An Estimator for the Hardness of the MQ Problem

Emanuele Bellini[1], Rusydi H. Makarim[1], Carlo Sanna[2], and Javier Verbel[1]

[1] Technology Innovation Institute, UAE
[2] Department of Mathematical Sciences, Politecnico di Torino, Torino, IT

{emanuele.bellini, rusydi.makarim, javier.verbel}@tii.ae
carlo.sanna.dev@gmail.com

Abstract. The Multivariate Quadratic (MQ) problem consists in finding the solutions of a given system of m quadratic equations in n unknowns over a finite field, and it is an NP-complete problem of fundamental importance in computer science. In particular, the security of some cryptosystems against the so-called algebraic attacks is usually given by the hardness of this problem. Many algorithms to solve the MQ problem have been proposed and studied. Estimating precisely the complexity of all these algorithms is crucial to set secure parameters for a cryptosystem. This work collects and presents the most important classical algorithms and the estimates of their computational complexities. Moreover, it describes a software that we wrote and that makes possible to estimate the hardness of a given instance of the MQ problem.

Keywords: MQ problem, Estimator, Polynomial solving, multivariate cryptography.

1 Introduction

The problem of solving a multivariate quadratic system over a finite field is known as the Multivariate Quadratic (MQ) problem. This problem is known to be NP-complete, and it seems to be hard on average for an extensive range of parameters.

Despite of its clear hardness, there exists a considerable amount of algorithms to solve the MQ problem [17, 18, 33, 34, 38, 39, 53, 55, 57, 67]. Their complexities depend on several values: the ratio of the number of variables and the number of polynomials, the size of the field, the characteristic of the field, and the number of solutions to the underlying problem. So it is difficult to determine what is the complexity of the best algorithm to solve a particular instance of the MQ problem. Moreover, some of the algorithms have optimized some parameters to provide the best asymptotic complexity. Since this does not mean that such optimizations provide the best complexity for a particular set of parameters, once a specific instance is provided, the parameters optimizing the complexity should be computed.

The MQ problem has been extensively used in cryptography. In cryptanalysis, it appears in the so-called *algebraic attacks*. These attacks break the security of a cryptosystem by solving one system of polynomial equations over a finite field (e.g., see [65]). In post-quantum cryptography, the MQ problem constitutes the building block of the Multivariate public key cryptosystems (MPKCs). These are promising post-quantum secure alternatives to the current public-key schemes, which would no longer be hard in the presence of a quantum computer [11].

In general, the MPKCs used for signature schemes are divided into two categories, named trapdoor and one-way multivariate signature schemes. The trapdoor ones are built upon a trapdoor multivariate polynomial map. They have very short signatures and fast signature verification that can be implemented in low-cost devices [21, 30]. Some examples of this kind of schemes are UOV [52], Rainbow [31] GeMSS [21], MAYO [16]. The one-way signature schemes are based on an identification scheme that uses the knowledge of a solution of a random multivariate polynomial map to authenticate a legitimate party (e.g., [63]). Then, the multivariate identification scheme is converted into a signature scheme using a standard protocol [42]. The main advantage of this kind of multivariate signature scheme is that their security is based (directly) on the hardness of the MQ problem, and they enjoy a small compressed public key. Some of the these schemes are SOFIA [22], MQDSS [23], MUDFISH [13], and the signature proposed by Furue, Duong and Takagi in [44].

In the design of MPKCs, once the specific parameters of the scheme are selected, the size of the finite field q, the number of variables n, and the number of polynomials m of the corresponding MQ problem are fixed. Thus, a good estimate of the hardness of solving such an instance with given parameters q, n, and m of the MQ problem will allow the designers to set parameters to get efficient schemes while keeping high levels of security. Also, MPKCs are potentially strong candidates for the upcoming NIST call for proposals to standardize a new signature scheme that is not based on structural lattices [58].

Our first contribution is to gather the algorithms available in the literature for solving the MQ problem and expressing their estimates of time and space complexities as a function of their parameters if any. For the Crossbred algorithm, we establish a more tightly bound that takes into account the field equations when $q > 2$. Our second contribution is to provide a software named the MQ estimator [9], that given an instance of the MQ problem estimates the complexities of each algorithm to solve it. We use our software to estimate the best algorithm, in terms of time complexity, for different regimes of parameters, see Figures 1a, 1c, 1b and 1d. Also, we estimate the security of the multivariate schemes Rainbow, MAYO, MUDFISH, and MQDSS against the direct attack. We found that all the parameter sets of MAYO, MUDFISH, MQDSS and the category I parameters of Rainbow are under the claimed security, see table 2, 3 and 4.

Our paper is somehow analog to the work that Albrecht, Player, and Scott did for the problem Learning with Errors [1], and the work by Bellini and Esser for the Syndrome Decoding problem [10, 37]. Regarding the problem of solving polynomial systems over an arbitrary field, we refer the reader to the book edited

by Dickenstein and Emiris [29] and to the survey of Ayad [4]. For collections of algorithms for the case of finite fields, we point out the Ph.D. theses of Bard [6], Mou [59], and Ullah [68]. Also, there are surveys on very specific techniques like, for instance, signature-based algorithms for computing Gröbner bases [36].

The structure of the paper is as follows. Section 2 shows some preliminary facts about computational complexity and the MQ problem. In Section 3 we list the main algorithms for solving the MQ problem, and we examine their time and space complexities. Also, we establish a generalization over $\mathbb{F}_{q\geqslant 2}$ to compute the complexity of the Crossbred algorithm. Finally, in Section 5 the usage of the MQ estimator is explained, and some of the estimates obtained are shown.

2 Preliminaries

2.1 General notation

We employ Landau's notation $f(n) = \mathcal{O}(g(n))$, with its usual meaning that $|f(n)| \leqslant C|g(n)|$ for some constant $C > 0$. Also, we write $f(n) = \tilde{\mathcal{O}}(g(n))$ whenever $f(n) = \mathcal{O}(n^k g(n))$ for some constant $k \geqslant 0$. For every real number x, let $\lfloor x \rfloor$ be the greatest integer not exceeding x. We let log denote the logarithm in base 2, while $e = 2.718\ldots$ denotes the Euler number. The Greek letter ω denotes the exponent in the complexity of matrix multiplication. That is, we assume that there is an algorithm that multiplies two $n \times n$ matrices over a field with $\mathcal{O}(n^\omega)$ field operations. We have $2 \leqslant \omega \leqslant 3$, with $\omega = 2.80736$ given by Strassen's algorithm [66] (currently, the best upper bound is $\omega < 2.37286$, given by the algorithm of Alman and Williams [2], but such improvements of Strassen's algorithm are never used in practice, due to the large hidden constants of their asymptotic complexities). We write $\mathbb{F}_q$ for a finite field of q elements. Lastly, we use n_{sol} to denote the number of solutions of a given MQ instance.

2.2 Computational complexity

Throughout this paper, the time and space complexities of each algorithm are given, assuming a computational model in which the operations of $\mathbb{F}_q$ (addition, multiplication, and division) are performed in constant time $\mathcal{O}(1)$ and in which every element of $\mathbb{F}_q$ is stored in constant space $\mathcal{O}(1)$ ($\mathbb{F}_q$-complexity). A more detailed analysis could assume a computational model in which the operations at bit-level are performed in constant time $\mathcal{O}(1)$ and in which every bit is stored in constant space $\mathcal{O}(1)$ (bit-complexity). However, the bit-complexities of addition, multiplication, and division in $\mathbb{F}_q$ are different (with the addition the least expensive and division the most) and depend on the algorithms implementing them, possible hardware optimizations, and eventually, q having a special form, like q being equal to a power of 2 or a Mersenne prime (see [46] for a survey). Therefore, there is no straightforward way to convert between $\mathbb{F}_q$-time complexity and bit-time complexity. Roughly, the bit-time complexity can be estimated by $(\log q)^\theta$ times the $\mathbb{F}_q$-time complexity, with $\theta \in [1, 2]$. On the other hand, the bit-space complexity is simply equal to $\log q$ times the $\mathbb{F}_q$-space complexity. Of course, for $q = 2$ the $\mathbb{F}_q$-complexity and the bit-complexity are equivalent.

2.3 The MQ problem

The input of the MQ problem consists of m quadratic polynomials $p_1, \ldots, p_m$ in n variables $x_1, \ldots, x_n$ and coefficients in a finite field $\mathbb{F}_q$. The output concerns the solutions (in $\mathbb{F}_q$) of the system of equations

$$p_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m. \tag{1}$$

There are three versions of the MQ problem:

- **Decision:** It asks to determine if (1) has a solution.
- **Search**: It asks to find a solution of (1), if there is any.

The decision version of MQ is known to be NP-complete. Moreover, the decision and search version of MQ are strictly related. On the one hand, obviously, solving the search version also solves the decision. On the other hand, by iteratively testing each variable, one can solve the search version by calling a subroutine for the decision version at most $(q-1)n$ times. In this paper, we focus only on the search version of the problem.

2.4 General strategies for underdetermined systems

A system of equations is said to be *underdetermined* if it has more unknowns than equations, that is, in our notation, $n > m$. There are algorithms that transform an underdetermined system into one with the same number of equations and variables so that a solution to the former system can efficiently be computed from a solution to the last system. This section describes some of these algorithms.

Fixing variables This approach consists in fixing the values of $n-m$ unknowns. The resulting systems $\mathcal{Q}$ has m unknowns and m equations and, assuming that we started from a random MQ problem, it has a solution with probability $e^{-1} = 0.367\ldots$ [45]. Thus, the complexity of this strategy is dominated by the complexity of solving $\mathcal{Q}$.

Thomae–Wolf and improvements Let P be an MQ problem with m equations in $n = \alpha m$ unknowns over $\mathbb{F}_q$, where $\alpha > 1$ is a rational number. Based on the ideas of Kipnis, Patarin, and Goubin [53], Thomae and Wolf [67] designed a mechanism to reduce P to another MQ problem $\mathcal{P}'$ of $m - \lfloor \alpha \rfloor + 1$ equations and variables. The time complexity of the reduction is $\mathcal{O}\left(m(\lfloor \alpha \rfloor m)^3\right)$ when q is even. For q odd, the complexity of the reduction is dominated by the complexity of quadratic system with $\lfloor \alpha \rfloor - 1$ equations over $\mathbb{F}_q$. Still, for the rage of parameters interesting in practice, e.g., see Table 2, and 3 the complexity of the reduction is insignificant compared with the complexity of solving the problem $\mathcal{P}'$.

This technique was further improved by Furue, Nakamura, and Takagi [43], who combined Thomae and Wolf's approach with fixing k variables. Their method reduces a system of m quadratic equations in n unknowns over $\mathbb{F}_{2^r}$ to a systems of $m - \alpha_k$ quadratic equations in $m - k - \alpha_k$ unknowns, where $\alpha_k := \lfloor (n-k)/(m-k) \rfloor - 1$ for $r > 1$ and $\alpha_k := \lfloor (n-k)/(m-k-1) \rfloor - 1$ for $r = 1$.

3 Algorithms for solving MQ

This section lists the most important classical algorithms for solving the MQ problem and describes their computational complexities. There are also quantum algorithms to solve the MQ problem, which speed up some of the classical algorithms via quantum walks, but they are out of the scope of this paper. We refer the reader to [12,40,64] for more information on quantum algorithms solving the MQ problem.

The complexity of some of the algorithms described in this section depends on the number of solutions n_{sol} of the underlying instance. We remark that for the non-underdetermined case, hard instances of the MQ problem are expected to have a small n_{sol} (e.g., 1, 2, or 3). In the other case, n_{sol} is expected to be around q^{m-n}.

3.1 Exhaustive search

Fast exhaustive search (FES) Bouillaguet et al. [18] proposed a more efficient way to perform exhaustive search over $\mathbb{F}_2^n$ by enumerating the solution space with Gray codes. The evaluation of an element in $\mathbb{F}_2^n$ is used to compute the evaluation of the next potential solution. The time complexity of finding one solution to the MQ problem with this algorithm is given by $4\log(n)\left(\frac{2^n}{n_{sol}+1}\right)$. The space complexity is given by the memory required to store the polynomial system, so $\mathcal{O}(n^2m)$. This approach can be extended to any field $\mathbb{F}_q$ using q-ary Gray codes [35]. In this case, the time complexity is given by $\mathcal{O}\left(\log_q(n)\frac{q^n}{n_{sol}+1}\right)$.

3.2 Algorithms designed for underdetermined systems

In this section, we describe some of the algorithms specially designed for underdetermined systems.

Kipnis–Patarin–Goubin (KPG) Kipnis, Patarin, and Goubin [53] proposed a polynomial-time algorithm to solve the MQ problem over a field of even characteristic when $n > m(m+1)$. We write KPG to refer to this algorithm. The main idea in the KPG algorithm is to find a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{n\times n}$ such that the change of variables $(y_1,\ldots,y_n)^\top = \mathbf{S}(x_1,\ldots,x_n)^\top$ results in a system of the form

$$\begin{aligned}
&\sum_{i=1}^{m} a_{i,1}y_i^2 + \sum_{i=1}^{m} y_i L_{i,1}(y_{m+1},\ldots,y_n) + Q_1(y_{m+1},\ldots,y_n) = 0 \qquad (2)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots\\
&\sum_{i=1}^{m} a_{i,m}y_i^2 + \sum_{i=1}^{m} y_i L_{i,m}(y_{m+1},\ldots,y_n) + Q_m(y_{m+1},\ldots,y_n) = 0,
\end{aligned}$$

where the $L_{i,j}(\cdot)$ are linear maps and the $Q_i(\cdot)$ are quadratic maps. Then, they solve (2) by solving, sequentially, two systems of linear equations in the y_i's. The overall time complexity of this algorithm is given by

$$\mathcal{O}\left(\sum_{i=1}^{m-1}(im)^\omega + m^\omega + m^{2\omega} + 3n^\omega\right) = \mathcal{O}\left(mn^\omega\right).$$

The space complexity is dominated either by the space needed to solve a square linear system of size $(m-1)m$ plus the space required to store the original polynomials. It is given by $\mathcal{O}(mn^2)$.

Courtois et al. applied the ideas of KPG over fields of odd characteristic. They provided an algorithm with time complexity $2^{40}(40+40/\log q)^{m/40}$ for $n \geqslant (40+40/\log q)^{m/40}(m+1)$, see [26, Sec. 4.2] for the details.

Miura–Hashimoto–Takagi (MHT) The algorithm of Miura, Hashimoto, and Takagi [57] works for $n \geqslant m(m+3)/2$ over any finite field. Its complexity is exponential in m over fields of odd characteristic. Precisely, its time complexity is

$$\begin{cases} \mathcal{O}\left(n^\omega m\right) & \text{if } q \text{ is even;} \\ \mathcal{O}\left(2^m n^\omega m\right) & \text{if } q \text{ is odd.} \end{cases}$$

The space complexity of MHT is dominated by the memory required to store the initial set of polynomials.

Huang–Bao (HB) In [48], Huang and Bao propose an algorithm that generalizes the MHT algorithm, and it works the broader range of parameters $n \geqslant m(m+1)/2$. The space complexity is the same of MHT, while the time complexity is given by

$$\begin{cases} \mathcal{O}\left(q(\log q)^2 \cdot n^\omega m\right) & \text{if } q \text{ is even;} \\ \mathcal{O}\left(q(\log q)^2 \cdot 2^m n^\omega m\right) & \text{if } q \text{ is odd.} \end{cases}$$

Courtois–Goubin–Meier–Tacier (CGMT-A) Courtois et al. [26] introduced an algorithm, so-called Algorithm A, to solve the MQ problem in the underdetermined case. Here we use CGMT-A to refer to this algorithm.

Let k be an integer, and let $f_1, \ldots, f_m$ be the input polynomials of the underlying MQ problem. For each $j = 1, \ldots, m$, we can write

$$f_j(x_1, \ldots, x_n) = g_j(x_1, \ldots, x_k) + h_j(x_{k+1}, \ldots, x_n) + \sum_{i=1}^{k} L_{i,j}(x_{k+1}, \ldots, x_n)x_i,$$

where the $L_{i,j}$'s are linear polynomials. In the CGMT-A algorithm one has to set $L_{i,j}(x_{k+1}, \ldots, x_n) = c_{i,j}$, where each $c_{i,j} \in \mathbb{F}_q$ is randomly chosen. It is also defined $g_j'(x_1, \ldots, x_k) := g_j(x_1, \ldots, x_k) + \sum_{i=1}^{k} c_{i,j}x_i$ for $j = 1, \ldots, 2k$.

Then, one has to compute and store (sorted) the q^k vectors of the set $\mathcal{G} = \{-(g'_1(a), \ldots, g'_{2k}(a)) : a \in \mathbb{F}_q^k\}$ along with its corresponding preimage a. After, we have to search for a vector $b \in \mathbb{F}_q^{n-k}$ such that

$$\{L_{i,j}(b) = c_{i,j}\}_{i,j=0}^{k,2k} \text{ and } (h_1(b), \ldots, h_{2k}(b)) \in \mathcal{G}. \tag{3}$$

Notice that any b be fulfilling Equation (3) yields to a vector $c := (a, b) \in \mathbb{F}_q^b$ such that $f_j(c) = 0$ for $j = 1, \ldots, 2k$, and $f_j(c) = 0$ for $j = 2k+1, \ldots, m$ with probability $q^{-(m-2k)}$.

Courtois et al. noticed that in order to find one b satisfying Equation (3) one has to compute on average q^k evaluations of the form $(h_1(b), \ldots, h_{2k}(b))$ for vectors b satisfying $\{L_{i,j}(b) = c_{ij}\}_{i,j=0}^{k,2k}$.

If $k := \min\lfloor m/2, \sqrt{n/2 - \sqrt{n/2}}\rfloor$ satisfies that $m - 2k < 2k^2 \leqslant n - 2k$, then CGMT-A succeed on finding a solution, and its average time complexity is given by

$$\mathcal{O}\left(2k\binom{n-k}{2}q^{m-k}\right).$$

The space complexity of this algorithm is dominated by the memory needed to store the q^k evaluations of the g'_i polynomials, i.e., $\mathcal{O}(2kq^k)$.

3.3 Gröbner basis

This section shows the complexity to solve the MQ problem via the so-called *Gröbner basis algorithms*. Throughout this section, we assume that the underlying sequence of polynomials is either *regular* or *semi-regular*, see Definition 3.2.

A solution to the MQ problem $f_1 = \cdots = f_m = 0$ can be efficiently extracted from a Gröbner basis G_{lex}, in the lexicographic order, of the ideal $I = \langle f_1, \ldots, f_m \rangle$ [28]. In practice, it is been verified that the most efficient way to compute G_{lex} is by first computing a Gröbner basis G_{grvlex} of I in the graded reverse lexicographic order. Then, one uses G_{grvlex} as an input of the FGLM [41] algorithm to compute G_{lex} [38].

The complexity of the Gröbner basis algorithms in the graded reverse lexicographic order is estimated via the *degree of regularity* of the ideal generated by the polynomial of the underlying MQ problem.

Definition 3.1. *The degree of regularity of a homogeneous ideal $I \subseteq \mathbb{F}_q[\boldsymbol{x}]$ is the minimum integer d, if any, such that $\dim(I_d) = \dim(R_d)$, where $I_d = R_d \cap I$, R_d is the set of elements in R of degree d.*

Definition 3.2 (Regular and semi-regular sequences). *A homogeneous sequence $\mathcal{F} \in \mathbb{F}_q[\boldsymbol{x}]^m$ is called semi-regular if*

$$\sum_{d \geqslant 0} \dim\left(R_d/I_d\right) z^d = \left[\frac{(1-z^q)^n}{(1-z)^n}\left(\frac{1-z^2}{1-z^{2q}}\right)^m\right]_+,$$

where $[H(z)]_+$ means that the series $H(z)$ is cut from the first non-positive coefficient. An affine sequence $\mathcal{G} = (g_1, \ldots, g_m)$ is semi-regular if the sequence $(\tilde{g}_1, \ldots, \tilde{g}_m)$ does, where $\tilde{g}_i$ is the homogeneous part of g_i of highest degree.

The role of the field equations In the ring $\mathbb{F}_q[x_1, \ldots, x_n]$, the set of elements $\{x_1^q - x_1, \ldots, x_n^q - x_n\}$ is called the *field equations*. By Lagrange's Theorem, every element in $\mathbb{F}_q$ is a root of the univariate polynomial $x^q - x$. Thus, the set of solutions of a given MQ problem (as defined in 2.3) does not change if we add the field equations to the original set of polynomials.

In terms of computing a Gröbner basis, adding the field equations to a set of polynomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ is equivalent to considering the polynomials in the quotient ring

$$R = \mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle.$$

Given a positive integer d, let R_d and $\mathbb{F}_q[x_1, \ldots, x_n]_d$ denote the $\mathbb{F}_q$-vector spaces of homogeneous polynomials of degree d in R and $\mathbb{F}_q[x_1, \ldots, x_n]$, respectively. Hence, $\dim(R_d)$ represent the number of monomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ of degree d such that no variable is raised to a power greater than $q - 1$. This value is essential in to determine the complexity of computing a Gröbner basis of an ideal $I \subset R$. The following proposition shows a way to compute $\dim(R_d)$.

Proposition 3.1. *Let d be a positive integer. Then, $dim(R_d) = [z^d]H(z)$, where $H(z)$ is the series defined by*

$$H(z) := \left(\frac{1 - z^q}{1 - z}\right)^n.$$

That is, $dim(R_d)$ is the coefficient of z^d in the series $H(z)$. Similarly,

$$\dim\left(\mathbb{F}_q[x_1, \ldots, x_n]_d\right) = [z^d]\left(\frac{1}{(1 - z)^n}\right),$$

or equivalently, $\dim\left(\mathbb{F}_q[x_1, \ldots, x_n]_d\right) = \binom{n+d-1}{d}$.

Let $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal. For small values of q, the computation of a Gröbner basis of I can be significantly easier if I is seen into the ring R. This is because, for a given integer $d > 0$, R_d could be significantly smaller than $\mathbb{F}_q[x_1, \ldots, x_n]_d$. For instance see Example 3.1.

Example 3.1. Let $q = 2$ and $n = 10$. Then,

$$\left(\frac{1 - z^q}{1 - z}\right)^n = 1 + 10z + 45z^2 + 120z^3 + 210z^4 + 252z^5 + \cdots,$$

and

$$\frac{1}{(1 - z)^n} = 1 + 10z + 55z^2 + 220z^3 + 715z^4 + 2002z^5 + \cdots.$$

Thus, $\dim\left(F[x_1, \ldots, x_n]_5\right) = 2002$, while $\dim(R_5) = 252$.

F4/F5 This section regards the complexity of solving the MQ problem via the F4 and F5 algorithms. All the estimations are computed over the hypothesis that the underlying sequences of polynomials are regular or semi-regular. The monomial order in the underlying polynomial ring is the graded reverse lexicographic (grevlex) monomial order.

The F4 and F5 are algorithms to compute a Gröbner basis of an ideal of polynomial equations over an arbitrary field. They were introduced by Faugère [38,39] and they are based on the original ideas of Lazard [54]. These two algorithms could be seen as generalizations of the XL [27] and the Mutant-XL algorithms [20].

It might be the case that one of the algorithms outperforms the other for a particular set of parameters, but it is slower for a different set of parameters. For instance, F5 is expected to be faster than F4 for underdetermined systems, but this is not the case for overdetermined systems [39]. Even though such an improvement is estimated to be no more than a factor of two for large values of n [32]. Asymptotically, the time and memory complexities of both algorithms are the same. Thus, in this paper and in our estimator, we use the F5's complexities to refer also to the complexities of F4.

The square case Bardet, Faugère, and Salvy [7] estimated the complexity of computing a Gröbner basis of a homogeneous ideal of polynomials in the grevlex order. To estimate such complexity for a non-homogenous ideal I, we use the fact that a Gröbner basis of I can be obtained by specializing to $h = 1$ in every polynomial in $\mathcal{G}^{(h)}$, where $\mathcal{G}^{(h)}$ is Gröbner basis of $I^{(h)}$, and $I^{(h)}$ denotes the homogenization of I.

By Theorem 2 in [7] with $\ell = 1$, computing a Gröbner basis in the grevlex order of a non-homogeneous ideal I can be done, without using the field equations, in

$$\mathcal{O}\left((n+1)2^{4.29(n+1)} + n \cdot n_{sol}^3\right) = \tilde{\mathcal{O}}\left(2^{4.29n}\right)$$

in arithmetic operations in $\mathbb{F}_q$.

The overdetermined case In the overdetermined case the time complexity is given by

$$\mathcal{O}\left(\binom{n + d_{reg}}{d_{reg}}^{\omega} + n \cdot n_{sol}^3\right),$$

while the space complexity:

$$\mathcal{O}\left(\binom{n + d_{reg} - 1}{d_{reg}}^{2}\right),$$

where $d_{reg} = \deg(P(z)) + 1$, and $P(z)$ is the series given in Definition 3.2.

3.4 Hybrid algorithms

A hybrid algorithm for solving the MQ problem combines a partial exhaustive search with another procedure. In this section, we consider the complexity estimations of hybrid algorithms.

BooleanSolve/FXL The BooleanSolve algorithm and its generalization, the FXL algorithm, were proposed by Bardet et al [8] and Courtois et al. [25], respectively. The idea of both algorithms is to guess a number k of variables repeatedly and then check the consistency of the resulting MQ problem.

In BooleanSolve, such a check is done by checking the consistency a linear system of the form $\mathbf{z}\mathbf{M}_d = (0, 0, .., 1)$, where $\mathbf{M}_d$ is the Macaulay matrix [8] of the resulting system up to degree d, and d is a large enough integer. Once a set of k variables being part of a solution is found, an exhaustive search algorithm is apply on the resulting system with $n - k$ variables.

In this section, we consider the FXL as a naïve generalization of BooleanSolve. That is, after guessing k variables, the consistency of the resulting system of $n-k$ variables is checked by using the Macaulay matrix at large enough degree. Notice that this is different from what Hybrid-F5 does, where the resulting system is solved after each guess of k variables.

There are methods to check the consistency of the system of the form $\mathbf{z}\mathbf{M}_d = (0, 0, .., 1)$. One uses Gaussian elimination (known as the *deterministic* variant), and the other one uses the probabilistic Wiedemann's algorithm (known as *Las Vegas* variant). For the deterministic variant, the time complexity of this algorithm is dominated by the guessing and consistency checking parts. Thus, it is given by

$$q^k \cdot \tilde{\mathcal{O}}\left(\binom{n-k+d_{wit}}{d_{wit}}^{\omega}\right),$$

while the space complexity is

$$\mathcal{O}\left(\binom{n-k+d_{wit}-1}{d_{wit}}^{2}\right),$$

where $d_{wit} = \deg(P_k(z)) + 1$, and if we consider the field equations

$$P_k(z) = \left[\frac{(1-z^q)^{n-k}}{(1-z)^{n-k+1}}\left(\frac{1-z^2}{1-z^{2q}}\right)^m\right]_+. \tag{4}$$

While Las Vegas variant of this method has time complexity

$$q^k \cdot \mathcal{O}\left(3\binom{n-k+2}{2}\binom{n-k+d_{wit}}{d_{wit}}^{2}\right),$$

and the space complexity is computed as the number of bits needed to store the whole Macaulay matrix [3]. So that one requires

$$m \cdot w + T \cdot w \cdot \frac{\log_2 N}{\log_2 q} + N\frac{\log_2 m}{\log_2 q}$$

bits of memory, where $w := \binom{n-k+2}{2}$, $T := \binom{n-k+d_{wit}-2}{d_{wit}}$, and $N := \binom{n-k+d_{wit}}{d_{wit}}$ [60, Section 4.5.3].

[3] Every row of the Macaulay matrix can be compute on the fly, but it will introduce an overhead in the time complexity.

Hybrid-(F4/F5) Another way to apply a hybrid algorithm is just by guessing a set of k variables, and then solve the resulting system by applying whether the F4 or F5 the algorithm. In this paper, we refer to this method as the Hybrid-F5 algorithm. The complexity of this method is given by

$$q^k \cdot \mathcal{O}\left(\binom{n-k+d_{reg}}{d_{reg}}^{\omega} + nn_{sol}^3\right)$$

Memory complexity:

$$\mathcal{O}\left(\binom{n-k+d_{reg}-1}{d_{reg}}^2\right),$$

where $d_{reg} = \deg(P_k(z))+1$, and $P_k(z)$ is the polynomial shown in Equation (4).

Crossbred The Crossbred algorithm was introduced by Joux and Vitse [50]. Its complexity was initially analyzed by Chen et al. in [23], and later Duarte provided a more detailed complexity analysis by encoding the information in several bivariate series to determine the complexity [35].

For $q = 2$, Duarte used the field equations to derive formulas of the aforementioned bivariate series, but he not consider them in the case of $q > 2$. Inspired by the ideas of Duarte, we extend the complexity of the Crossbred algorithm by taking into account the field equations even when $q > 2$.

The Crossbred algorithm is parameterized by three positive integers D, d, and k, and it is divided into two steps, the *preprocessing* step and the *linearization* step.

Preprocessing step. Roughly speaking, in the preprocessing step, we work with the degree-D Macaulay matrix of the input polynomials. We perform a sequence of row operations such that we produce a set of rows with corresponding polynomials fulfilling the following property: Every specialization of the last $n-k$ variables yields to a set of polynomials of degree at most d in the first k variables.

To precisely describe how the prepossessing step works introduce some notation. Let $\mathbf{M}_D$ denote the Macaulay matrix of degree D of the input polynomials in the graded reverse lexicographic order. Let $\mathbf{M}_{D,d}$ denote the submatrix of $\mathbf{M}_D$ formed by the columns corresponding to monomials of degree greater than d in the first k variables.

Here, we have to find at least $\binom{k+d}{d}$ linearly independent vectors in the left kernel of $\mathbf{M}_{D,d}$ that are not in the left kernel of $\mathbf{M}_D$. Notice that each one of those kernel vectors can be used to produce one polynomial satisfying the property mentioned above.

Linearization step. In the linearization step, the Crossbred algorithm uses the set of $\binom{k+d}{d}$ degree D polynomials produced in the previous step. For each specialization of the last $n-k$ variables, it tests the consistency of the resulting system (of degree at most d and k variables) by linearization.

Complexity. The number of columns of the submatrix $\mathbf{M}_{D,d}$ is

$$n_{cols} = \sum_{d_k=d+1}^{D} \sum_{d'=0}^{D-d_k} [x^{d_k} z^{d'}] H_k(x) H_{n-k}(z), \text{ and } \tilde{n}_{cols} = [z^d] \left(\frac{H_k(z)}{1-z} \right),$$

where $H_k(z) := \frac{(1-z^q)^k}{(1-z)^k}$. Hence, if use Gaussian elimination to get the full left kernel space of $\mathbf{M}_{D,d}$, then the complexity of the preprocessing step is given by $\mathcal{O}\left(n_{cols}^{\omega}\right)$, where $2 \leqslant \omega \leqslant 3$. On another hand, we can repeatedly use the Block-Wiedemann algorithm (on average) $\binom{k+d}{d}$ times find get the same number of linearly independent vectors in the left kernel space of $\mathbf{M}_{D,d}$. In this case, the complexity preprocessing step is

$$3\binom{k+d}{d}\binom{n+2}{2} \cdot n_{cols}^2$$

We refer to [51, Thm. 7] for details of the complexity of the Block-Wiedemann algorithm.

To derive the complexity of the linearization step algorithm we assume that after each specialization of the last $n-k$ variables the remain system is semi-regular. Hence, it is given by $\mathcal{O}\left(m \cdot q^{n-k}\binom{k+d}{d}^{\omega}\right)$.

Overall, the complexity of the Crossbred algorithm is given by

$$\min\left\{ n_{cols}^{\omega}, 3\binom{k+d}{d}\binom{n+2}{2} \cdot n_{cols}^2 \right\} + \mathcal{O}\left(m \cdot q^{n-k} \binom{k+d}{d}^{\omega} \right).$$

Admissible parameters. Given a non-negative integer k, not all pairs (D,d), with $D > d$, are expected to configure the algorithm so that the Crossbred can solve an MQ problem $\mathcal{S}$ with an underlying semi-regular sequence of polynomials. We say that (D,d,k) are *admissible parameters* for $\mathcal{S}$ if the Crossbred algorithm with parameters (D,d,k) is expected to solve $\mathcal{S}$.

In [35], Duarte shows the case $q=2$ of the following fact: if the coefficient of the monomial $x^D y^d$ in the bivariate series

$$\frac{(1+x)^{n-k}}{(1-x)(1-y)} \left(\frac{(1+xy)^k}{(1+x^2y^2)^m} - \frac{(1+x)^k}{(1+x^2)^m} \right) - \frac{(1+y)^k}{(1-x)(1-y)(1+y^2)^m} \tag{5}$$

is non-negative, then (D,d,k) are admissible parameters for $\mathcal{S}$.

Here we remark that for the case $q \geqslant 2$, it can be generalized as

$$S_k(x,y) := \frac{\tilde{H}_{k,m}(xy) \cdot H_{n-k}(x) - \tilde{H}_{n,m}(x) - \tilde{H}_{k,m}(y)}{(1-x)(1-y)}, \tag{6}$$

where $\tilde{H}_{k,m}(y)$ is the Hilbert series of a semi-regular sequence with m equations in k variables, i.e.,

$$\tilde{H}_{k,m}(y) := \left[\frac{(1-y^q)^k}{(1-y)^k} \left(\frac{1-y^2}{1-y^{2q}} \right)^m \right]_+.$$

Notice that the Equation (6) reduces to the Equation (5) when $q = 2$.

The space complexity is dominated by the memory needed to store the matrices in the preprocessing and the linearization steps. Thus, the space complexity is given by

$$\mathcal{O}\left(n_{cols}^2 + \binom{k+d}{d}^2\right).$$

3.5 Probabilistic algorithms

In this section, we describe the complexity of the so-called *probabilistic algorithms* for solving the MQ problem. A common and interesting feature of these algorithms is that they asymptotically outperform the exhaustive search algorithms in the worst case, and their complexity does not depend on any assumption about the input polynomials.

Lokshtanov et al.'s Lokshtanov et al. [55] were the first to introduce a probabilistic algorithm that, in the worst case, solves a square ($m = n$) polynomial system over $\mathbb{F}_q$ in time $\tilde{\mathcal{O}}(q^{\delta n})$, for some $\delta < 1$ depending only on q and the degree of the system, without relying on any unproven assumption. Overall, Lokshtanov et al.'s is an algorithm to check the consistency, i.e., determining whether or not the system has a solution to a given MQ problem. Computing a solution can be done by using this algorithm several times. Letting $q = p^d$, for a prime number p and a positive integer d, the time complexity of Lokshtanov et al.'s algorithm is:

- $\tilde{\mathcal{O}}\left(2^{0.8765n}\right)$ for $q = 2$.
- $\tilde{\mathcal{O}}\left(q^{0.9n}\right)$ for $q > 2$ being a power of 2.
- $\tilde{\mathcal{O}}\left(q^{0.9975n}\right)$, when $p > 2$, $\log p < 8e$.
- For $\log p \geqslant 8e$, the complexity is given by

$$\tilde{\mathcal{O}}\left(q^n \cdot \left(\frac{\log q}{2ed}\right)^{-dn}\right).$$

To derive a more precise time complexity estimation, let us assume that $C(n, m, q)$ is the time complexity of checking the Loskshtanov et al.'s algorithm to check the consistency of a given MQ problem. Then, the complexity of finding one solution to the MQ problem is upper bounded by

$$(q-1)\sum_{i=0}^{n-1} C(n-i, m, q).$$

Lokshtanov et al. [55] proved that

$$C(n, m, q) = \mathcal{O}\left(100n\log_2(q)\left(q^{n-n'} + q^{n'} \cdot M(n-n', 2(q-1)(n'+2), q) \cdot n^{6q}\right)\right),$$

where $n' = \lfloor \delta n \rfloor$, and $M(n, d, q)$ is the number of monomials of degree at most d on n variables in $\mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$, i.e., $M(n, d, q)$ is the coefficient of z^d in the series

$$\frac{(1 - z^q)^n}{(1 - z)^{n+1}}.$$

See [69, Lemma 1] for further details. The space complexity of this algorithm is given by

$$\mathcal{O}\left(M(n - n', k(q-1)(n'+2), q) + \log_2 n q^{n-n'}\right).$$

Björklund et al.'s Based on the ideas from Lokshtanov et al., Björklund et al. [17] devised an algorithm to compute the parity of the number of solutions of a given MQ problem over $\mathbb{F}_2$. They showed how their algorithm could be used to solve the MQ problem with time complexity $\tilde{\mathcal{O}}\left(2^{0.803225n}\right)$.

Let $\mathcal{F}$ be a sequence of polynomials over $\mathbb{F}_2$, and for a given integer n_1, let $x = (y, z)$ be a partition of the variables in x, where y and z correspond to the first $n - n_1$ and the last n_1 variables in x, respectively. The main idea behind the Björklund et al.'s algorithm is to approximate the values of the polynomial $P(y)$ given by

$$P(y) = \sum_{\hat{z} \in \{0,1\}^{n_1}} F(y, \hat{z}), \text{ where } F(y, z) = \prod_{f \in \mathcal{F}} (1 + f(y, z)). \tag{7}$$

This approximation is made by computing $48n + 1$ times the values of $\tilde{P}(\tilde{y})$ for every $\hat{y} \in \{0, 1\}^{n-n_1}$, where $\tilde{P}$ is the polynomial given by

$$\tilde{P}(y) = \sum_{\hat{z} \in \{0,1\}^{n_1}} \tilde{F}(y, \hat{z}), \text{ where } \tilde{F}(y, z) = \prod_{g \in \mathcal{G}} (1 + g(y, z)), \tag{8}$$

and $\mathcal{G}$ is a sequence of polynomials formed by random $\mathbb{F}_2$-linear combinations of the polynomials in $\mathcal{F}$. Notice, for a fixed $\hat{y} \in \{0, 1\}^{n-n_1}$ the value $\tilde{P}(\hat{y})$ is the parity of the function $\tilde{F}(\hat{y}, z)$, i.e., the parity of the addition of all $\tilde{F}(\hat{y}, \hat{z})$ with $\hat{z} \in \{0, 1\}^{n_1}$. Also, the parity of $\tilde{F}$ is the parity of the sum of all values $\tilde{P}(\tilde{y})$, and at the same time is an approximation to the parity of F. Since this last parity is clearly the parity of the number of solutions of the MQ problem, we can say the Bjöklund et al. parity counting algorithm reduces to many smaller instances of the same problem, namely, the parity of $\tilde{F}(\hat{y}, z)$ for every $\hat{y} \in \{0, 1\}^{n-n_1}$ for each of the $48n + 1$ polynomial s $\tilde{F}$.

Let $T(n, m)$ be the time complexity of Björklund et al.'s parity-counting algorithm. Thus, it can be transformed to solve the MQ problem in polynomial time. More precisely, the expected number of operations is given by

$$\mathcal{O}\left(8k \log(n) \sum_{i=1}^{n-1} T(n - i, m + k + 2)\right),$$

where $k = \lfloor \log(n_{sol} + 1) \rfloor$[4].

[4] The factor $8k \log n$ comes from the expected complexity of the Valiant–Vazirani isolation's algorithm with probability $1 - 1/n$, see [17, Sec. 2.5] for more details.

It is proven that

$$T(n,m) \leqslant s \cdot \tau(\ell) \left(T(\ell, \ell+2) + \mathcal{O}\left(\cdot (n + (\ell+2)m \binom{n}{\downarrow 2} + 2^{n-\ell}(n-\ell) \right) \right),$$

where $\ell = \lfloor \lambda n \rfloor$, $0 < \lambda < 1$, $s = 48n + 1$, $\tau(\ell) := \binom{n-\ell}{\downarrow \ell+4}$, and $\binom{n}{\downarrow k} := \sum_{j=0}^{k} \binom{n}{j}$. For the space complexity, denoted by $S(n,m)$, the following formula is proven

$$S(n,m) \leqslant S(\ell, \ell+2) + \mathcal{O}\left(2^{n-\ell} \log s + m \binom{n}{\downarrow 2} \right).$$

Dinur's first algorithm Inspired by the ideas from Björklund et al., Dinur [34] proposed an algorithm to compute the parity of the number of solutions of a given MQ problem over $\mathbb{F}_2$. Unlike Björklund et al.'s, Dinur's parity-counting algorithm does not compute one by one the 2^{n-n_1} values of an specialization of the polynomial $\tilde{P}$ given in (8). Instead, it computes all the values $\tilde{P}(\hat{y})$ at once via an algorithm to solve so-called *multiple parity counting* problem, such an algorithm is named the *multiparity algorithm*. Like in the Björklund et al.'s the Dinur's parity-counting algorithm computes $48n + 1$ time all the values of $\tilde{P}(\hat{y})$ to approximate the values of the polynomial $P(y)$ given in (7). Then, it adds the 2^{n-n_1} approximated values $\tilde{P}(\hat{y})$ to get the parity of the number of solutions.

The complexity of then Dinur's parity-counting algorithm is given by $T(n_1, n-n_1, m)$, where $0 \leqslant n_1 < n$, and $T(\cdot,\cdot,\cdot)$ is defined by

$$T(n_1, w, m) := \mathcal{O}\left(t \cdot \left(T(n_2, 2\ell - n_2, \ell) + n \binom{n-n_1}{\downarrow w} 2^{n_1-n_2} + \right.\right.$$
$$\left.\left. + n \binom{n-n_2}{\downarrow 2\ell - n_2} \right) + \ell m \binom{n}{\downarrow 2} \right),$$

with $t = 48n + 1$, $n_1 = \lfloor \kappa_0 n \rfloor$, $n_2 = \lfloor n_1 - \lambda n \rfloor$, $\ell = \min\{n_2 + 2, m\}$, $0 < \kappa_0 \leqslant \frac{1}{3}$, and $0 < \lambda < 1$. In the particular case, $n_2 \leqslant 0$, in this case $T(n,w) := n \cdot \binom{n-n_1}{\downarrow w} 2^{n_1}$.

Similarly to the Björklund et al.'s algorithm, the complexity of solving the MQ problem using Dinur's parity-counting is given by

$$\mathcal{O}\left(8k \log(n) \sum_{i=1}^{n-1} T(n_{1i}, n-i-n_{1i}, m+k+2) \right),$$

where $k = \lfloor \log(n_{sol} + 1) \rfloor$, and $n_{1i} = \lfloor \kappa_0 (n-i) \rfloor$. The $\tilde{\mathcal{O}}(\cdot)$ complexity estimated by Dinur is $\tilde{\mathcal{O}}\left(2^{0.6943n}\right)$. The space complexity of the algorithm is dominated by $\mathcal{O}\left((48n+1)2^{n-n_1}\right)$.

Dinur's second algorithm Dinur [33] proposed a more efficient algorithm to solve the MQ problem over $\mathbb{F}_2$.

Let $\mathcal{F}$ be the sequence of polynomials of a given MQ problem. The main difference with his previous algorithm is that the new algorithm does not approximate the parity of $n_{sol}(\mathcal{F})$ by computing the parity of $48n+1$ specializations of the polynomial $\tilde{F}$ from Equation 8. Instead, it computes all the solution of few specializations (up to 4 or 5) of the probabilistic sequence $\mathcal{G}$ from Equation 8. All of such solutions are tested on the original sequence $\mathcal{F}$ until a solution is found.

Let n_1 be an integer satisfying that $m \leqslant 2(n_1 + 1) + 2$. Then, the time complexity $T(n, n_1)$ of the Dinur's second algorithm is tightly upper bounded by

$$T(n, n_1) \leqslant 16 \log n \cdot 2^{n_1} \cdot \binom{n - n_1}{\downarrow n_1 + 2} + n_1 \cdot n \cdot 2^{n-n_1} + 2^{n-2n_1+1} \binom{n}{\downarrow 2}.$$

Moreover, the fact $n_1 \approx n/5.4$ yields to $T(n, n_1) \leqslant n^2 \cdot 2^{n-n_1}$. The space complexity is given by

$$8 \cdot (n_1 + 1) \cdot \binom{n - n_1}{\downarrow n_1 + 3}.$$

4 Algorithms not considered in our estimator

Here we collect the algorithms designed to solve the MQ that are not considered explicitly in the software of our estimator.

Hashimoto's algorithm Hashimoto [47] proposed two algorithms, namely Has-1 and Has-2, to solve the MQ problem in the underdetermined case. However, Miura, Hashimoto, and Takagi [57] showed that Has-1 does not work, and they provided a better analysis for Has-2, see Section 3.2.

Linearization algorithms The linearization algorithms reduce the MQ problem to solve a, usually large, linear system of equations. The most general of this kind of algorithms is the **X**tended **L**inearization (XL) algorithm introduced by Courtois et al. in [25]. Moreover, XL is essentially the same algorithm independently devised by Lazard [54], and it can be viewed as a redundant variant of a Gröbner basis algorithms F4 and F5 [3]. Also, in the estimator, we do include the more general variant of XL called FXL, see Section 3.4.

Courtois et al.'s algorithms In our estimator, we do not consider the so-called algorithm B nor algorithm C by Courtois et al. [26]. Algorithm B is a particular case of the Crossbred algorithm where D and d are set to be 2 and 1, respectively. While algorithm C is the same strategy described by Thomae and Wolf in [67] over fields of characteristic 2.

5 The estimator

Here we describe the software, named the MQ estimator, to estimate the hardness of the MQ problem. Also, we show how its estimates compare with the best-known practical results and previous estimations for some multivariate-based signature schemes.

5.1 Description/Usage

This section documents the overall structure of our MQ estimator. Full documentation is publicly available in [9].

Main class:

`MQEstimator`: This class is to compute the complexity estimates for a given instances q, n, and m of the MQ problem. By default, for each algorithm X that works on a problem defined by q, n, and m, it builds one method to estimate the complexity of X.

The main parameters of this class are:

`q`: the size of the field
`n`: the number of variables
`m`: the number of polynomials

other optional parameters are:

`w`: the matrix multiplication complexity exponent (Default value: `2`).
`theta`: the real number $\theta \in [0, 2]$ such that $(\log q)^\theta$ is the ratio between field operations complexity and bit complexity, see Section 2.2 (Default value: `0`).
`h`: An integer specifying the external hybridization parameter. It assumes that the computation is split into q^h subsystem of $n - h$ variables and m equations. Hence, the time complexity of solving q^h of the aforementioned subsystems. In contrast, the space complexity is the amount of memory needed to solve only one subsystem (Default value: `0`).
`excluded_algorithms`: list of algorithms not to be consider (Default value: `[]`).
`tilde_o_complexity`: the Boolean value determining if the complexity is estimated by just plugged in number is the $\tilde{O}$ complexity formula, if any (Default value: `False`).
`nsolutions`: the number of solutions of the underlying problem (Default value: `1`).

5.2 Numerical results

Table 1 compares the complexity estimates of some of the algorithms used to break instances of the FMQ challenge [70]. The column *algorithm used* shows the

algorithm used to break the corresponding challenge, while the column *equivalent algorithm* gives the algorithm in our MQ estimator tool used to estimate the complexity of the algorithm used. The column *best algorithm* shows, according to our estimator, the best theoretical algorithm to solve a given instance. The time and memory estimates under the column *practical* (resp. *theoretical*) are practical (resp. theoretical) estimates of the number of clock cycles (resp. bit time complexity) and memory (resp. bit space complexity) used to solve the corresponding MQ problem. The value h denotes the external hybridization parameter (see Section 5.1). All theoretical estimates are computed with our MQ estimator tool, where we used $\omega = 2.81$, and $\theta = 2$. For the *best algorithm* we set $h = 0$, while for the *equivalent algorithm* we set h to be the same used to break the challenge.

For all the algorithms but M4GB, our estimator provides tight estimates compared to the practical ones computed from the information provided by the challenge's breakers. The small gap between the time estimates of *algorithm used* and the *equivalent algorithm* is due to the different metrics used in each case. We use the number of clock cycles for the practical estimates and the number of bit operations for the theoretical ones. It is well known that several bit operations can be done in a single clock cycle. Still, the exact number depends on the device used and the implementation of the specific task[5].

The best time complexity for the Type I and IV parameters is way better than the one used to break the challenge. Still, the space complexities of the former are significantly larger than the latter, which makes it difficult to solve in practice. For Type II, III, and VI parameters, both the space complexity of the best algorithm and the algorithm used are relatively similar. Still, the time complexity is way better in the former than in the latter. This fact suggests that the estimated best algorithm would outperform (in practice) the one used in the MQ challenge. Finally, for Type V, we have that the FXL algorithm is better in terms of time and memory than the Hybrid-F5 algorithm.

Figures 1a, 1c, and 1b illustrate the bit complexity estimates of the MQ problem for different parameters q, n and m. Each line represents a different value of q, while a bullet over a given line indicates the algorithm providing such a complexity estimate. These figures are generated by setting $\mathtt{w} = 2.81$ and $\theta = 2$ in our estimator, and we left by default the rest of the inputs. In each figure, we emphasize on a different regime of the ratio m/n. Figure 1a $m/n = 1$ (the square case), Figure 1c $m/n > 1$ (the overdetermined case), and Figure 1b $m/n < 1$ (the underdetermined case).

In the square case we observe that, for $q = 2$ the Dinur's second algorithm provide the best estimates given $n > 30$, while the the exhaustive search strategy

[5] For instance, for the Type IV parameters where $n = 66$, the authors used several Spartan-6 FPGAs to break the challenge. There the authors implemented the FES algorithm to solve a system with 48 variables and equations. Such particular implementation allowed them to test 2^{10} potential solutions per clock cycle, which means they are computing at least 2^{10} bit operations per clock cycle.

[6] No publication describing this implementation in detail.

Table 1: Comparison between the estimates of some of the algorithms used to break FMQ challenges [70] and the estimates using the MQ estimator.

Parameters				Complexity estimates					
				Practical *algorithm used* (*parameters used*)		Theoretical *equivalent algorithm*		Theoretical *best algorithm* (*optimal parameters*)	
Type	q	n	m	time	memory	time	memory	time	memory
I	2	74	148	Crossbred [61] ($k=22, D=4, d=1, h=10$)		Crossbred		Crossbred ($k=28, D=6, d=1$)	
				64.3	34.6	71.9	27.0	67.1	46.7
IV	2	99	66	FES [19] ($h=0$)		FES		Dinur2 ($n_1=12$)	
				55.9	n/a	69.6	18.1	64	50.3
IV	2	103	69	Crossbred[6] ($k=15, D=4, d=1, h=17$)		Crossbred		Dinur2 ($n_1=12$)	
				67	n/a	79.1	17.7	66.6	52.7
II	2^8	37	74	F4-style [49] ($h=0$)		F5		Crossbred ($k=33, D=7, d=1$)	
				60.1	38.5	76.8	52.9	67.2	48.0
V	2^8	28	19	M4GB [56] ($h=1$)		Hybrid-F5		FXL ($k=2$, *variant* = *las vegas*)	
				54.3	$\leqslant 37.9$	79.7	47.7	77	33.5
III	31	38	76	XL [24] ($h=0$)		FXL		Crossbred ($k=31, D=7, d=1$)	
				59.0	35.5	66.7	39.2	66.3	47.8
VI	31	30	20	M4GB [56] ($h=2$)		Hybrid-F5		Crossbred ($k=12, D=8, d=2$)	
				55.3	$\leqslant 37.9$	71.3	40.1	66.8	41.3

is the best for $n \leqslant 30$. A similar observation was noted in practice [5]. For $q = 2^8$ and $q = 31$, the algorithms FXL and Crossbred provides the best time estimates for all instances. In the case $m = 2n$, the Crossbred mostly outperform the others as long as $n > 28$, and doing exhaustive search is the best for $n \leqslant 28$ and $q = 2$. The case $n = 1.5m$ exhibits a similar behavior than the square case. Here instead, for $q = 2$, the Dinur's second algorithm starts being better than exhaustive search just until $n > 46$.

Remark 5.1. In Figure 1c, the exponent of the time complexity grows a the same rate for several consecutive values of n, but at a specific value, this rate suddenly increases. This is because the values of D providing the best time complexity for Crossbred increases by one after n reaches a particular value, and this causes a considerable increase in the size of the matrix in the preprocessing step. Hence the cost of the processing step increases considerably.

Figure 1d shows bit complexity as a function of the ratio m/n for $q = 2$. Here, each line represents a different value of n, and a bullet represents the algorithm's name providing the corresponding complexity. Note that, for every q, the Crossbred algorithm gets better than the others once μ is bigger than a threshold. This threshold is 1.7, 1.5, and 1.4 for $n = 40, 60, 80$, respectively.

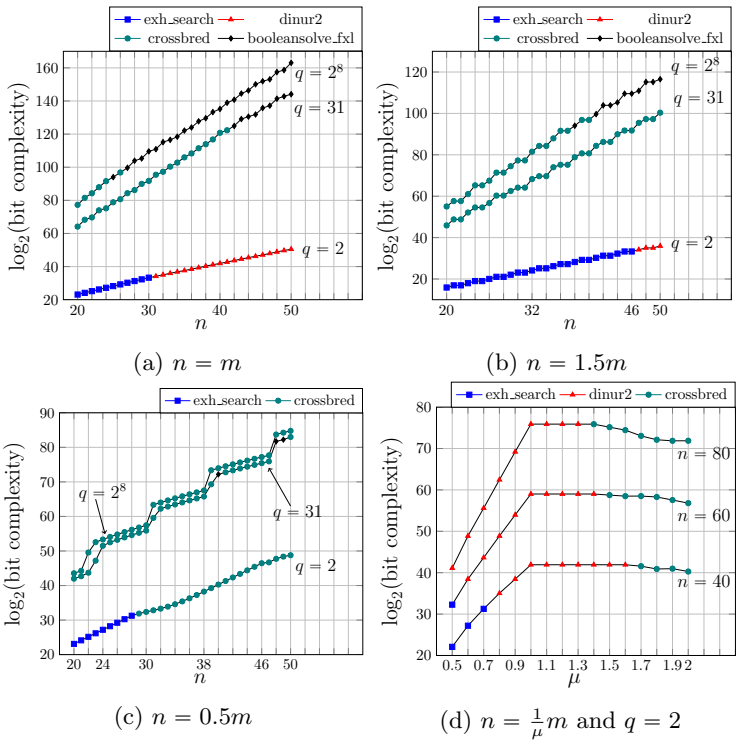

Fig. 1: Numerical estimates of the bit-complexity of the MQ problem.

5.3 Security of MPKCs against the direct attack

In this section, we use our MQ estimator to estimate the security against the direct attack of Rainbow [30], MAYO [16], MQDSS [23], and MUDFISH [14]. We configure out estimator with $\omega = 2.81$, $\theta = 2$, and $D \leqslant 40$ in the Crossbred algorithm.

Commonly, the public key of an MPKC in a quadratic map $P : \mathbb{F}_q^n \to \mathbb{F}_q^m$ over a finite field $\mathbb{F}_q$. Undermined the security of these schemes is always possible by solving an MQ problem $P(x) = y$, where $y \in \mathbb{F}_q^m$. This attack is known as the *direct attack*.

Instead of directly trying to find a preimage of P, an attacker could try to recover the secret key by using the knowledge of P and the underlying structure of the cryptosystem. These attacks are called *structural attacks*. For some cryptosystems, e.g., Rainbow there is a structural attack that is more efficient than the direct one [15]. Still, for some cryptosystems, the direct attack is better than any other known structural attack, e.g., MAYO.

In [62], NIST defines five security categories, namely, category I, II, III, IV, and V. It establishes that cryptographic schemes with classical security categories I, III, and V must provide 143, 207 and 272 bits of security, respectively.

Table 2: Security estimates against the direct attack for Rainbow.

Category	q	n	m	Claimed security	Estimated security	Algorithm (*parameters*)
I	16	100	64	**164**	166	FXL ($k = 15$, variant = *las vegas*)
III	256	148	80	**234**	240.1	FXL ($k = 7$, variant = *las vegas*)
V	256	196	100	**285**	291.7	FXL ($k = 9$, variant = *las vegas*)

Tables 2 and 3 show the security estimates of Rainbow and the MAYO, respectively. We observed that the parameters for Category I fail to achieve their target security level. Instead, all the Rainbow parameters achieve the claimed bit security. Finally, in Table 4 we see that all the parameters of MUDFISH and MQDSS are far below the claimed bit security. Still, all the parameters achieve their target category of security.

Table 3: Security estimates against the direct attack for MAYO.

Category	q	n	m	Claimed security	Estimated security	Algorithm (*parameters*)
I	7	962	76	145	**140.4**	Crossbred ($k = 23, D = 19, d = 1$)
I	7	1140	78	145	**140.4**	Crossbred ($k = 23, D = 19, d = 1$)
III	7	2220	117	210	**209.7**	Crossbred ($k = 39, D = 28, d = 2$)
III	7	2240	117	209	**207.9**	Crossbred ($k = 39, D = 28, d = 2$)
V	7	2960	152	**273**	274	FXL ($k = 44$, variant = *las vegas*)
V	7	3874	157	**273**	274	FXL ($k = 44$, variant = *las vegas*)

Notice that our better complexity estimates are mainly due to new complexity estimates for the Crossbred algorithm, see Section 3.4. These estimates take into account the role of the field equations over a field with few elements, i.e., for small values of q. We believe that such more general formulas were not

used in previous complexity estimates. Thus, we provide more accurate security estimates for the schemes considered in this section.

Table 4: Security estimates against the direct attack for MUDFISH and MQDSS

Category	q	n	m	Claimed security	Estimated security	Algorithm (*parameters*)
I	4	88	88	156	**148.7**	Crossbred ($k = 25, D = 18, d = 1$)
III	4	128	128	230	**211.6**	Crossbred ($k = 41, D = 25, d = 2$)
V	4	160	160	290	**261.1**	Crossbred ($k = 42, D = 32, d = 1$)

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015), https://eprint.iacr.org/2015/046
2. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms pp. 522–539 (2021)
3. Ars, G., Faugère, J.C., Imai, H., Kawazoe, M., Sugita, M.: Comparison between XL and Gröbner basis algorithms. In: Lee, P.J. (ed.) Advances in Cryptology - ASIACRYPT 2004. pp. 338–353. Springer Berlin Heidelberg (2004)
4. Ayad, A.: A survey on the complexity of solving algebraic systems. Int. Math. Forum **5**(5-8), 333–353 (2010)
5. Barbero, S., Bellini, E., Sanna, C., Verbel, J.: Practical complexities of probabilistic algorithms for solving Boolean polynomial systems. Discrete Appl. Math. **309**, 13–31 (2022)
6. Bard, G.V.: Algorithms for Solving Linear and Polynomial Systems of Equations over Finite Fields with Applications to Cryptanalysis. Theses, University of Maryland (2007)
7. Bardet, M., Faugère, J.C., Salvy, B.: On the complexity of the F5 Gröbner basis algorithm. Journal of Symbolic Computation **70**, 49–70 (2015). https://doi.org/https://doi.org/10.1016/j.jsc.2014.09.025
8. Bardet, M., Faugère, J.C., Salvy, B., Spaenlehauer, P.J.: On the complexity of solving quadratic Boolean systems. Journal of Complexity **29**(1), 53–75 (2013). https://doi.org/https://doi.org/10.1016/j.jco.2012.07.001
9. Bellini, E., Makarim, R., Verbel, J.: An estimator for the complexity of the MQ problem. (2021), https://github.com/Crypto-TII/multivariate_quadratic_estimator
10. Bellini, E., Esser, A.: Syndrome decoding estimator (2021), https://github.com/Crypto-TII/syndrome_decoding_estimator

11. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post–Quantum Cryptography. Springer-Verlag Berlin Heidelberg (2009)
12. Bernstein, D.J., Yang, B.Y.: Asymptotically faster quantum algorithms to solve multivariate quadratic equations. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 487–506. Springer International Publishing, Cham (2018)
13. Beullens, W.: Sigma protocols for MQ, PKP and SIS, and fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 183–211. Springer International Publishing, Cham (2020)
14. Beullens, W.: Sigma protocols for mq, pkp and sis, and fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 183–211. Springer International Publishing, Cham (2020)
15. Beullens, W.: Improved cryptanalysis of UOV and Rainbow. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 348–373. Springer International Publishing, Cham (2021)
16. Beullens, W.: Mayo: Practical post-quantum signatures from oil-and-vinegar maps. In: Hülsing, A., AlTawy, R. (eds.) Selected Areas in Cryptography. Springer International Publishing (2021)
17. Björklund, A., Kaski, P., Williams, R.: Solving systems of polynomial equations over GF(2) by a parity-counting self-reduction. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) International Colloquium on Automata, Languages and Programming – ICALP 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.26
18. Bouillaguet, C., Chen, H., Cheng, C., Chou, T., Niederhagen, R., Shamir, A., Yang, B.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: Cryptographic Hardware and Embedded Systems, CHES 2010. pp. 203–218 (2010)
19. Bouillaguet, C., Cheng, C., Chou, T., Niederhagen, R., Yang, B.: Fast exhaustive search for quadratic systems in $\mathbb{F}_2$ on FPGAs. In: Lange, T., Lauter, K.E., Lisonek, P. (eds.) Selected Areas in Cryptography - SAC 2013. Lecture Notes in Computer Science, vol. 8282. Springer (2013). https://doi.org/10.1007/978-3-662-43414-7_11
20. Buchmann, J.A., Ding, J., Mohamed, M.S.E., Mohamed, W.S.A.E.: MutantXL: Solving multivariate polynomial equations for cryptanalysis. In: Handschuh, H., Lucks, S., Preneel, B., Rogaway, P. (eds.) Symmetric Cryptography. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009), http://drops.dagstuhl.de/opus/volltexte/2009/1945
21. Casanova, A., Faugère, J.C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS: A great multivariate short signature. NIST CSRC (2017), official website: https://www-polsys.lip6.fr/Links/NIST/GeMSS.html
22. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: SOFIA: $\mathcal{MQ}$-based signatures in the qrom. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography – PKC 2018. pp. 3–33. Springer International Publishing, Cham (2018)
23. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: MQDSS specifications (2020), http://mqdss.org/specification.html
24. Cheng, C.M., Chou, T., Niederhagen, R., Yang, B.Y.: Solving quadratic equations with XL on parallel architectures. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. pp. 356–373. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
25. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. EUROCRYPT 2000, LNCS **1807**, 392–407 (2000)

26. Courtois, N., Goubin, L., Meier, W., Tacier, J.D.: Solving underdefined systems of multivariate quadratic equations. In: Naccache, D., Paillier, P. (eds.) Public Key Cryptography. pp. 211–227. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
27. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) Advances in Cryptology — EUROCRYPT 2000. pp. 392–407. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
28. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
29. Dickenstein, A., Emiris, I.Z.: Solving Polynomial Equations. Foundations, Algorithms, and Applications, Algorithms and Computation in Mathematics, vol. 14. Springer-Verlag Berlin Heidelberg (2005)
30. Ding, J., Chen, M., Petzoldt, A., Schmidt, D., Yang, B.: Rainbow. NIST CSRC (2017), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions
31. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 164–175. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
32. Ding, J., Zhang, Z., Deaton, J.: How much can F5 really do. Cryptology ePrint Archive, Report 2021/051 (2021), https://eprint.iacr.org/2021/051
33. Dinur, I.: Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021. pp. 374–403. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_14
34. Dinur, I.: Improved algorithms for solving polynomial systems over GF(2) by multiple parity-counting. In: ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 2550–2564 (2021). https://doi.org/10.1137/1.9781611976465.151
35. Duarte, J.D.: On the complexity of the crossbred algorithm. Cryptology ePrint Archive, Report 2020/1058 (2020), https://eprint.iacr.org/2020/1058
36. Eder, C., Faugère, J.C.: A survey on signature-based algorithms for computing Gröbner bases. Journal of Symbolic Computation **80**, 719–784 (2017)
37. Esser, A., Bellini, E.: Syndrome decoding estimator. Cryptology ePrint Archive, Report 2021/1243 (2021), https://ia.cr/2021/1243
38. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139**(1), 61–88 (1999)
39. Faugère, J.C.: 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. p. 75–83. ISSAC '02, Association for Computing Machinery, New York, NY, USA (2002)
40. Faugère, J., Horan, K., Kahrobaei, D., Kaplan, M., Kashefi, E., Perret, L.: Fast quantum algorithm for solving multivariate quadratic equations. CoRR **abs/1712.07211** (2017)
41. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation **16**(4), 329 – 344 (1993)
42. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)

43. Furue, H., Nakamura, S., Takagi, T.: Improving Thomae-Wolf algorithm for solving underdetermined multivariate quadratic polynomial problem. In: Cheon, J.H., Tillich, J.P. (eds.) PQcrypto 2021. LNCS. pp. 65–78. Springer International Publishing, Cham (2021)
44. Furue, H., Duong, D., Takagi, T.: An efficient MQ-based signature with tight security proof. International Journal of Networking and Computing **10**(2), 308–324 (2020), http://www.ijnc.org/index.php/ijnc/article/view/238
45. Fusco, G., Bach, E.: Phase transition of multivariate polynomial systems. In: Cai, J.Y., Cooper, S.B., Zhu, H. (eds.) Theory and Applications of Models of Computation. pp. 632–645. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
46. Gashkov, S.B., Sergeev, I.S.: Complexity of computations in finite fields. Fundam. Prikl. Mat. **17**(4), 95–131 (2011/12)
47. Hashimoto, Y.: Algorithms to solve massively under-defined systems of multivariate quadratic equations. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E94.A**(6), 1257–1262 (2011). https://doi.org/10.1587/transfun.E94.A.1257
48. Huang, H., Bao, W.: Algorithm for solving massively underdefined systems of multivariate quadratic equations over finite fields (2015)
49. Ito, T., Shinohara, N., Uchiyama, S.: An efficient F4-style based algorithm to solve MQ problems. In: Attrapadung, N., Yagi, T. (eds.) Advances in Information and Computer Security. pp. 37–52. Springer International Publishing, Cham (2019)
50. Joux, A., Vitse, V.: A crossbred algorithm for solving Boolean polynomial systems. In: Kaczorowski, J., Pieprzyk, J., Pomykała, J. (eds.) Number-Theoretic Methods in Cryptology. pp. 3–21. Springer International Publishing, Cham (2018)
51. Kaltofen, E.: Analysis of coppersmith's block wiedemann algorithm for the parallel solution of sparse linear systems. Mathematics of Computation **64**(210), 777–806 (1995), http://www.jstor.org/stable/2153451
52. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. EUROCRYPT 1999. LNCS **1592**, 206–222 (1999)
53. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT '99. pp. 206–222. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
54. Lazard, D.: Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In: Computer Algebra, EUROCAL '83, European Computer Algebra Conference, London, England, March 28-30, 1983, Proceedings. pp. 146–156 (1983)
55. Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields. In: Symposium on Discrete Algorithms. p. 2190–2202. SODA '17, Society for Industrial and Applied Mathematics, USA (2017)
56. Makarim, R.H., Stevens, M.: M4GB: an efficient Gröbner-basis algorithm. In: Burr, M.A., Yap, C.K., Din, M.S.E. (eds.) Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2017, Kaiserslautern, Germany, 2017. pp. 293–300. ACM (2017), https://doi.org/10.1145/3087604.3087638
57. Miura, H., Hashimoto, Y., Takagi, T.: Extended algorithm for solving underdefined multivariate quadratic equations. In: Gaborit, P. (ed.) Post-Quantum Cryptography. pp. 118–135. Springer Berlin Heidelberg (2013)
58. Moody, D.: The homestretch: the beginning of the end of the NIST PQC 3rd round. In: International Conference on Post-Quantum Cryptography (2021), https://pqcrypto2021.kr/download/program/2.2_PQCrypto2021.pdf

59. Mou, C.: Solving Polynomial Systems over Finite Fields: Algorithms, Implementation and Applications. Theses, Université Pierre et Marie Curie (May 2013)
60. Niederhagen, R.: Parallel Cryptanalysis. Ph.D. thesis, Eindhoven University of Technology (2012), http://polycephaly.org/thesis/index.shtml
61. Ning, K.C.: An adaption of the crossbred algorithm for solving multivariate quadratic systems over $\mathbb{F}_2$ on GPUs (2017), https://pure.tue.nl/ws/portalfiles/portal/91105984/NING.K_parallel_cb_v103.pdf
62. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2017), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
63. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 706–723. Springer Berlin Heidelberg (2011)
64. Schwabe, P., Westerbaan, B.: Solving binary $\mathcal{MQ}$ with Grover's algorithm. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 303–322. Springer International Publishing, Cham (2016)
65. Seres, I.A., Horváth, M., Burcsi, P.: The Legendre pseudorandom function as a multivariate quadratic cryptosystem: Security and applications. Cryptology ePrint Archive, Report 2021/182 (2021), https://ia.cr/2021/182
66. Strassen, V.: Gaussian elimination is not optimal. Numerische mathematik **13**(4), 354–356 (1969)
67. Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography – PKC 2012. pp. 156–171. Springer Berlin Heidelberg (2012)
68. Ullah, E.: New Techniques for Polynomial System Solving. Theses, Universität Passau (Feb 2012)
69. Yang, B.Y., Chen, J.M.: Theoretical analysis of XL over small fields. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) Information Security and Privacy. pp. 277–288. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
70. Yasuda, T., Dahan, X., Huang, Y.J., Takagi, T., Sakurai, K.: MQ challenge: Hardness evaluation of solving multivariate quadratic problems. In: NIST Workshop on Cybersecurity in a Post-Quantum World (2015), washington, D.C. https://www.mqchallenge.org