F}_2^{(k)}$ hold the coefficients of the monomials that are a product of one vinegar and one oil variable. Note that the lower-right corner is the zero matrix because there are no oil-oil monomials in the central map. Using this notation the public maps can be computed as

$$\mathbf{P}^{(k)} = \begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} = \mathcal{S}^T \begin{pmatrix} \mathbf{F}_1^{(k)} & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathcal{S}. \tag{3}$$

Equivalent keys. It was shown in [19] that for any instance of a UOV secret key $(\mathcal{F}', \mathcal{S})$, there exists an equivalent secret key $(\mathcal{F}, \mathbf{S})$ with

$$\mathbf{S} = \begin{pmatrix} \mathbf{I}_{v\times v} & \mathbf{S}_1 \\ \mathbf{0}_{m\times v} & \mathbf{I}_{m\times m} \end{pmatrix}. \tag{4}$$

For an attacker, it is not necessary to find the original secret key that was obtained at key generation, i.e. the one that a valid signer is in possession of. The goal is rather to find any of the equivalent keys as they can all be used to forge a signature. Since finding a secret key where $\mathbf{S}$ is of the form as in Equation (4) seems to be the least computationally intensive, we consider the complexity of this as a baseline for a key recovery attack that aims at finding the secret matrix $\mathbf{S}$. As a consequence, a key of this specific form can be used as part of the specification of the cryptosystem, in which case the matrix $\mathbf{S}$ can be stored using fewer entries. We refer to this as the equivalent-keys optimization, and this optimization is used in most modern instantiations of UOV. Note however that this optimization technique introduces vulnerabilities when we consider physical attacks [1].

2.3 The MQ-Sign digital signature scheme

MQ-Sign is a UOV-based signature scheme, where the main focus is to reduce the size of the public and secret key compared to traditional UOV. In the first-round proposal of MQ-Sign, this was achieved by using sparse polynomials for the quadratic part of the central map. In the central map, we distinguish two main parts: the vinegar-vinegar part, which consists of monomials that contain two vinegar variables, and the vinegar-oil part, comprised of the monomials containing one vinegar variable and one oil variable. There were initially four variations of the scheme: MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR and MQ-Sign-RR, where the suffix specifies, for the two parts of the quadratic maps, whether they are defined with sparse or random polynomials.

Three attacks on MQ-Sign were published during the first round of the competition, which eliminated all but the last variant of MQ-Sign, denoted MQ-Sign-RR. This is the most conservative variant as it is built on the UOV trapdoor without any additional structure. The first algebraic attack on MQ-Sign was proposed by Aulbach, Samardjiska, and Trimoska [2], and it exploits the sparseness of the vinegar-oil part of the secret key. The attack also relies on the fact that the map $\mathcal{S}$ is chosen to be given by a matrix of the form as in Equation (4). Recall that, this typically does not reduce the security of a UOV-based scheme because it was shown in [19] that for any instance of a UOV secret key $(\mathcal{F}', \mathbf{S}')$, there is an equivalent key $(\mathcal{F}, \mathbf{S})$ where $\mathbf{S}$ has the form as in (4). However, coupling this optimization technique with the specific structure of the central map in MQ-Sign yields many linear constraints that allow for a polynomial-time key recovery. The attack was fully implemented, and it was reported to run in 0.6 seconds for the proposed parameters for security level I, 2.3 seconds for security level III and 6.9 seconds for security level V.

Following this, Ikematsu, Jo, and Yasuda proposed another algebraic attack that also targets the MQ-Sign-{S/R}S variants but is not dependent on $\mathbf{S}$ having the equivalent keys structure [13]. These two attacks eliminated both variants where the vinegar-oil part of the secret key is sparse.

Another algebraic attack was proposed in [2] which targets specifically the variant where only the vinegar-vinegar part of the secret key is sparse. This attack is not practical, but shows that the security of MQ-Sign-SR does not meet the required security level and this variant is also removed from the updated submission in round 2.

MQ-Sign advanced to the second round of KpqC and the second-round submission includes the variant corresponding to traditional UOV, MQ-Sign-RR, and a new design that leads to an additional variant called MQ-Sign-LR. This new non-conservative variant of MQ-Sign has smaller secret keys, while maintaining the same signature size as MQ-Sign-RR. This variant reportedly yields better performance for both key generation and signing. Note that since the reduction of the size of the secret key is in the vinegar-vinegar part, when the equivalent-keys optimization is used, this yields a reduction in the public key size as well. This is because when $\mathbf{S}$ is of the form as in Equation (4), the vinegar-vinegar part of the public key is equal to the vinegar-vinegar part of the secret key. This will be explained in more detail in the following section.

The main difference between the two MQ-Sign variants is in the structure of the vinegar-vinegar part of the central map. In MQ-Sign-LR, the vinegar-vinegar part of the central map is constructed as a product of a circulant matrix where the entries are the vinegar variables $(x_1, \ldots, x_v)$ and a vector whose entries are linear combinations of the vinegar variables. Specifically, the central map is defined as

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_v \\ x_v & x_1 & \ldots & x_{v-1} \\ \ldots & \ldots & \ldots & \ldots \\ x_{v-m+2} & x_{v-m+3} & \ldots & x_{v-m+1} \end{pmatrix} \cdot \begin{pmatrix} L_1 \\ L_2 \\ \ldots \\ L_v \end{pmatrix}, \tag{5}$$

where $L_i = \sum_{j=1}^{v} \gamma_{ij} x_j$, for $i \in \{1, \ldots, v\}$ and each row of the product matrix gives a polynomial in $\mathcal{F}$. As a result, the vinegar-vinegar part of the central map can be represented with v^2 field elements[3], instead of the $\frac{v^2 m}{2}$ field elements that are required in the MQ-Sign-RR variant.

For reference for the rest of the paper, we recall here the parameters of MQ-Sign for all security levels. The parameter choices are the same between the two variants MQ-Sign-LR and MQ-Sign-RR. They are chosen such that the cryptosystem resists all generic attacks againts the UOV family.

Table 1. The parameters of MQ-Sign.

Level	q	v	m
I	2^8	72	46
III	2^8	112	72
V	2^8	148	96

2.4 Multivariate system solving

A problem that comes up naturally in analyzing such UOV-like systems is that of solving multivariate (quadratic) systems. In these problems, one is presented with a system, $\mathcal{F} = (f_1, \ldots, f_m)$, consisting of m equations in n variables over $\mathbb{F}_q$ and one would like to find a solution to the system. In this work, we will only consider such problems with $m \geq n$, also known as (over-)determined systems. Current state-of-the-art algorithms for computing such solutions are, for example, F4 [10], F5 [11], XL, and FXL [8]. All of these algorithms are extensions of the Buchberger algorithm [3]. These algorithms have much the same underlying theory for deriving their asymptotic complexity. We give a simple overview here, focusing on the complexity of XL as this algorithm behaves most predicatively.

The central parameter indicating the complexity is the solving degree d_{solv}. This is the minimum degree d for which the rowspace [6] of the Macaulay matrix

[3] The submission counts $v(v+m)$ elements, which corresponds to the evaluation costs, but not to the number of stored elements.

of degree d contains the Gröbner basis of the system (w.r.t. the *grevlex* monomial order). Computing the solving degree is difficult in general, but it can be upper-bounded by other relevant invariants [6], including the degree of regularity d_{reg} as shown in [20]. Following [4, 12], the degree of regularity is defined by

$$d_{reg} = \min\{d \geq 0 \mid (\mathcal{F}^{top})_d = R_d\},$$

where $\mathcal{F}^{top}$ is the top homogenous part of our system, R_d are all monomials of degree d, and $(\mathcal{F}^{top})_d = (\mathcal{F}) \cap R_d$. Now, if $\deg f_i \leq d_{solv}$ for all i, then one obtains the bound

$$d_{solv} \leq d_{reg} + 1.$$

In the case of a semi-regular sequence of quadratic equations, we can actually compute the degree of regularity as the first non-positive coefficient of the Hilbert series

$$\frac{(1-t^2)^m}{(1-t)^n}.$$

However, for structured systems, these estimations may not hold.

Once the solving degree of the system is determined, the complexity of the XL algorithm is given as

$$\mathcal{C}_{XL(n,m,q)} = 3\binom{n+d_{solv}}{d_{solv}}^2 \binom{n+2}{2} \left(\log_2(q)^2 + \log_2(q)\right).$$

This is the cost of applying the block-Wiedemann algorithm on a matrix of size $\binom{n+d_{solv}}{d_{solv}}$, with density $\binom{n+2}{2}$ and a field operation cost of $\left(\log_2(q)^2 + \log_2(q)\right)$.

Then, the FXL algorithm improves on this by fixing g variables first. Choosing the correct g is part of a trade-off and depends on n, m, q. Note that d_{solv} is now generally dependent on g. The complexity of the algorithm is now given by

$$\mathcal{C}_{FXL(n,m,q)} = \min_{g \leq n} q^g \cdot \mathcal{C}_{XL(n-g,m,q)}.$$

3 A forgery attack

We show in this section how a forgery attack can be mounted against MQ-Sign-LR. The attack relies on the specific structure arising from the new design, as well as on the use of the equivalent keys optimization. Recall that when this optimization is used, the secret matrix $\mathbf{S}$ is of the form as in Equation (4), in which case the matrices representing the public and the secret map share a common block. Following the notation in this paper, the common block is the upper-left block that corresponds to the vinegar-vinegar part of the maps. This can be observed from the equation defining the computation of the public key

$$\begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(k)} & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

which in upper-triangular matrix form simplifies to

$$\begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1^{(k)} & (\mathbf{F}_1^{(k)} + \mathbf{F}_1^{(k)\top})\mathbf{S}_1 + \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathsf{Upper}(\mathbf{S}_1^\top \mathbf{F}_1^{(k)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(k)}) \end{pmatrix}.$$

In the following, we show how an attacker can forge a signature of a message M, which consists in finding a preimage of $\mathbf{t} = H(M||r)$ under the public map $\mathcal{P}$. We will write in block matrix representation since we are using the specific structure of the upper-left block of the public map. The goal is to find a vector $(\mathbf{x}_v, \mathbf{x}_m) \in \mathbb{F}_q^n$ such that

$$\begin{pmatrix} \mathbf{x}_v & \mathbf{x}_m \end{pmatrix} \begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_v \\ \mathbf{x}_m \end{pmatrix} = \mathbf{x}_v^T \mathbf{P}_1^{(k)} \mathbf{x}_v + \mathbf{x}_v^T \mathbf{P}_2^{(k)} \mathbf{x}_m + \mathbf{x}_m^T \mathbf{P}_4^{(k)} \mathbf{x}_m = t_k$$

holds for every $k \in \{1, \dots, m\}$. Note here that $\mathbf{P}_1^{(k)}$ has the same shape as $\mathbf{F}_1^{(k)}$ because of the equivalent keys form of $\mathbf{S}$, and we will use this structure in the attack. Let us denote by $V(\mathbf{t})$ the variety (the set of solutions) of the ideal generated by $\mathcal{P}(\mathbf{x}) = \mathbf{t}$. Since we have m equations in n variables with $n > m$, we can afford to add another $v = n - m$ affine constraints and still expect, heuristically, to have a solution. We use only m (recall that $m < v$) of those to eliminate the non-structured part of this system. Specifically, we assign all variables in $\mathbf{x}_m$ to zero. We are then left with the system

$$\mathbf{x}_v^T \mathbf{P}_1^{(k)} \mathbf{x}_v = t_k \tag{6}$$

that contains only the block $\mathbf{P}_1^{(k)}$, and thus inherits the structure depicted in Equation (5).

3.1 Quadratic maps in MQ-Sign-LR

Since our focus is on solving polynomial systems arising from the constraint in Equation (6), we first take a closer look into the structure of these systems. Recall that the polynomials in this system are obtained as in Equation (5). Hence, we have the following system of equations

$$\begin{aligned} &x_1 L_1 + x_2 L_2 + \dots + x_v L_v = t_1, \\ &x_1 L_2 + x_2 L_3 + \dots + x_v L_1 = t_2, \\ &\dots \\ &x_1 L_m + x_2 L_{m+1} + \dots + x_v L_{m-1} = t_m. \end{aligned} \tag{7}$$

Let us denote by σ the mapping that sends L_i to its coordinate vector, i.e.

$$\sigma: \quad L_i = \sum_{j=1}^{v} \gamma_{ij} x_j \mapsto \sigma(L_i) = (\gamma_{i1}, \dots, \gamma_{iv}).$$

When we rewrite the equations from (7) in matrix form we obtain

$$\begin{aligned}
&\mathbf{x}_v^\top \mathbf{P}_1^{(1)} \mathbf{x}_v = t_1, \\
&\mathbf{x}_v^\top \mathbf{P}_1^{(2)} \mathbf{x}_v = t_2, \\
&\dots \\
&\mathbf{x}_v^\top \mathbf{P}_1^{(m)} \mathbf{x}_v = t_m,
\end{aligned}$$

where

$$\mathbf{P}_1^{(1)} = \begin{pmatrix} \sigma(L_1) \\ \sigma(L_2) \\ \dots \\ \sigma(L_v) \end{pmatrix}, \quad \mathbf{P}_1^{(2)} = \begin{pmatrix} \sigma(L_2) \\ \sigma(L_3) \\ \dots \\ \sigma(L_1) \end{pmatrix}, \quad \dots \quad , \quad \mathbf{P}_1^{(m)} = \begin{pmatrix} \sigma(L_m) \\ \sigma(L_{m+1}) \\ \dots \\ \sigma(L_{m-1}) \end{pmatrix}. \tag{8}$$

This is because the i-th row of the matrix contains the coefficients of all monomials containing x_i, which for the k-th polynomial, is exactly given by the coordinates of $L_{i+k \bmod v}$. Note that these matrices are not the canonical representation of the polynomials, as they are not symmetric or upper-triangular. Nevertheless, they are a valid representation of the corresponding polynomials, since we have that the sum of the entry $\{i, j\}$ and the entry $\{j, i\}$ is equal to the coefficient of the monomial $x_i x_j$. We will keep this non-canonical representation, as it facilitates the exposition.

From Equation (8), it is easy to spot that the matrices representing these quadratic forms are equal up to cyclic row-shifts. Specifically, we have that $\mathbf{P}_1^{(k+1)}$ is obtained from a cyclic upward shift of the rows of $\mathbf{P}_1^{(k)}$. Let us denote by $\mathbf{T}$ the matrix representing the permutation corresponding to a cyclic upward row shift. Then, $\mathbf{T}^i$ represents a cyclic upward row shift by i steps. The system of equations that we aim to solve takes the following form

$$\begin{aligned}
&\mathbf{x}_v^\top \mathbf{P}_1^{(1)} \mathbf{x}_v = t_1, \\
&\mathbf{x}_v^\top \mathbf{T} \mathbf{P}_1^{(1)} \mathbf{x}_v = t_2, \\
&\dots \\
&\mathbf{x}_v^\top \mathbf{T}^{m-1} \mathbf{P}_1^{(1)} \mathbf{x}_v = t_m.
\end{aligned} \tag{9}$$

Since we can now express all equations in terms of $\mathbf{P}_1^{(1)}$, we will write $\mathbf{P} = \mathbf{P}_1^{(1)}$.

3.2 Computing preimages of weak targets

The specific structure of the quadratic maps described in the previous section allows us to find preimages of certain target vectors $\mathbf{t}$ more easily. We will refer to these vectors as weak targets. The reason it is easier to find preimages for these weak targets is that, because of the cyclic structure of the maps, we can construct our input vectors in such a way that they fulfill some constraints by design. To explain this idea, we will start by looking at the homogenous case, i.e. finding preimages of $\mathbf{0}$. The vector $\mathbf{0}$ is indeed one of the weak targets.

We start again from the observation that in the system in Equation (9), every subsequent matrix $\mathbf{P}_1^{(k)}$ is obtained by a cyclic row-shift of the previous one. Equivalently, we can imagine that we have the same matrix m times, but the permutation is on the $\mathbf{x}_v^\top$ vectors on the left side, performing cyclic shifts on vector entries (a permutation on columns of $\mathbf{x}_v^\top$, since $\mathbf{T}$ acts on the right). It is then evident that for vectors $\mathbf{x}_v$ where all the entries are equal to each other, if $\mathbf{x}_v$ is a solution to the first equation, then $\mathbf{x}_v$ is a solution to the entire system. We generalize this observation to other specific vectors that have a repeating subsequence.

Let us denote by $\sim$ the binary relation on $\mathbb{F}_q^v$ described informally as "$\mathbf{a}$ is equal to $\mathbf{b}$ up to a cyclic right-shift (without loss of generality)". This is indeed an equivalence relation because (i) $\mathbf{a} \sim \mathbf{a}$ by a shift of zero, (ii) if $\mathbf{a}$ is obtained by performing a right-shift of k on $\mathbf{b}$, then $\mathbf{b}$ can be obtained by a right-shift of $v-k$ on $\mathbf{a}$, and (iii) if $\mathbf{a}$ is a k-shift away from $\mathbf{b}$ and $\mathbf{b}$ is an l-shift away from $\mathbf{c}$, then $\mathbf{a}$ is a $((k+l) \bmod v)$-shift away from $\mathbf{c}$. We further know that, for a given v, the number and size of such equivalence classes in $\mathbb{F}_q^v$ can be derived by looking at the divisors of v. For each divisor d of v, we have up to q^d equivalence classes of size d. We now make the following observation for the system in Equation (7). If $\mathbf{x}$ belongs to an equivalence class of size d and $\mathbf{x}$ is a solution to the first d equations in (7), then $\mathbf{x}$ is a solution to the entire system. This observation tells us that the system does not behave like a *random* system and that there are some vectors that are probabilistically more likely to be a solution to the system than others, as they only need to satisfy a subset of the equations. We exploit this by looking for such solutions using the following strategy.

For each divisor d of v, excluding v [4] and taken in ascending order, build a smaller system by taking the first d equations of the initial system in Equation (7) and replacing the unknown $\mathbf{x}_v = (x_1, \ldots, x_v)$ by

$$\mathbf{x} = (x_1, \ldots, x_d, x_1, \ldots, x_d, \ldots, x_1, \ldots, x_d).$$

This is a quadratic system of d equations in d variables. Denote $\mathbf{x}_d = (x_1, \ldots, x_d)$. To concretely describe the structure, we introduce the matrices

$$\mathbf{J}_{d,n} = (I_d, \ldots, I_d) \in \mathcal{M}^{d\times n}(\mathbb{F}_q)$$

for $d \mid n$. For simplicity, we define $\mathbf{J}_{n,d} = \mathbf{J}_{d,n}^T$. This allows us to write

$$\mathbf{x} = (x_1, \ldots, x_d, x_1, \ldots, x_d, \ldots, x_1, \ldots, x_d) = \mathbf{J}_{n,d}(x_1, \ldots, x_d) = \mathbf{J}_{n,d}\mathbf{x}_d.$$

Also note how this commutes with the cyclic shift matrices

$$\mathbf{T}_n^i \mathbf{J}_{n,d} = \mathbf{J}_{n,d}\mathbf{T}_d^i = \mathbf{J}_{n,d}\mathbf{T}_d^{i \bmod d}.$$

[4] In fact, we also need $d < m$. However, for the systems considered, this holds for all divisors d of v except v itself.

Now let us take a closer look at the equations that we obtain if we look for solutions $\mathbf{x}$ that map to $\mathbf{0}$

$$\begin{aligned} 0 &= \mathbf{x}^\top \mathbf{T}^{i-1}\mathbf{P}\mathbf{x} \\ &= \mathbf{x}_d^\top \mathbf{J}_{d,v}\mathbf{T}^{i-1}\mathbf{P}\mathbf{J}_{v,d}\mathbf{x}_d \\ &= \mathbf{x}_d^\top \mathbf{T}^{i-1}\mathbf{J}_{d,v}\mathbf{P}\mathbf{J}_{v,d}\mathbf{x}_d. \end{aligned} \tag{10}$$

Since these type of equations are central in the rest of this work we will define the following problem to aid the discussion.

Problem 1. SMQ(n, q) (Shifted MQ problem)
Let $\mathbf{P} \in \mathcal{M}^{n\times n}(\mathbb{F}_q)$ be a matrix and let $\mathbf{t} \in \mathbb{F}_q^m$ be a vector. The *Shifted MQ problem* asks to find — if any — a solution $\mathbf{x} \in \mathbb{F}_q^n$ to the following system of equations

$$\mathbf{x}^\top \mathbf{T}^{i-1}\mathbf{P}\mathbf{x} = t_i \quad \text{for all} \quad 1 \le i \le n. \tag{11}$$

Here, $\mathbf{P}$ is called the initial matrix and $\mathbf{t}$ the target vector.

Now, Equation (10) depicts exactly the shifted MQ problem $SMQ(d,q)$ with initial matrix $\mathbf{J}_{d,v}\mathbf{P}\mathbf{J}_{v,d}$ and target vector $\mathbf{t} = \mathbf{0}$.

Remark 1. With this attack, we are able to find one (or a few) out of many elements in $V(\mathbf{0})$. The secret oil subspace, denoted O, is contained in $V(\mathbf{0})$ and in the case that the obtained vector is part of the oil subspace this would lead to a full key recovery. It is well known from the reconciliation attack [9] that finding the first oil vector is the bottleneck of recovering the secret oil space, and recently it was shown, first in [1] and then in [18], that once we have found the first oil vector, the remaining steps to recover the entire oil space can be done in polynomial time. However, since $V(\mathbf{0})$ is of dimension $n-m$ and O is of dimension m, the probability that the vector we obtain is in O is negligibly small, concretely q^{-n+2m}. Hence, we conclude that this forgery attack does not lead to a key-recovery attack.

Periodic targets. The reason why the above discussion worked so well is because $\mathbf{0}$ is periodic. In fact, it is even 1-periodic. We can extend the above algorithm to any periodic $\mathbf{t}$. Let $d' \mid v$ with $d' \le m$ and let $\mathbf{t}$ be a d'-periodic vector, in other words $t_i = t_{i+d'}$ for all $1 \le i \le m-d'$. Then for any d with $d' \mid d$ and $d \mid v$ we can apply the same trick and look for d-periodic solutions $\mathbf{x} = (\mathbf{x}_d, \ldots, \mathbf{x}_d)$. Building the same equations as above, we obtain the following system of equations

$$\mathbf{x}_d^\top \mathbf{T}^{i-1}\mathbf{J}_{d,v}\mathbf{P}\mathbf{J}_{v,d}\mathbf{x}_d = t_i \quad \text{for all} \quad 1 \le i \le d. \tag{12}$$

This is again the $SMQ(d,q)$ problem with initial matrix $\mathbf{J}_{d',v}\mathbf{P}\mathbf{J}_{v,d'}$, but now with a target vector $\mathbf{t} = (t_1, \ldots, t_d)$.

3.3 Forging a signature

Given that we can find solutions to quite a bit of weak targets, the question is if we can build a signature forgery attack from these findings. In this section, we respond in the affirmative, showing that we can sign any message using this technique, without any knowledge of the secret key. For the MQ-Sign-LR parameter sets, this can be done with a complexity that breaks the security claims for all three levels of security.

Given a message M and a salt r the probability that $\mathbf{t} = H(M||r)$ is d-periodic is quite small. In fact, this probability is given by q^{d-m}. However, as an attacker, we can simply keep on choosing random salts r until we find a hash that is d-periodic. For example, for level I, with $d = v/2 = 36$ and $m = 46$, we only need to resample the salt 2^{80} times on average. Note that the salt is 32 bytes, so for a cryptographic hash function, we will certainly be able to find a good one. Note however that having a d-periodic target is not the only requirement. We also need the corresponding $SMQ(d, q)$ problem to have a solution. In contrast to the signing procedure where the common approach is to resample the chosen values for the v arbitrarily assigned variables, our attack requires that the variables in $\mathbf{x}_m$ are fixed to zero and the rest are chosen as described in the previous section. Let $p_{d,q}$ be the probability that an $SMQ(d, q)$ problem has a solution, and we denote by $\mathcal{C}_{SMQ(d,q)}$ and $\mathcal{C}_H$ the complexity of solving an $SMQ(d, q)$ instance and of computing one hash respectively. We can compute the complexity of forging a signature as

$$\min_{d|v, d<m} p_{d,q}^{-1}(\mathcal{C}_{SMQ(d,q)} + q^{m-d}\mathcal{C}_H). \tag{13}$$

In Section 5 we find that, empirically, $p_{d,q}$ is 0.63 and this value is largely independent of d and $q = 2^r$. Furthermore, it turns out that for all parameter sets, the only viable d would be $v/2$. For other d values, the cost of sampling salts would dominate the costs and surpass the security requirement. For this choice of divisor, the amount of salts to generate in the different levels is given in Table 2. This estimation is computed as $q^{m-v/2}$, as the probability of having a $v/2$-periodic hash is $q^{v/2-m}$.

Table 2. The average number of salts to try before finding a $v/2$-periodic hash.

Level	q	v	m	salts
I	2^8	72	46	2^{80}
III	2^8	112	72	2^{128}
V	2^8	148	96	2^{176}

4 Solving the SMQ problem

As we saw in the last section, the bottleneck of the forgery attack is based on finding solutions to the SMQ problem for random $\mathbf{t}$. In this section, we turn to tacking this problem and perform an initial analysis of our proposed solutions. To use the usual complexity estimates for algorithms in the XL family, we have to find the solving degree. Normally we would assume that our system behaves as a semi-regular system and we would compute the solving degree from that assumption, as explained in the background section. However, given the amount of structure that we observe in the system this seems unlikely. Nevertheless, as we will see in Section 5, this assumption still seems to be a good estimator for the solving degree in our experiments. Hence, in this paper, all of the theoretical estimates on the solving degree are done under this counterintuitive assumption. Coupled with extensive experimental work, these estimations are used to show the relevance of the attack, but they should not be used, for instance, for determining security parameters. Indeed, further research is needed to confirm the theoretical findings, or find more precise bounds on the solving degree for instances of the SMQ problem.

4.1 Solving $SMQ(n, q)$ using FXL

Since the problem at hand consists of a multivariate quadratic system of n equations in n variables, we are going to approach this using a Gröbner basis algorithm. With these parameters, $m \approx n$ and $q = 2^8$, employing a guessing strategy improves the complexity of solving the system. The FXL algorithm [8] is an algorithm that exactly fills that role. The resulting complexities can be found in the left column of Table 3 for several values of n. This is a standard approach and the background section contains further explanations on how these complexities are computed.

Table 3. The theoretical complexity of solving $SMQ(n, 256)$ using FXL with and without improved guessing. The ordinary enumeration due to FXL is denoted by g, whereas the number of guesses made with the improved guessing strategy is denoted by k.

	FXL		**Improved guessing FXL**	
n	$(g,\, d_{solv})$	$\log_2$ cost	$(k,\, g,\, d_{solv})$	$\log_2$ cost
24	(1, 14)	89	(3, 0, 8)	77
36	(2, 17)	121	(3, 0, 13)	108
56	(3, 24)	174	(1, 1, 26)	171
74	(3, 32)	220	(1, 3, 29)	215

4.2 An improved guessing strategy

Due to the structure of the problem, it turns out that our guessing can be made more effective than usual. Let $\zeta \in \mathbb{F}_q$ be an nth root of unity in $\mathbb{F}_q$. Define the vectors $\mathbf{v}_\zeta, \bar{\mathbf{v}}_\zeta \in \mathbb{F}_q^n$ as $(v_\zeta)_i = \zeta^i$ and $(\bar{v}_\zeta)_i = \zeta^{-i}$. Then we can make the following observation

$$\sum_i \zeta^i \mathbf{T}^i = \bar{\mathbf{v}}_\zeta \mathbf{v}_\zeta^\top .$$

Now we can take and manipulate the following linear equations

$$\begin{aligned}\sum_i \zeta^i t_{i+1} &= \sum_i \zeta^i \mathbf{x}^\top \mathbf{T}^i \mathbf{P} \mathbf{x} \\ &= \mathbf{x}^\top \left(\sum_i \zeta^i \mathbf{T}^i \right) \mathbf{P} \mathbf{x} \\ &= \mathbf{x}^\top \bar{\mathbf{v}}_\zeta \mathbf{v}_\zeta^\top \mathbf{P} \mathbf{x} \\ &= \left(\sum_i \zeta^{-i} x_i \right) \cdot \left(\sum_{i,j} \zeta^i \mathbf{P}_{ij} x_j \right) .\end{aligned}$$

Example 1. Pick $\zeta = 1$. Then we get the following equation

$$\sum_i t_i = \left(\sum_i x_i \right) \cdot \left(\sum_i (\mathbf{P}\mathbf{x})_i \right) .$$

If we now guess the constraint $\sum_i \zeta^{-i} x_i = \gamma \in \mathbb{F}_q^*$, then we get the linear constraint $\sum_i \zeta^i (\mathbf{P}\mathbf{x})_i = \gamma^{-1} \sum_i \zeta^i t_{i+1}$ for free. To be more precise, using this guess, we were able to turn a quadratic equation into a linear equation. A common technique when we have a linear equation is to trade it to remove one variable. For instance, an equation $\gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n + \gamma_0 = 0$ is rewritten as $x_1 = -\gamma_1^{-1}(\gamma_2 x_2 + \cdots + \gamma_n x_n + \gamma_0)$ and used to substitute x_1 in the system. Hence, when using k different such roots of unity, we get a system of $n - k$ equations in $n - 2k$ variables using k guesses.

Something interesting happens when $2 \mid n$ in even characteristic. We define the vectors $\mathbf{w}$ and $\bar{\mathbf{w}}$ by

$$(w_\zeta)_i = \begin{cases} \zeta^i & \text{if } i \equiv 0 \mod 2 \\ 0 & \text{if } i \equiv 1 \mod 2 \end{cases} \qquad (\bar{w}_\zeta)_i = \begin{cases} \zeta^{-i} & \text{if } i \equiv 0 \mod 2 \\ 0 & \text{if } i \equiv 1 \mod 2. \end{cases}$$

In that case we have the following observation:

$$\sum_{i \equiv 1 \mod 2} \zeta^i \mathbf{T}^i = \bar{\mathbf{w}}_\zeta \mathbf{v}_\zeta^\top + \bar{\mathbf{v}}_\zeta \mathbf{w}_\zeta^\top . \tag{14}$$

So now if we guess $\mathbf{x}^\top \bar{\mathbf{v}}_\zeta = \gamma \in \mathbb{F}_q^*$ with induced $\mathbf{v}_\zeta^\top \mathbf{P}\mathbf{x} = \beta$ we get:

$$\sum_{i \equiv 1 \mod 2} \zeta^i t_{i+1} = \sum_{i \equiv 1 \mod 2} \zeta^i \mathbf{x}^\top \mathbf{T}^i \mathbf{P}\mathbf{x} \tag{15}$$
$$= \mathbf{x}^\top \left(\bar{\mathbf{w}}_\zeta \mathbf{v}_\zeta^\top + \bar{\mathbf{v}}_\zeta \mathbf{w}_\zeta^\top \right) \mathbf{P}\mathbf{x} \tag{16}$$
$$= \mathbf{x}^\top \bar{\mathbf{w}}_\zeta \mathbf{v}_\zeta^\top \mathbf{P}\mathbf{x} + \mathbf{x}^\top \bar{\mathbf{v}}_\zeta \mathbf{w}_\zeta^\top \mathbf{P}\mathbf{x} \tag{17}$$
$$= \mathbf{x}^\top \bar{\mathbf{w}}_\zeta \beta + \gamma \mathbf{w}_\zeta^\top \mathbf{P}\mathbf{x} \tag{18}$$

And this is a linear constraint again! A consequence is that in this case, we can guess k constraints (up to the amount of distinct nth roots of unity) to obtain a quadratic system of $n - 2k$ equations in $n - 3k$ variables.

Remark 2. One might worry that the guessed constraints and the induced constraints are linearly dependent, or worse yet, inconsistent. However, experiments point out that this is not the case with high probability if $n \gg 3k$ (or $n \gg 2k$ when n or q is odd).

Given the above solving strategy, we want to compute the complexity of solving the remaining system using a Gröbner basis approach. We use again the assumption that the systems (after guessing some variables) behave as semi-regular systems. Since $\mathbb{F}_q$ has at most $\gcd(q-1, n)$ distinct nth roots of unity, we can guess only that many variables in the way described above. Therefore, for some n we would instead like to guess g additional variables for the best trade-off in the FXL algorithm. Then, we can compute the complexity, for even n, as

$$\min_{k \leq \gcd(q-1,n)} q^k \cdot \mathcal{C}_{FXL(n-3k, n-2k, q)}. \tag{19}$$

Here $\mathcal{C}_{FXL(n,m,q)}$ is the cost of FXL over $\mathbb{F}_q$ with m quadratic equations in n variables. The summary of these results can be found in Table 3.

4.3 Complexity of forging signatures for MQ-Sign-LR

Now, almost everything is in place to compute the complexity of forging a signature for MQ-Sign-LR. We will assume that $p_{d,q} = 0.63$ as found experimentally in Section 5. Furthermore, we will assume that $\mathcal{C}_H \leq 2^{30}$. Generally, for cryptographic hashes, this is a huge overestimation. However, for this attack, picking this bound, makes sure that solving SMQ dominates the complexity. The complexities that result can be found in Table 4. Recall that the solving degree is computed under the assumption that the systems are random enough to use the theory developed for semi-regular systems.

5 Experiments

As we saw in Section 3, we are interested in the probability that a random SMQ system has a solution. Furthermore, our analysis on the solving complexity of SMQ is based on some heuristics. Therefore, we also provide experimental evidence backing up our claims.

Table 4. The theoretical complexity of forging MQ-Sign-LR signatures.

Level	q	v	m	$\log_2$ cost
I	256	72	46	108
III	256	112	72	172
V	256	148	96	216

5.1 Probability of SMQ having a solution

First, let us consider the probability of a random SMQ system to be solvable. As explained in Section 3, when faced with the problem of finding a preimage of a vector $\mathbf{t}$, the setup of the attack does not allow for any choice of the assignment of the variables in $\mathbf{x}_m$. If the corresponding polynomial system does not have a solution, the only option an attacker has is to search for another weak target $\mathbf{t}$. As a result, the probability of an SMQ instance having a solution intervenes in the asymptotic complexity of the attack, and we need to obtain an estimate of this probability.

As the systems are structured, we can not use existing theoretical analysis to derive the probability, and thus we take an experimental approach. We generate and solve many instances of the SMQ problem for different values of n and for q fixed as in the parameters of MQ-Sign. We find empirically that such an SMQ instance has a 0.63 chance to have a solution, independent of $n \geq 3$. The first row in Table 5 shows the number of runs that we performed for a given n, and the second row shows the derived probability, averaged over all runs. The results

Table 5. Probability of a random $SMQ(n, 256)$ system having a solution.

	5	6	7	8	9	10	11	12
Experiments	10^6	10^6	10^6	10^6	10^5	10^4	10^4	2000
Probability	0.630	0.630	0.631	0.631	0.631	0.623	0.628	0.633

were consistent, and we use this empirically obtained probability to calculate the overall complexity of the forgery attack.

5.2 Solving SMQ

Now let us turn to experiments of actually solving SMQ systems. We first aim to confirm our theoretical findings in Section 4, focusing on the solving degree of such systems. As we noted before, these systems are not semi-regular. However, we can still use the Hilbert series estimation to predict the degree of regularity of such systems as if they were semi-regular. Recall that this bounds the solving degree of such systems.

Now we do some experimental work to determine if and how far off our estimations are. In these experiments we generate a random SMQ system and solve it using MAGMA's [5] `GroebnerBasis()`. We denote as d_{MAGMA} the highest degree reached in the computation of `GroebnerBasis()`. Following the notation from Section 4, the experiment was done for several values of n and k, and in all experiments we set $g = 0$. We chose to limit ourselves to $1 \leq k \leq 3$ and n that are divisors of the different v in MQ-Sign-LR. Note that we still require $k \leq \gcd(n, q-1)$. The results can be found in Table 6.

Table 6. The solving degrees for SMQ instances with different n and k, and $g = 0$. The reported degrees are formatted as d_{MAGMA}/d_{reg}. The degree d_{reg} is computed under the assumption that the systems behave as semi-regular.

n	7	8	9	12	14	16	18	24
$k = 1$	4/4	4/4	5/5	6/6	7/7	7/8	9/9	—
$k = 2$	—	—	4/4	4/4	—	—	7/7	8/9
$k = 3$	—	—	3/3	3/3	—	—	5/5	6/7

Examining Table 6, we see that in all experiments, d_{MAGMA} was equal or lower than the predicted degree of regularity. The lower values can be explained by the extra structure that is present in the system, even after the improved guessing of linear constraints. For example, in these specific cases we found k extra degree falls at degree 3. The propagation of these degree falls might have led to a lower observed degree d_{MAGMA}. It would be interesting to better understand, theoretically, how the actual solving degree behaves. However, practically, these experiments suggest that algorithms for solving these systems have a similar or lower complexity than anticipated. In other words, the proposed attacks might be faster than expected in some instances.

Finally, we turn to solving some instances of weak targets in practice, and we derive average running times[5]. Our experiments include both weak instances that can be solved in practical time on our machine and weak instances of the scale required for mounting a universal forgery attack. For more precise timing estimates, we opted for simulating a single enumeration step of our algorithm by fixing the variables that are supposed to be enumerated to an arbitrary guess. We then account for the enumeration step by multiplying with the corresponding value. Since the enumeration step has predictable complexity, this approach gives us the closest estimate to real running times. Similarly, for practical reasons, the choice of number of variables to enumerate does not necessarily coincide with the choice that yields the optimal trade-off asymptotically.

Again, we use MAGMA's `GroebnerBasis()` to solve the randomly generated instances. For the values of d for which this was infeasible, we report the time for solving a system with higher k and g than optimal. Note that, especially in

[5] All timing experiments were run on an AMD EPYC 7502P.

these cases, being able to pick a lower g would decrease the overall time of the algorithm. Table 7 shows running times for forging signatures having d-periodic hashes, essentially finding preimages of weak targets. We choose d to be a divisor of $v = 72$, so that these experiments correspond exactly to the weak targets of MQ-Sign-LR parameters intended for the first security level. For reference, we also included the fraction of such hashes occurring for the level I parameters. This is an indication of the expected number of times we need to re-salt and hash until we find a weak target. Specifically, the last row contains the period for which it is feasible to perform a universal forgery in time less than the time required to reach the first security level. Interestingly, in the $d = 36$ experiment, the observed degree d_{MAGMA} is equal to 8, which also lines up with the estimated degree of regularity of a semi-regular system with similar parameters.

Table 7. Running times for forging signatures having d-periodic hashes. The last column is the fraction of weak hashes and is computed as q^{m-d} for parameters $q = 2^8$ and $m = 46$. †Expected running time.

		Enumeration	Time		
d	(k, g)	Expected iterations	Single	Full†	Fraction (Level I)
12	(1, 0)	2^7	0.05 s	6.4 s	2^{-272}
18	(2, 0)	2^{15}	1.56 s	14.2 h	2^{-224}
24	(3, 0)	2^{23}	5.94 s	1.6 y	2^{-176}
36	(3, 5)	2^{63}	4.35 h	10^{16} y	2^{-80}

These experiments show that we are able to find preimages of a considerable fraction of weak targets in up to 14.2 hours for the parameters of MQ-Sign-LR. For these parameters where it was practical (up to $d = 18$), we were also able to do the full forgery.

6 Countermeasures

The universal forgery attack proposed in this paper has an exponential time complexity. In fact, Table 4 shows how far the current MQ-Sign-LR parameters are from reaching the required security level. Considering only the theoretical complexity of the forgery attack, it might be tempting to increase the parameter choices for all three security levels so that the system reaches the corresponding security requirements. However, this is not a sufficient countermeasure, as we have shown that the system suffers from other vulnerabilities like the many weak targets that we detect in Section 3. This means that if MQ-Sign-LR is deployed in real world applications, a verifier can never accept signatures that are build from a weak target, as the preimages of those can be computed in faster running times, sometimes even seconds, as shown in Section 5.

The design of the system needs to be modified in such a way that our attack is countered and the existence of weak targets is completely eliminated. Looking at the requirements that we define in Section 3 for a vector being a weak target, a straightforward countermeasure for this attack would be to choose parameters such that v is a prime number. Note that it is also possible to adjust the parameter v in such a way that the inverse of the probability of hashing into a weak target surpasses the security threshold. For parameters of the scale of MQ-Sign for instance, it would be enough to take v odd, as even if the greatest divisor is $v/3$, the probability to find a $v/3$-periodic hash is negligibly low. We do not propose this countermeasure because of the same reasons exposed in the previous paragraph. A verifier must never accept such weak targets and hence a different solution that involves substantially changing the specification of the protocol behind the trapdoor construction needs to be developed. This in turn evokes a substantial study on the provable security analysis of the system. We conclude that choosing v prime is the only countermeasure that successfully counters the vulnerabilities found in this work.

Since our attack also relies on the equivalent-keys optimization, excluding its use would protect against this concrete attack. However, further analysis is required to gain confidence in the security of using the cyclic structure of the central map. Indeed, the work in this paper also highlights that the underlying hardness assumptions of MQ-Sign-LR are substantially different than for classic UOV. Furthermore, in this case only the secret key benefits from reduced sizes, whereas for most use cases, it is more advantageous to reduce the public key size.

Acknowledgement. We thank Simona Samardjiska for initial discussions on the attack idea. We thank Magali Bardet for meticulously reviewing our work and for many helpful comments and suggestions.

References

[1] T. Aulbach, F. Campos, J. Krämer, S. Samardjiska, and M. Stöttinger. Separating oil and vinegar with a single trace side-channel assisted kipnis-shamir attack on UOV. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):221–245, 2023.

[2] T. Aulbach, S. Samardjiska, and M. Trimoska. Practical key-recovery attack on MQ-sign and more. In M.-J. Saarinen and D. Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 168–185. Springer, Cham, June 2024.

[3] B. B. Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal. *Ph. D. Thesis, Math. Inst., University of Innsbruck*, 1965.

[4] M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* Theses, Université Pierre et Marie Curie - Paris VI, Dec. 2004.

[5] W. Bosma, J. Cannon, and C. Playoust. The Magma Algebra System. I. The User Language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[6] A. Caminata and E. Gorla. Solving degree, last fall degree, and related invariants. *Journal of Symbolic Computation*, 114:322–335, 2023.

[7] J. Cottaar, K. Hövelmanns, A. Hülsing, T. Lange, M. Mahzoun, A. Pellegrini, A. Ravagnani, S. Schäge, M. Trimoska, and B. de Weger. Report on evaluation of KpqC candidates. Cryptology ePrint Archive, Report 2023/1853, 2023.

[8] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, Berlin, Heidelberg, May 2000.

[9] J. Ding, B.-Y. Yang, C.-H. O. Chen, M.-S. Chen, and C.-M. Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, editors, *ACNS 08: 6th International Conference on Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pages 242–257. Springer, Berlin, Heidelberg, June 2008.

[10] J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[11] J. C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, page 75–83, New York, NY, USA, 2002. Association for Computing Machinery.

[12] J.-C. Faugère, M. Bardet, and B. Salvy. On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. 2004.

[13] Y. Ikematsu, H. Jo, and T. Yasuda. A security analysis on MQ-Sign. Cryptology ePrint Archive, Paper 2023/581, 2023. `https://eprint.iacr.org/2023/581`.

[14] A. Kipnis, J. Patarin, and L. Goubin. Unbalanced Oil and Vinegar signature schemes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, Berlin, Heidelberg, May 1999.

[15] A. Kipnis and A. Shamir. Cryptanalysis of the Oil & Vinegar signature scheme. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Springer, Berlin, Heidelberg, Aug. 1998.

[16] NIST. *Post-Quantum Cryptography: Additional Digital Signature Schemes. Round 2 Additional Signatures*, 2024. `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

[17] J. Patarin. The oil and vinegar signature scheme. Dagstuhl Workshop on Cryptography, 1997.

[18] P. Pébereau. One vector to rule them all: Key recovery from one vector in UOV schemes. In M.-J. Saarinen and D. Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 92–108. Springer, Cham, June 2024.

[19] A. Petzoldt. *Selecting and reducing key sizes for multivariate cryptography*. PhD thesis, Darmstadt University of Technology, Germany, 2013.

[20] F. Salizzoni. An upper bound for the solving degree in terms of the degree of regularity, 2023.

[21] The KpqC project. *Korean post-quantum Cryptography*, 2022. `https://www.kpqc.or.kr/`.