Polynomial XL: A Variant of the XL Algorithm Using Macaulay Matrices over Polynomial Rings

Hiroki Furue and Momonari Kudo

Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
The University of Tokyo, Tokyo, Japan
furue-hiroki261@g.ecc.u-tokyo.ac.jp
kudo@mist.i.u-tokyo.ac.jp

Abstract. Solving a system of m multivariate quadratic equations in n variables over finite fields (the MQ problem) is one of the important problems in the theory of computer science. The XL algorithm (XL for short) is a major approach for solving the MQ problem with linearization over a coefficient field. Furthermore, the hybrid approach with XL (h-XL) is a variant of XL guessing some variables beforehand. In this paper, we present a variant of h-XL, which we call the *polynomial XL (PXL)*. In PXL, the whole n variables are divided into k variables to be fixed and the remaining $n-k$ variables as "main variables", and we generate a Macaulay matrix with respect to the $n-k$ main variables over a polynomial ring of the k (sub-)variables. By eliminating some columns of the Macaulay matrix over the polynomial ring before guessing k variables, the amount of manipulations required for each guessed value can be reduced. Our complexity analysis of PXL gives a new theoretical bound, and it indicates that PXL is efficient in theory on the random system with $n = m$, which is the case of general multivariate signatures. For example, on systems over $\mathbb{F}_{2^8}$ with $n = m = 80$, the numbers of manipulations deduced from the theoretical bounds of the hybrid approaches with XL and Wiedemann XL and PXL with optimal k are estimated as 2^{252}, 2^{234}, and 2^{220}, respectively.

Keywords: MQ problem, MPKC, XL, hybrid approach, Macaulay matrices

1 Introduction

In the field of computer science, the problem of solving a multivariate polynomial system of degree ≥ 2 over a finite field (*the MP problem*) is one of the most important problems, where "solve" means to find (at least) one root of the system. The particular case where polynomials are all quadratic is called *the MQ problem*, and both the MP and MQ problems are known to be NP-hard [17]. Moreover, the hardness of the MQ problem is nowadays applied to constructing various cryptosystems (e.g., multivariate public key cryptosystems (MPKCs) such as Rainbow [11] and GeMSS [7]). Therefore, the analysis even for the quadratic case is a very important task both in theory and in practice, and thus we mainly focus on solving the MQ problem in this paper.

A precise definition of the MQ problem is the following: Let n and m be positive integers, and let q be a power of a rational prime p. Given a system $F = (f_1, \ldots, f_m)$ of m quadratic polynomials $f_1, \ldots, f_m$ in n variables $x_1, \ldots, x_n$ over a finite field $\mathbb{F}_q$, the MQ problem requires to find $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ such that $f_i(a_1, \ldots, a_n) = 0$ for all $1 \leq i \leq m$. Throughout the rest of this paper, we deal with only the case of $n \leq m$ (*overdetermined* case). This is because algorithms solving the overdetermined MQ problem can be easily applied to the case of $n > m$, since, after $n - m$ variables are randomly specified, the resulting system will have a solution on average.

In the literature, there are various methods for solving the MQ problem such as Gröbner basis method, Linearization, resultant-based method [9, Chapter 3], and Wu's method [26]. In particular, Gröbner basis method is a generic method to solve the MQ problem. The most classical method to compute Gröbner bases is Buchberger's algorithm [6], and ones of the currently most efficient algorithms are Faugère's F_4 and F_5 algorithms [13, 14]. Once a Gröbner basis for the input F is computed for a given monomial order (typically a graded reverse lexicographic order is chosen for practical efficiency) with the above algorithms, the FGLM conversion [15] enables us to obtain its lexicographical Gröbner basis, from which solutions can be easily derived [10, Chapter 3].

As a linearization-based algorithm, Courtois et al. [8] proposed the *XL algorithm* at 2000, and this algorithm is an extension of Relinearization algorithm [20]. The main idea of XL, which is already used in [22] by Lazard in order to analyze Buchberger's algorithm, is: Linearize the given system by regarding each monomial as one variable, and then, similarly to F_4, use linear algebra to the coefficient matrix of the linearized system. More concretely, we first construct a shift S of F, that is, the set of polynomials of the form $t \cdot f_i$ for all $1 \leq i \leq m$ with monomials t up to given degree. By linearizing the system S, we then generate its coefficient matrix (this matrix is nothing but a *Macaulay matrix* of S), and compute its reduced row echelon form by the row reduction (Gaussian elimination). If the shift S is sufficiently large, then the number of independent polynomials in S becomes close to the total number of monomials, and hence a univariate equation would be obtained from the reduced form of the Macaulay matrix. We then solve the obtained univariate equation and repeat such processes with respect to the remaining variables. Note that XL is considered to be a redundant variant of F_4 algorithm (see [1, 2] for details). Furthermore, Yang et al. [29] analyzed a variant of the XL algorithm called *Wiedemann XL (WXL)*, which adopts Wiedemann's algorithm [24] instead of row reduction algorithms in the XL framework. WXL provides another complexity estimate which is used to evaluate the security of various MPKCs such as Rainbow [11].

One of the most effective improvements of XL is to apply the *hybrid approach* [4, 28] (first proposed as FXL in [28] for XL, in which the "F" stands for "fix"), which is proposed as an approach applying an MQ solver such as F_4, F_5, or XL efficiently. This approach fixes the values of k among n variables (say $x_1, \ldots, x_k$), and then solves the remaining system in the $n-k$ variables $x_{k+1},\dots,x_n$ using an MQ solver. These processes are iterated until a solution is found. In the case of $n \approx m$, the hybrid approach may be effective, since the gain obtained by working on systems with less variables may overcome the loss due to the exhaustive search on the fixed variables. In this paper, we call the hybrid approach with XL (resp. WXL) *h-XL* (resp. *h-WXL*).

Our contributions In this paper, we propose a new variant of the XL algorithm, which we call *polynomial XL* (PXL), as an improvement of h-XL. With notation same as in h-XL described above, the main idea of our improvement is the following: Before fixing the values of the variables $x_1,\dots,x_k$, we partly perform Gaussian elimination on a Macaulay matrix *over the polynomial ring* $\mathbb{F}_q[x_1,\dots,x_k]$, with keeping $x_1,\dots,x_k$ as indeterminates. More specifically, for a given MQ system $F=(f_1,\dots,f_m)\in\mathbb{F}_q[x_1,\dots,x_n]^m$, we first regard each f_i as a polynomial in $(\mathbb{F}_q[x_1,\dots,x_k])[x_{k+1},\dots,x_n]$, and construct a shift of F by multiplying all f_i's by monomials in $x_{k+1},\dots,x_n$ (up to some degree). We then generate the Macaulay matrix $\mathcal{PM}$ of the shift with respect to a graded monomial order in $x_{k+1},\dots,x_n$, where $\mathcal{PM}$ is a *polynomial* matrix with entries in the polynomial ring $\mathbb{F}_q[x_1,\dots,x_k]$. Here, due to the gradedness of the monomial order, $\mathcal{PM}$ is *almost upper-block triangular*, and all of its (nearly-)diagonal blocks are matrices with entries in $\mathbb{F}_q$, *not* in $\mathbb{F}_q[x_1,\dots,x_k]$. Thus we can execute row operations on these blocks efficiently, and as a result, we also obtain a partly-reduced matrix. Since the size of the uneliminated part of this resulting matrix is much smaller than that of the original one (e.g., in the case where $n=m=40$ and $k=10$, the sizes of the original matrix and the uneliminated part are approximately 2^{30} and 2^{21}, respectively.), the amount of manipulations for each guessed value can be reduced compared with h-XL. As we will see in Subsection 4.3 below, this enables us to solve the system with smaller complexity for some parameters.
We also discuss the time and space complexities, and theoretically compare them with those of h-XL and h-WXL. Comparing the time complexities, we show that our PXL is the most efficient in theory for the case of $n\approx m$, see Table 1 and Figure 1 for details. For example, on the system over $\mathbb{F}_{2^8}$ with $n=m=80$, the numbers of manipulations in $\mathbb{F}_q$ required for the execution of h-WXL and PXL are estimated as 2^{234} and 2^{220}, respectively. On the other hand, in terms of the space complexity, the proposed algorithm might be not well compared to h-WXL since the sparsity of the Macaulay matrix is not maintained through an execution of the proposed algorithm. Therefore, the relationship between PXL and h-WXL can be seen as a trade-off between time and memory.

Organizations The rest of this paper is organized as follows: Section 2 reviews the XL algorithm and the hybrid approach. Section 3 is devoted to describing the proposed algorithm PXL. We estimate the time complexity, and theoretically compare it with those of h-XL and h-WXL in Section 4, and Section 5 introduces experimental results obtained by our implementation of PXL. Finally, Section 6 is devoted to the conclusion, where we summarize the key points and suggest possible future works.

2 Preliminaries

In this section, we recall the definition of the XL algorithm [8], and discuss its complexity. We also explain the hybrid approach, which combines an exhaustive search with an MQ solver such as XL.

2.1 Notation and Macaulay matrices

We first fix the notations that are used in the rest of this section. Let $R[x] = R[x_1, \ldots, x_n]$ denote the polynomial ring with n variables over a commutative ring R with unity, and let $\mathrm{Mon}(R[x])$ denote the set of monomials in $R[x]$, say $\mathrm{Mon}(R[x]) = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} : (\alpha_1, \ldots, \alpha_n) \in (\mathbb{Z}_{\geq 0})^n\}$. For each $d \geq 0$, we also denote by T_d (resp. $T_{\leq d}$) the set of all monomials in $R[x]$ of degree d (resp. $\leq d$). For a subset $F \subset R[x]$, the ideal of $R[x]$ generated by F is denoted by $\langle F \rangle_{R[x]}$ or simply $\langle F \rangle$. In particular, when F is a finite set $\{f_1, \ldots, f_m\}$, we denote it by $\langle f_1, \ldots, f_m \rangle$. For finite subsets $F \subset R[x]$ and $T \subset \mathrm{Mon}(R[x])$, we set $T \cdot F = \{t \cdot f : t \in T,\ f \in F\}$, which is called the *shift* of F by T. For a polynomial $f \in R[x]$ and a monomial $t \in \mathrm{Mon}(R[x])$, let $\mathrm{coeff}(f, t)$ denote the coefficient of t in f.
Here, we recall the definition of Macaulay matrices. Let $\succ$ be a monomial order on $\mathrm{Mon}(R[x])$. Writing above F and T as $F = \{f_1, \ldots, f_m\}$ and $T = \{t_1, \ldots, t_\ell\}$ with $t_1 \succ \cdots \succ t_\ell$, we define the *Macaulay matrix* $\mathrm{Mac}_\succ(F, T)$ of F with respect to T as an $(m \times \ell)$-matrix over R whose (i, j)-entry is the coefficient of t_j in f_i, say

$$\mathrm{Mac}_\succ(F,T) := \begin{array}{c} \\ f_1 \\ \vdots \\ f_m \end{array} \!\!\begin{array}{c} \begin{array}{ccc} t_1 & \cdots & t_\ell \end{array} \\ \begin{pmatrix} \mathrm{coeff}(f_1,t_1) & \cdots & \mathrm{coeff}(f_1,t_\ell) \\ \vdots & & \vdots \\ \mathrm{coeff}(f_m,t_1) & \cdots & \mathrm{coeff}(f_m,t_\ell) \end{pmatrix} \end{array}.$$

When $\succ$ is clear from the context, we simply denote it by $\mathrm{Mac}(F, T)$.
Conversely, for an $(m \times \ell)$-matrix $A = (a_{i,j})$ over R and for T given as above, let $\mathrm{Mac}_\succ^{-1}(A, T)$ (or $\mathrm{Mac}^{-1}(A, T)$ simply) denote a unique list F' of polynomials in $R[x]$ such that $\mathrm{Mac}_\succ(F', T) = A$, namely, we set $g_i := \sum_{j=1}^{\ell} a_{i,j} t_j$ for $1 \leq i \leq m$, and $\mathrm{Mac}_\succ^{-1}(A, T) := \{g_1, \ldots, g_m\}$.

Example 1. Consider the following three quadratic polynomials (over $R = \mathbb{Z}$) in two variables x_1 and x_2:

$$\begin{aligned} f_1 &= 5x_1^2 + 6x_1x_2 + 4x_1 + 5x_2 + 3, \\ f_2 &= 4x_1^2 + 5x_1x_2 + 3x_2^2 + 6x_1 + 2x_2 + 2, \\ f_3 &= 2x_1^2 + 4x_1x_2 + 2x_2^2 + 6x_1 + x_2 + 2. \end{aligned}$$

Putting $F := (f_1, f_2, f_3)$, we construct a Macaulay matrix of the shift $S := T_1 \cdot F = \{x_i f_j : 1 \leq i \leq 2,\ 1 \leq j \leq 3\}$ of F by T_1, where T_1 is the set of monomials in x_1 and x_2 of degree one. We order elements of S as follows: $S = \{x_1f_1, x_1f_2, x_1f_3, x_2f_1, x_2f_2, x_2f_3\}$. Let $\succ_{glex}$ be the graded lex order on the monomials in x_1 and x_2 with $x_1 \succ x_2$, that is, $x_1^{\alpha_1}x_2^{\alpha_2} \succ_{glex} x_1^{\beta_1}x_2^{\beta_2}$ if $|\alpha_1+\alpha_2| > |\beta_1+\beta_2|$, or $|\alpha_1+\alpha_2| = |\beta_1+\beta_2|$ and $x_1^{\alpha_1}x_2^{\alpha_2}$ is greater than $x_1^{\beta_1}x_2^{\beta_2}$ with respect to the lexicographical order with $x_1 \succ x_2$. When we order elements of $T_{\leq 3}$ (which is the set of monomials in $R[x]$ of degree ≤ 3) by $\succ_{glex}$, the Macaulay matrix $\mathrm{Mac}_{\succ_{glex}}(S, T_{\leq 3})$ of S with respect to $T_{\leq 3}$ is given as follows:

$$\mathrm{Mac}_{\succ_{glex}}(S, T_{\leq 3}) = \begin{array}{c} \\ x_1f_1 \\ x_1f_2 \\ x_1f_3 \\ x_2f_1 \\ x_2f_2 \\ x_2f_3 \end{array} \!\!\begin{array}{c} \begin{array}{cccccccccc} x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & x_1^2 & x_1x_2 & x_2^2 & x_1 & x_2 & 1 \end{array} \\ \begin{pmatrix} 5 & 6 & 0 & 0 & 4 & 5 & 0 & 3 & 0 & 0 \\ 4 & 5 & 3 & 0 & 6 & 2 & 0 & 2 & 0 & 0 \\ 2 & 4 & 2 & 0 & 6 & 1 & 0 & 2 & 0 & 0 \\ 0 & 5 & 6 & 0 & 0 & 4 & 5 & 0 & 3 & 0 \\ 0 & 4 & 5 & 3 & 0 & 6 & 2 & 0 & 2 & 0 \\ 0 & 2 & 4 & 2 & 0 & 6 & 1 & 0 & 2 & 0 \end{pmatrix} \end{array}.$$

In the XL algorithm in Subsection 2.2, the reduced row echelon form of a Macaulay matrix of a shift of F is computed, with R a finite field $\mathbb{F}_q$. This corresponds to computing a basis G of the $\mathbb{F}_q$-vector space generated by the shift, and clearly the computed basis also generates the ideal $\langle F \rangle_{\mathbb{F}_q[x]}$, i.e., $\langle G \rangle_{\mathbb{F}_q[x]} = \langle F \rangle_{\mathbb{F}_q[x]}$. In general, G computed as above is not necessarily a Gröbner basis of $\langle F \rangle_{\mathbb{F}_q[x]}$, but we will review in Appendix A that for sufficiently large shifts, G becomes a Gröbner basis.

2.2 XL algorithm

This subsection briefly reviews *the XL algorithm* (which stands for eXtended Linearizations), which is proposed in [8] by Courtois et al. to find a solution to a system of multivariate polynomials over finite fields. We write down the XL algorithm in Algorithm 1 below, where the notations are the same as in the previous subsections. We also suppose that the input system is zero-dimensional (see Proposition 2 for the definition). Note also that the input polynomials are assumed to be all quadratic as in the original paper [8], but in fact, their idea is applicable to a general multivariate system of higher degree.

Algorithm 1 (XL, [8, Section 3, Definition 1])

Input: An MQ system $F = (f_1, \ldots, f_m) \subset \mathbb{F}_q[x_1, \ldots, x_n]^m$, and a degree bound D.

Output: A solution over $\mathbb{F}_q$ to $f_i(x_1, \ldots, x_n) = 0$ for $1 \leq i \leq m$.

*(1) **Multiply:** Computing all the products $t \cdot f_i$ with $t \in T_{\leq D-2}$, construct the shift $I_{\leq D} := T_{\leq D-2} \cdot F$ of F by $T_{\leq D-2}$.*

*(2) **Linearize:** Make the Macaulay matrix $A := \mathrm{Mac}_{\succ}(I_{\leq D}, T_{\leq D})$ with respect to some elimination monomial order $\succ$ such that all the terms containing one variable (say x_n) are eliminated last. Compute the reduced row echelon form B of A, and put $G := \mathrm{Mac}_{\succ}^{-1}(B, T_{\leq D})$. A univariate polynomial $g(x_n)$ in x_n of degree at most D is surely contained in G when D is sufficiently large (see Subsection 2.3 or Appendix A for details).*

(3) ***Solve:*** *Compute the roots in $\mathbb{F}_q$ of g by e.g., combining square-free, distinct-degree and equal-degree factorization algorithms such as [32], [19] and [18] respectively.*

(4) ***Repeat:*** *Substitute a root into x_n, simplify the equations of G, and then find the values of the other variables.*

Note that in the generation of $\mathrm{Mac}_{\succ}(I_{\leq D}, T_{\leq D})$, one can sort elements in $I_{\leq D}$ arbitrarily. The condition of the degree bound D for the success of XL is discussed in the next subsection.

2.3 Degree bounds for the success of XL

Algorithm 1 has an input parameter D called a degree bound, and it is known that the algorithm surely finds a zero of $\langle F \rangle$ for sufficiently large D. This subsection reviews bounds on such D both in theory and in practice.

A well-known (theoretical) upper bound is *Dubé's degree bound* [12] given by $D(n,d) := 2\left((d^2/2)+d\right)^{2^{n-1}}$, where $d := \max\{\deg(f_i) : 1 \leq i \leq m\}$: The reduced row echelon form of $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$ yields a Gröbner basis of $\langle F \rangle$ with respect to an elimination order, for any degree D larger than or equal to the Dubé's bound, see Corollary 1 in Appendix A. Hence, for such a D one can obtain a root (in fact *all* roots) of F with Algorithm 1 (cf. Proposition 2 below). However, this bound is not practical and we recall a practical bound given in [27] in the following.

We here review the estimation of this practical bound, considering the rank of $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$ with some reasonable assumption. We may suppose for Algorithm 1 to use an elimination order such that $x_n^D, x_n^{D-1}, \ldots, x_n, 1$ are listed at the end. It is straightforward that the last nonzero row vector of the reduced row echelon form of $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$ yields a univariate equation of x_n if $\mathrm{rank}(\mathrm{Mac}(I_{\leq D}, T_{\leq D}))$ is larger than the number of columns minus $D+1$, i.e.,

$$\mathrm{rank}(\mathrm{Mac}(I_{\leq D}, T_{\leq D})) \geq |T_{\leq D}| - D. \tag{2.1}$$

Thus it suffices to estimate the value of the minimal D satisfying (2.1). For this, we estimate $\mathrm{rank}(\mathrm{Mac}(I_{\leq d}, T_{\leq d}))$ for $d \geq 2$ as the dimension of the $\mathbb{F}_q$-linear space $\langle I_{\leq d} \rangle_{\mathbb{F}_q}$ generated by the set $I_{\leq d}$. To obtain the dimension of $\langle I_{\leq d} \rangle_{\mathbb{F}_q}$, we consider the linear dependency of multiples of f_i and monomials in $x_1, \ldots, x_n$. Writing each f_i as

$$f_i(\mathbf{x}) = \sum_{k \leq \ell} a_{k,\ell}^{(i)} x_k x_\ell + \sum_k b_k^{(i)} x_k + c^{(i)}$$

with $a_{k,\ell}^{(i)}, b_k^{(i)}, c^{(i)} \in \mathbb{F}_q$ for $1 \leq k \leq \ell \leq n$, it follows from $f_i f_j = f_j f_i$ that

$$\begin{aligned}&\sum_{k \leq \ell} a_{k,\ell}^{(i)} (x_k x_\ell f_j) + \sum_k b_k^{(i)} (x_k f_j) + c^{(i)} f_j \\ &= \sum_{k \leq \ell} a_{k,\ell}^{(j)} (x_k x_\ell f_i) + \sum_k b_k^{(j)} (x_k f_i) + c^{(j)} f_i\end{aligned} \tag{2.2}$$

for each $1 \le i \le j \le m$. The equality (2.2) means that a set of polynomials $\{t \cdot f_\ell \mid t \in T_{\le 2},\, \ell \in \{i,j\}\}$ is linearly dependent over $\mathbb{F}_q$. Furthermore, equations obtained by multiplying the both sides of (2.2) by monomials in $x_1, \ldots, x_n$ indicate linear dependencies at degree larger than 2. Assuming no other source of dependencies than the above, the dimension of $\langle I_{\le d} \rangle_{\mathbb{F}_q}$ is determined as follows:

Proposition 1 ([27, Proposition 1]). *For all $d < \min\{q, D_{reg}\}$, if all dependencies of $I_{\le d}$ result from the dependency of $\{t \cdot f_\ell \mid t \in T_{\le 2},\, \ell \in \{i,j\}\}$, then we have*

$$|T_{\le d}| - \dim_{\mathbb{F}_q}(\langle I_{\le d} \rangle_{\mathbb{F}_q}) = \operatorname{coeff}\left((1-t)^{m-n-1}(1+t)^m, t^d\right).$$

Here D_{reg} is given by

$$D_{reg} := \min\left\{d \mid \operatorname{coeff}\left((1-t)^{m-n-1}(1+t)^m, t^d\right) \le 0\right\}, \qquad (2.3)$$

and it is called the degree of regularity for XL.

It follows from Proposition 1 that the minimum D required for the success of XL is given by

$$D = \min\left\{d \mid \operatorname{coeff}\left((1-t)^{m-n-1}(1+t)^m, t^d\right) \le d\right\}, \qquad (2.4)$$

in the case where q is sufficiently large. We call this degree D the *solving degree of XL*. Note that if n and m are determined, then the solving degree can be computed. One can easily confirm that the solving degree tends to be much smaller than Dubé's degree bound (e.g., the solving degree of XL on systems with $n = 10$ and $m = 11$ is 11, whereas Dubé's degree bound on the same system is approximately 10^{309}).

Remark 1. Ars et al. compared XL with Gröbner basis algorithms such as F_4 and F_5 in [2], and obtained the same estimation as in (2.4) for the minimal value of D for which XL succeeds, under some assumptions. Specifically, they described XL as a redundant variant of F_4, assuming that the input system has only one solution over a finite field. They also estimated the minimal value D as (2.4), by considering Matrices appearing in the execution of (matrix-)F_5 on the input system of XL: While Proposition 1 assumes that all dependencies come from trivial relations (2.2), they assumed that the input system is *semi-regular*. The relationship between these two assumptions has not been clarified (as long as we searched related references), but the same estimation (2.4) can be obtained in both cases.

2.4 Complexity

In this subsection, we estimate the time complexity of (plain) XL together with that of its variant Wiedemann XL (WXL). Here WXL uses Wiedemann's algorithm [24] instead of Gaussian elimination in the XL framework, which was first analyzed in [29]. Wiedemann's algorithm generally solves sparse linear systems more efficiently than Gaussian elimination.

Complexity of XL We first consider plain XL (Algorithm 1), where the **Linearize** step is clearly dominant in terms of the time complexity. Recall from Subsection 2.2 that XL outputs a solution of the input system for D equal to or larger than the solving degree given in (2.4), and thus we may assume to take D to be this solving degree. In the **Linearize** step, one uses linear algebra to obtain the reduced row echelon form of a Macaulay matrix with $m \cdot \binom{n+D-2}{D-2}$ rows and $\binom{n+D}{D}$ columns. However, in fact, the cost of this step can be estimated as that of Gaussian elimination on a matrix with $\binom{n+D}{D}$ rows and columns, assuming the following practical assumption as in [23]: If we pick rows at random under the constraint that we have enough equations at each degree $d \leq D$, then usually we have a linearly independent set. From this assumption, by regarding the complexity of Gaussian elimination as that of LU-decomposition, the complexity of XL is roughly estimated as

$$O\left(\binom{n+D}{D}^{\omega}\right), \tag{2.5}$$

where $2 \leq \omega < 3$ is the constant in the complexity of matrix multiplication.

Complexity of WXL According to [11], the complexity of WXL is estimated as

$$O\left(\binom{n}{2} \cdot \binom{n+D}{D}^{2}\right), \tag{2.6}$$

where D is the solving degree of XL given in (2.4). (We remove the constant part from the complexity in [11], since we focus on asymptotic complexity.) WXL consumes less memory than the plain XL, since it can deal with the Macaulay matrix as a sparse matrix, and its memory consumption is estimated as $O\left(\binom{n}{2} \cdot \binom{n+D}{D}\right)$, see [24].

2.5 Improving XL via hybrid approach

One of the most effective improvements of XL is to apply the *hybrid approach* [4, 28], which is the best known technique for solving the MQ problem. The hybrid approach combines an exhaustive search with an MQ solver, and it was proposed in [4] (resp. [28]) for Gröbner basis algorithms such as F_4 and F_5 (resp. XL). Specifically, given an MQ system of m equations in n variables, the values for k $(0 \leq k \leq n)$ variables are randomly guessed and fixed before an MQ solver is applied to the system in the remaining $n-k$ variables; this is repeated until a solution is obtained. The hybrid approach for XL presented in [28] is called FXL, where "F" stands for "fix", and it is constructed by adding the first and last steps below into Algorithm 1:

Algorithm 2 (Hybrid approach with XL (h-XL))

Input: An MQ system $F = (f_1, \ldots, f_m) \subset \mathbb{F}_q[x_1, \ldots, x_n]^m$*, the number* k *of guessed variables, and a degree bound* D.

Output: A solution over $\mathbb{F}_q$ *to* $f_i(x_1, \ldots, x_n) = 0$ *for* $1 \leq i \leq m$.

(1) ***Fix:*** *Fix the values for the* k *variables* $x_1, \ldots, x_k$ *randomly.*

(2) ***Multiply:*** *Construct the shift $I_{\leq D} := T_{\leq D-2} \cdot F$.*
(3) ***Linearize:*** *Compute the reduced row echelon form of* $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$.
(4) ***Solve:*** *Compute the root of a univariate polynomial obtained in* ***Linearize****.*
(5) ***Repeat:*** *Find the values of the other variables.*
(6) *If there exists no solution, return to (1)* ***Fix****.*

The complexities of the hybrid approaches using the plain XL and WXL as MQ solvers are estimated as

$$O\left(q^k \cdot \binom{n-k+D}{D}^{\omega}\right), \tag{2.7}$$

$$O\left(q^k \cdot \binom{n-k}{2} \cdot \binom{n-k+D}{D}^{2}\right), \tag{2.8}$$

respectively, by using the estimations (2.5) and (2.6). Here D is the solving degree of XL on systems of m equations in $n-k$ variables. In the use of the hybrid approach, the number k of guessed variables is chosen such that the function inside brackets in (2.7) or (2.8) takes the minimum value.

3 Main Algorithm

In this section, we propose a new variant of the XL algorithm solving the MQ problem of m equations in n variables over $\mathbb{F}_q$ where $n \leq m$. We first discuss Macaulay matrices over polynomial rings, and second describe the outline of our proposed algorithm "polynomial XL (PXL)". After that, the details of the most technical step will be described in Subsection 3.3, and degree bounds for the success of PXL will be discussed in Subsection 3.4. Throughout this section, let $F = (f_1, \ldots, f_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ be an MQ system of m polynomials in n variables $x_1, \ldots, x_n$ over $\mathbb{F}_q$, where q is a power of a prime.

3.1 Macaulay matrices over polynomial rings

In this subsection, we fix the notations that are used in the rest of this section. In particular, we construct a Macaulay matrix *over the polynomial ring* $\mathbb{F}_q[x_1, \ldots, x_k]$ with respect to $x_{k+1}, \ldots, x_n$ for $1 \leq k \leq n$, where each entry belongs to $\mathbb{F}_q[x_1, \ldots, x_k]$. Namely, a Macaulay matrix whose coefficient ring is $\mathbb{F}_q[x_1, \ldots, x_k]$ will be constructed. Such a Macaulay matrix, together with our construction, plays a key role in the main algorithm in Subsection 3.2 below. Note that most of the notations given below are similar to those defined in Subsection 2.1 for the case where the coefficient ring is a general ring.
In the following, an integer k is fixed, unless otherwise noted. Similarly to the hybrid approach reviewed in Subsection 2.5, the main algorithm divides $x_1, \ldots, x_n$ into k variables $x_1, \ldots, x_k$ and the remaining $n-k$ variables, and then regards $f_1, \ldots, f_m$ as elements of the polynomail ring

$(\mathbb{F}_q[x_1, \ldots, x_k])[x_{k+1}, \ldots, x_n]$. As in Subsection 2.1, we define subsets T_a, $T_{a;b}$, $T_{\leq a}$, I_a, $I_{a;b}$ and $I_{\leq a}$ of $(\mathbb{F}_q[x_1, \ldots, x_k])$ $[x_{k+1}, \ldots, x_n]$ as follows:

$$\begin{aligned} T_a &:= \left\{ x_{k+1}^{\alpha_{k+1}} \cdots x_n^{\alpha_n} : \sum_{i=k+1}^{n} \alpha_i = a \right\}, \\ T_{a;b} &:= T_a \cup T_{a+1} \cup \cdots \cup T_b, \\ T_{\leq a} &:= T_{0;a} \end{aligned}$$

for $b \geq a \geq 0$, and

$$\begin{aligned} I_a &:= \bigcup_{i=1}^{m} \{t \cdot f_i : t \in T_{a-2}\}, \\ I_{a;b} &:= I_a \cup I_{a+1} \cup \cdots \cup I_b, \\ I_{\leq a} &:= I_{2;a} \end{aligned}$$

for $b \geq a \geq 2$. In particular, $I_{\leq a}$ is the shift of F by the set $T_{\leq a-2}$ of monomials in $x_{k+1}, \ldots, x_n$ of degree $\leq a - 2$.
Here, we construct a Macaulay matrix of the shift $I_{\leq D}$ with respect to $T_{\leq D}$ for each $D \geq 2$, as in the plain XL. For this, unlike the plain XL, we use a *graded* monomial order (e.g., graded lexicographic order), which is a monomial order first comparing the total degree of two monomials. Furthermore, as for the order of elements in $I_{\leq D}$, we also use an order that first compares the degree of two polynomials.
For simplicity of notation, we denote by $\mathcal{PM}$ the Macaulay matrix $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$ constructed as above, and call it a *Macaulay matrix of F at degree D over $\mathbb{F}_q[x_1, \ldots, x_k]$*. For two integers d_1 and d_2 ($2 \leq d_1 \leq D$, $0 \leq d_2 \leq D$), we also denote by $\mathcal{PM}[I_{d_1}, T_{d_2}]$ the submatrix of $\mathcal{PM}$ whose rows (resp. columns) correspond to polynomials of I_{d_1} (resp. monomials of T_{d_2}). Then, $\mathcal{PM}$ is clearly divided by submatrices $\mathcal{PM}[I_{d_1}, T_{d_2}]$ $(2 \leq d_1 \leq D, 0 \leq d_2 \leq D)$.
Thanks to our choice of a monomial order together with the quadraticity of F, the following lemma holds clearly:

Lemma 1. *For an MQ system F and positive integers $k \leq n$ and $D \geq 2$, let $\mathcal{PM}$ be a Macaulay matrix of F at degree D over $\mathbb{F}_q[x_1, \ldots, x_k]$. Then, for $2 \leq d \leq D$, every $\mathcal{PM}[I_d, T_{d'}]$ with $d' \notin \{d, d-1, d-2\}$ is a zero matrix, and all elements of $\mathcal{PM}[I_d, T_d]$ belong to $\mathbb{F}_q$.*

Proof. The first statement comes from the fact that $t \cdot f_i \in I_d$ with $t \in T_{d-2}$ includes only monomials with degree d, $d-1$, and $d-2$. Furthermore, if the second one does not hold, then the degree of a given polynomial is larger than 2. $\square$

Due to this lemma, we can partly perform row reduction on $\mathcal{PM}$, which is a key operation of the proposed algorithm in the next subsection.

3.2 Outline of our algorithm PXL

This subsection describes the proposed algorithm polynomial XL (PXL). As in the h-XL described in Subsection 2.5, PXL first sets the first k

variables $x_1, \ldots, x_k$ as guessed variables, whereas the main difference between our PXL and h-XL is the following: While h-XL performs row reduction after substituting actual k values to $x_1, \ldots, x_k$, PXL *partly* performs Gaussian elimination *before* fixing k variables. These manipulations are possible due to our construction of Macaulay matrices over $\mathbb{F}_q[x_1, \ldots, x_k]$ described in Lemma 1.
Here, we give the outline of PXL. The notations are same as those in Subsection 3.1.

Algorithm 3 (Polynomial XL)
Input: An MQ system $F = (f_1, \ldots, f_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$*, the number* k *of guessed variables, and a degree bound* D.
Output: A solution over $\mathbb{F}_q$ *to* $f_i(x_1, \ldots, x_n) = 0$ *for* $1 \leq i \leq m$.

(1) ***Multiply****: Compute the set* $I_{\leq D}$ *of all the products* $t \cdot f_i$ *with* $t \in T_{\leq D-2}$.
(2) ***Linearize(1)****: Generate* $\mathcal{PM} := \mathrm{Mac}(I_{\leq D}, T_{\leq D})$*, which is the Macaulay matrix of* F *at degree* D *over* $\mathbb{F}_q[x_1, \ldots, x_k]$*, and partly perform Gaussian elimination on it. (The details will be described in Subsection 3.3.)*
(3) ***Fix****: Fix the values for the* k *variables* $x_1, \ldots, x_k$ *in the resulting matrix of step 2.*
(4) ***Linearize(2)****: Compute the row echelon form of the resulting matrix of step 3.*
(5) ***Solve****: If step 4 yields a univariate polynomial, compute its root.*
(6) ***Repeat****: Substitute the root, simplify the equations, and then repeat the process to find the values of the other variables.*
(7) If there exists no solution, return to (3) ***Fix****.*

Note that the definition of the resulting matrix of step 2 is given in the following paragraph.
Let us here describe only the first two steps, since the last four steps are executed similarly to h-XL. The **Multiply** step generates the shift $I_{\leq D}$ of F by $T_{\leq D-2}$, defined in Subsection 3.1, by regarding each polynomial as that in $(\mathbb{F}_q[x_1, \ldots, x_k])[x_{k+1}, \ldots, x_n]$. At the beginning of the **Linearize(1)** step, $\mathcal{PM}$ is a polynomial matrix with entries in the polynomial ring $\mathbb{F}_q[x_1, \ldots, x_k]$, but by Lemma 1 it is almost upper-block triangular, and all of its (nearly-)diagonal blocks are matrices with entries in $\mathbb{F}_q$. By utilizing this property, the **Linearize(1)** step repeats to transform such a block into the row echelon form and to eliminate entries of its upper blocks. After the **Linearize(1)** step, the resulting Macaulay matrix is supposed to be the following form $\begin{pmatrix} I & * \\ 0 & A \end{pmatrix}$, by interchanging rows (and columns). Here I is an identity matrix, and A is a matrix over $\mathbb{F}_q[x_1, \ldots, x_k]$. Then, the last four steps deal with only the submatrix composed of rows and columns including no leading coefficient of the reduced part, which corresponds to A. This submatrix A is called the *resulting matrix of* **Linearize(1)**.

3.3 Details of Linearize(1) step

In this subsection, we describe the details of the **Linearize(1)** step in the proposed algorithm, and show that it works well as row operations on $\mathcal{PM}$. We use the same notations as in Subsection 3.1. In the following, we also denote by $\mathcal{PM}[I_a, T_b]$ the same part even after $\mathcal{PM}$ is transformed. The **Linearize(1)** step is mainly performed on each $\mathcal{PM}[I_d, T_{(d-2);d}]$, starting from $d = D$ down to 2. Each iteration d consists of the following three substeps:

(d)-1. Perform Gaussian elimination on $\mathcal{PM}[I_d, T_d]$.

(d)-2. Perform the same row operations as those of (d)-1 on the submatrix $\mathcal{PM}[I_d, T_{(d-2);(d-1)}]$.

(d)-3. Using the *leading coefficients* of the resulting $\mathcal{PM}[I_d, T_d]$, eliminate the corresponding columns of $\mathcal{PM}$. Here, a leading coefficient is the leftmost nonzero entry in each row of the row echelon form of a matrix.

Here, we show that the **Linearize(1)** step described above works well as row operations on $\mathcal{PM}$. Note that for any $3 \leq d \leq D$, the (d)-3 step does not affect the submatrix $\mathcal{PM}[I_{\leq(d-1)}, T_d]$, since $\mathcal{PM}[I_{\leq(d-1)}, T_d]$ is always a zero matrix by Lemma 1. This indicates that $\mathcal{PM}[I_d, T_{\leq D}]$ does not change from the original structure at the beginning of the (d)-1 step. Therefore, from Lemma 1, the manipulations in the (d)-1 and (d)-2 steps can be performed correctly and seen as row operations on $\mathcal{PM}$. Furthermore, the (d)-3 step can be also performed correctly, since the leading coefficients of the resulting $\mathcal{PM}[I_d, T_d]$ belong to $\mathbb{F}_q$. As a result, we have that all the manipulations are practicable and regarded as row operations on $\mathcal{PM}$.

After the **Linearize(1)** step, all manipulations are performed on the resulting matrix of **Linearize(1)** obtained by concatenating rows and columns including no leading coefficient of the row echelon form $\mathcal{PM}[I_d, T_d]$ with $2 \leq d \leq D$.

3.4 Degree bounds for the success of PXL

This subsection estimates the minimum value D where PXL succeeds in finding a solution, under the same assumption as in Proposition 1. We call this minimum value D the *solving degree of PXL*. Specifically, we show that the solving degree of PXL can be upper bounded by the degree of regularity for XL on systems of m equations in $n-k$ variables (cf. (2.3)). Note that the success of PXL means the following: For some evaluation of $(x_1, \ldots, x_k)$ to $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{F}_q^k$ in the **Fix** step, the remaining steps finds a solution $(a_{k+1}, \ldots, a_n) \in \mathbb{F}_q^{n-k}$ to the multivariate system in $x_{k+1}, \ldots, x_n$ corresponding to the resulting matrix of the **Linearize(1)** step, and then $(a_1, \ldots, a_n)$ is exactly a solution to the original system.

To estimate the solving degree of PXL, we first discuss the rank of the resulting matrix of **Linearize(1)**. Recall from Subsection 3.2 that the **Linearize(1)** step transforms the Macaulay matrix into a matrix of the form $\begin{pmatrix} I & * \\ 0 & A \end{pmatrix}$, by interchanging rows (and columns). Here I is an identity matrix, and A is a matrix over $\mathbb{F}_q[x_1, \ldots, x_k]$. The resulting matrix of the

Linearize(1) step is A, and let α be the number of columns of A. For $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{F}_q^k$, we denote by $A^{(\mathbf{a})}$ (resp. $\mathrm{Mac}(I_{\leq D}, T_{\leq D})^{(\mathbf{a})}$) the matrix obtained by substituting $(a_1, \ldots, a_k)$ to $(x_1, \ldots, x_k)$ in A (resp. $\mathrm{Mac}(I_{\leq D}, T_{\leq D})$). Since an evaluation of $x_1, \ldots, x_k$ and elementary row operations over $\mathbb{F}_q[x_1, \ldots, x_k]$ (without multiplying rows by elements in $\mathbb{F}_q[x_1, \ldots, x_k]$ of degree ≥ 1) are commutative, we have the following:

Lemma 2. *With notation as above, we have* $\alpha - \mathrm{rank}(A^{(\mathbf{a})}) = |T_{\leq D}| - \mathrm{rank}(\mathrm{Mac}(I_{\leq D}, T_{\leq D})^{(\mathbf{a})})$.

Furthermore, we here assume the following to estimate the solving degree of PXL

Expectation 1 *For any* $(a_1, \ldots, a_k) \in \mathbb{F}_q^k$, $F(a_1, \ldots, a_k, x_{k+1}, \ldots, x_n)$ *satisfies the same assumption about dependencies of* $I_{\leq d}$ *as in Proposition 1.*

We here suppose that the **Linearize(2)** step is performed with an elimination order on monomials corresponding to column indices of the resulting matrix of **Linearize(1)**, which include $1, x_n, \ldots, x_n^{d'}$ with $d' \leq D$. Then, due to Lemma 2 and the above conjecture, the solving degree D' of PXL is given as

$$D' = \min \left\{ D \mid \mathrm{coeff}\left((1-t)^{m-n-1} (1+t)^m, t^D \right) \leq d' \right\},$$

similarly to the solving degree of XL (2.4). If we use the degree D' obtained from the above equation, then the **Linearize(2)** step yields a univariate equation composed of monomials $1, x_n, \ldots, x_n^{d'}$. We then have

$$D' \leq D_{reg}$$

from (2.3), and thus the solving degree of PXL is upper bounded by D_{reg} on systems of m equations in $n-k$ variables. Indeed, we experimentally confirmed that PXL finds a solution at D_{reg}. Note that D_{reg} is the same as the degree obtained by the solving degree of XL in most cases.

Remark 2. If we assume that the zero-dimesional system $F(x_1, \ldots, x_n)$ has only one solution $(a_1, \ldots, a_n)$ over $\mathbb{F}_q$ as in [2], then we need not to use elimination order in the **Linearize(2)** step. Indeed, in that case, the ideal generated by $F(a_1, \ldots, a_k, x_{k+1}, \ldots, x_n)$ has the reduced Gröbner basis $\{x_{k+1} - a_{k+1}, \ldots, x_n - a_n\}$ (resp. $\{1\}$) with respect to the graded order deduced from that on $\mathbb{F}_q[x_1, \ldots, x_n]$ for valid (resp. invalid) $(a_1, \ldots, a_k)$, similarly to the discussion in [2, Section 4.2]. This implies that the reduced row echelon form of $\mathrm{Mac}(I_{\leq D}, T_{\leq D})^{(\mathbf{a})}$ yields the reduced Gröbner basis if $|T_{\leq D}| - \mathrm{rank}(\mathrm{Mac}(I_{\leq D}, T_{\leq D})^{(\mathbf{a})}) \leq 1$, where elements in $T_{\leq D}$ are ordered by the graded order as in the **Linearize(1)** step. Thus we can expect that PXL finds the unique solution at D_{reg}.

Remark 3. As in the XL algorithm, in practice, PXL randomly chooses approximately $|T_{\leq D}|$ independent rows from the Macaulay matrix with

$|I_{\leq D}|$ rows, and executes the **Linearize(1)** step on the submatrix composed of chosen row vectors. We then assume that the rank of the resulting matrix of **Linearize(1)** is large enough to yield a univariate equation, and this was experimentally confirmed.

Remark 4. We here briefly discuss the relationship between our algorithm PXL and XFL [8, 27] proposed as a variant of h-XL. XFL is roughly described as follows: First, the k variables to be fixed are chosen and generate a shift of the given system by all monomials in the remaining $n-k$ variables up to some degree $D-2$. Second, construct a Macaulay matrix (over $\mathbb{F}_q$, but not over $\mathbb{F}_q[x_1, \ldots, x_k]$) of the shift with respect to all monomials in the whole n variables up to the degree D, and then eliminate only monomials of degree D including only the $n-k$ variables. Third, substitute actual values for the k variables, and execute XL for a system in $n-k$ variables obtained by the substitution.
The first step of XFL clearly coincides with the **Multiply** step of our PXL. The main difference of XFL from PXL is the second step: The second step of XFL eliminates monomials in the $n-k$ variables of degree D, and it corresponds to eliminating only $\mathcal{PM}[I_D, T_D]$ in the second step of our PXL (in fact, PXL eliminates every block $\mathcal{PM}[I_d, T_d]$ with $2 \leq d \leq D$). Therefore, PXL can be regarded as the extension of XFL, and the size of the uneliminated part of the second step of XFL is larger than that of PXL.

4 Complexity

In this section, we first estimate the size of the resulting matrix of **Linearize(1)**. After that, we estimate the time complexity of PXL and compare it with those of h-XL and h-WXL. We take D to be the degree of regularity for XL so that PXL can find a solution (as described in Subsection 3.4).

4.1 Size of resulting matrix of Linearize(1)

Let α be the number of columns of the resulting matrix of **Linearize(1)**. In the following, we estimate the value of this α and show that it can be quite smaller than the number of the columns of the original Macaulay matrix $\mathcal{PM}$. We also show that the resulting matrix of **Linearize(1)** can be assumed to be an $\alpha \times \alpha$ matrix.
For each $2 \leq d \leq D$, we define the set I_d^* of polynomials by

$$I_d^* := \{\text{the degree } d \text{ part of } a \mid a \in I_d\}.$$

If we denote by $\langle I_d^* \rangle_{\mathbb{F}_q}$ the $\mathbb{F}_q$-linear space generated by the set I_d^*, then the number of columns eliminated in the step (d)-1 of **Linearize(1)** on $\mathcal{PM}[I_d, T_d]$ is equal to the rank of $\mathcal{PM}[I_d, T_d]$, that is $\dim_{\mathbb{F}_q}(\langle I_d^* \rangle_{\mathbb{F}_q})$.

Therefore, we have

$$\begin{aligned}\alpha &= |T_{\leq D}| - \sum_{d=2}^{D} \dim_{\mathbb{F}_q}(\langle I_d^* \rangle_{\mathbb{F}_q}) \\ &= \sum_{d=2}^{D} \left(|T_d| - \dim_{\mathbb{F}_q}(\langle I_d^* \rangle_{\mathbb{F}_q})\right) + |T_1| + |T_0|. \qquad (4.1)\end{aligned}$$

Using the same assumption as in Proposition 1, the value of (4.1) can be estimated as

$$\sum_{d=0}^{D} \max\left\{\text{coeff}\left((1-t)^{m-(n-k)}(1+t)^m, t^d\right), 0\right\}. \qquad (4.2)$$

Note that this can be quite smaller than $\binom{n+D}{D}$, which is the number of the columns of the whole Macaulay matrix $\mathcal{PM}$. For example, when $n = m = 40$ and $k = 10$, the solving degree D of PXL obtained by (2.3) is 10, and then α and $\binom{n+D}{D}$ are approximately 2^{21} and 2^{30}, respectively. Recall from Remark 3 that PXL randomly chooses approximately $|T_{\leq D}|$ independent rows from the Macaulay matrix. When $\tilde{I}_d$ denotes the subset of I_d including polynomials corresponding to randomly chosen rows and r_d denotes the rank of $\mathcal{PM}[\tilde{I}_d, T_d]$, suppose that $\tilde{I}_d$ satisfies the following equality

$$\sum_{d=2}^{D} \left(|\tilde{I}_d| - r_d\right) \approx \alpha. \qquad (4.3)$$

This can be realized by avoiding choosing too many rows from I_D, and, by doing so, the size of the resulting matrix of **Linearize(1)** is approximately $\alpha \times \alpha$.

4.2 Time complexity

In this subsection, we estimate the time complexity of PXL. Here, $C_{(d)1}$ (resp. $C_{(d)2}$, $C_{(d)3}$) denotes the estimation of the sum of the number of manipulations in $\mathbb{F}_q$ required for each $(d)-1$ (resp. $(d)-2$, $(d)-3$) in the **Linearize(1)** step with $2 \leq d \leq D$. Furthermore, C_{fix} and C_{li2} denote the estimation of the numbers of manipulations in $\mathbb{F}_q$ required for the **fix** and **Linearize(2)** steps, respectively. These estimations are determined from the number n of all variables, the number k of guessed variables, the degree D of regularity for XL, and the size α of the resulting matrix of **Linearize(1)**. After obtaining each of these five estimations, we give a practical estimation of total time complexity by (4.7) below.

Time Complexity of (d)*-1* Recall that the (d)-1 step performs Gaussian elimination on $\mathcal{PM}[\tilde{I}_d, T_d]$, and its complexity is given as $\max\{|\tilde{I}_d|, |T_d|\}^\omega$ for each $2 \leq d \leq D$. Since we have $\sum_{d=2}^{D} ||\tilde{I}_d| - |T_d|| \leq \alpha$ from (4.3), an upper bound on the sum of the complexity estimation of the (d)-1 step for $2 \le d \le D$ is given by

$$\begin{aligned}\sum_{d=2}^{D} \max\{|\tilde{I}_d|, |T_d|\}^{\omega} &\le \left(\sum_{d=2}^{D} \max\{|\tilde{I}_d|, |T_d|\}\right)^{\omega} \\ &\le (|T_{\le D}| + \alpha)^{\omega} \\ &\le (2 \cdot |T_{\le D}|)^{\omega} = O\left(\binom{n-k+D}{D}^{\omega}\right),\end{aligned}$$

and thus we set $C_{(d)1}$ to be $\binom{n-k+D}{D}^{\omega}$.

Time Complexity of (d)*-2* In each (d)-2 step, the complexity of executing the same row operations as those in (d)-1 step is estimated as that of multiplying an $|\tilde{I}_d| \times |\tilde{I}_d|$ matrix over $\mathbb{F}_q$ to the matrix $\mathcal{PM}[\tilde{I}_d, T_{(d-2);(d-1)}]$. Note that $\mathcal{PM}[\tilde{I}_d, T_{(d-2);(d-1)}]$ is a sparse matrix from the same discussion as in Subsection 3.3, where each row of it has at most $n-k+1$ non-zero entries. Thus, multiplying the two matrices are done by $O((n-k) \cdot |\tilde{I}_d|^2)$ additions and scalar multiplications in $\mathbb{F}_q[x_1, \ldots, x_k]$. Since polynomials appearing in each addition or scalar multiplication have degree ≤ 2, its cost is bounded by $O\left(\binom{k+2}{2}\right)$ with naive approach. Considering above together, each (d)-2 step has complexity $O\left(\binom{k+2}{2} \cdot (n-k) \cdot |\tilde{I}_d|^2\right)$, and hence the total complexity of (d)-2 for all $2 \le d \le D$ is given by

$$\begin{aligned}\sum_{d=2}^{D} \left(\binom{k+2}{2} \cdot (n-k) \cdot |\tilde{I}_d|^2\right) &\le \binom{k+2}{2} \cdot (n-k) \cdot |\tilde{I}_{\le D}|^2 \\ &= O\left(k^2 \cdot (n-k) \cdot \binom{n-k+D}{D}^2\right),\end{aligned}$$

and thus $C_{(d)2}$ is set to be $k^2 \cdot (n-k) \cdot \binom{n-k+D}{D}^2$.

Time Complexity of (d)*-3* To estimate the time complexity of (d)-3 with $2 \le d \le D$, we use the following lemma:

Lemma 3. *At the time of executing the* (d)*-3 step with* $2 \le d \le D-1$*, the degree of every element of* $\mathcal{PM}[\tilde{I}_{(d+1);D}, T_d]$ *is lower than or equal to* $D-d$*.*

Proof. By the induction, we prove that, at the time of the (d)-3 step, the degree of every element of $\mathcal{PM}[\tilde{I}_{(d+1);D}, T_d]$ and $\mathcal{PM}[\tilde{I}_{(d+1);D}, T_{d-1}]$ is lower than or equal to $D-d$ and $D-d+1$, respectively. In the case where $d = D-1$, the above statement clearly holds. In the following, we show that, if the statement holds when $d = d'$ with $3 \le d' \le D-1$, then it also holds when $d = d'-1$. Before executing the step (d')-3, $\mathcal{PM}[\tilde{I}_{(d'+1);D}, T_{d'-2}]$ is a zero matrix clearly. Then, the (d')-3 step adds row vectors, which are obtained by multiplying rows corresponding to $\tilde{I}_{d'}$ by a polynomial with the degree $D-d'$, to rows corresponding to $\tilde{I}_{(d'+1);D}$. Here, the degree of each entry of $\mathcal{PM}[\tilde{I}_{d'}, T_{d'-1}]$ and $\mathcal{PM}[\tilde{I}_{d'}, T_{d'-2}]$ are at most 1 and 2, respectively. Hence, through (d')-3, the degree of each entry of $\mathcal{PM}[\tilde{I}_{(d'+1);D}, T_{d'-2}]$ becomes at most $D-d'+2$ and that of $\mathcal{PM}[\tilde{I}_{(d'+1);D}, T_{d'-1}]$ remains at most $D-d'+1$, Therefore, the statement holds in the case where $d = d'-1$, as desired. □

Each (d)-3 step eliminates the corresponding columns using the leading coefficients of $\mathcal{PM}[\tilde{I}_d, T_d]$. This complexity is estimated as that of multiplying the following two matrices: **(A)** The submatrix of $\mathcal{PM}[\tilde{I}_{d'}, T_d]$ $(d+1 \le d' \le D)$ to be eliminated, and **(B)** The submatrix of $\mathcal{PM}[\tilde{I}_d, T_{(d-2);d}]$ consisting of columns with no leading coefficients and rows with leading coefficients. If we suppose for the efficiency that each (d)-3 step only eliminates elements of rows including no leading coefficients in $\mathcal{PM}[\tilde{I}_{d'}, T_{d'}]$ $(d+1 \le d' \le D)$, then the sizes of these matrices **(A)** and **(B)** are estimated as follows:

(A) The number of rows is at most α from the discussion in Subsection 4.1 and that of columns is equal to the rank r_d of $\mathcal{PM}[\tilde{I}_d, T_d]$.

(B) The number of rows is r_d and that of columns is upper-bounded by $|T_{(d-2);d}| = O(|T_d|)$.

Considering these estimation for **(A)** and **(B)** together with Lemma 3, the complexity of each (d)-3 step is given by that of multiplying an $\alpha \times r_d$ matrix of elements in $\mathbb{F}_q[x_1, \ldots, x_k]$ with degree $\le D-d$ and an $r_d \times |T_d|$ matrix of elements in $\mathbb{F}_q[x_1, \ldots, x_k]$ with degree ≤ 2. Note that each multiplication of an element in $\mathbb{F}_q[x_1, \ldots, x_k]$ with degree $\le D-d$ and that with degree ≤ 2 can be done in $O\left(\binom{k+2}{2} \cdot \binom{k+D-d}{D-d}\right)$ with a naive approach. Putting it all together, we estimate the complexity of the (d)-3 step as

$$O\left(\binom{k+D-d}{D-d} \cdot \binom{k+2}{2} \cdot \alpha \cdot r_d \cdot |T_d|\right) \le O\left(\binom{k+D-d}{D-d} \cdot \binom{k+2}{2} \cdot \alpha \cdot \binom{n-k+d-1}{d}^2\right).$$

Note that the (D)-3 step can be omitted since $\mathcal{PM}[\tilde{I}_{\le(D-1)}, T_D]$ is a zero matrix. Consequently, the sum of the complexity of the (d)-3 step for $2 \le d \le D-1$ is estimated by

$$\begin{aligned}
&\sum_{d=2}^{D-1} \left(\binom{k+D-d}{D-d} \cdot \binom{k+2}{2} \cdot \alpha \cdot \binom{n-k+d-1}{d}^2\right) \\
&\le \binom{k+2}{2} \cdot \alpha \cdot \left(\sum_{d=2}^{D-1} \binom{n-k+d-1}{d}\right) \cdot \left(\sum_{d=2}^{D-1} \left(\binom{k+D-d}{k} \cdot \binom{n-k+d-1}{n-k-1}\right)\right) \\
&\le O\left(k^2 \cdot \alpha \cdot \binom{n-k+D}{D} \cdot \binom{n+D}{D}\right),
\end{aligned}$$

and thus we set $C_{(d)3}$ to be $k^2 \cdot \alpha \cdot \binom{n-k+D}{D} \cdot \binom{n+D}{D}$.

*Time Complexity of **Fix*** The size of the resulting matrix of **Linearize(1)** is approximately $\alpha \times \alpha$ due to the discussion in Subsection 4.1, and the degree of every element in the matrix is lower than or equal to D from Lemma 3. Therefore, the time complexity of **Fix** is estimated as that of substituting k values to $x_1, \ldots, x_k$ in α^2 polynomials with degree D in $\mathbb{F}_q[x_1, \ldots, x_k]$. When we use a naive approach, the complexity of evaluation of a polynomial with degree d in n variables is estimated by $\binom{n+d}{d}$. Therefore, C_{fix} is given by

$$C_{\mathsf{fix}} = q^k \cdot \alpha^2 \cdot \binom{k+D}{D}, \tag{4.4}$$

since the **Fix** step is iterated for any values of $x_1, \ldots, x_k$.

Table 1. Complexities approximated by power of 2 between PXL (4.7), h-XL (2.7), and h-WXL (2.8), the optimal number k of guessed variables of PXL, the solving degree D of PXL, and the estimated size α of the resulting matrix of **Linearize(1)** on the MQ system with $n = m = 20$, 40, 60, and 80 over $\mathbb{F}_{2^8}$ (above) and over $\mathbb{F}_{31}$ (below)

$\mathbb{F}_{2^8}$	20	40	60	80
h-XL	2^{75}	2^{134}	2^{194}	2^{252}
h-WXL	2^{75}	2^{129}	2^{182}	2^{234}
PXL	$\mathbf{2^{62}}$	$\mathbf{2^{117}}$	$\mathbf{2^{169}}$	$\mathbf{2^{220}}$
k	3	6	8	10
D	9	14	19	24
α	2^{14}	2^{27}	2^{42}	2^{56}

$\mathbb{F}_{31}$	20	40	60	80
h-XL	2^{66}	2^{119}	2^{170}	2^{221}
h-WXL	2^{65}	2^{116}	2^{162}	2^{208}
PXL	$\mathbf{2^{57}}$	$\mathbf{2^{105}}$	$\mathbf{2^{152}}$	$\mathbf{2^{197}}$
k	5	8	11	13
D	7	12	16	21
α	2^{11}	2^{24}	2^{37}	2^{51}

*Time Complexity of **Linearize(2)*** The **Linearize(2)** step performs Gaussian elimination on an $\alpha \times \alpha$ matrix over $\mathbb{F}_q$, and thus we estimate C_{li2} by

$$C_{\mathsf{li2}} = q^k \cdot \alpha^{\omega}, \tag{4.5}$$

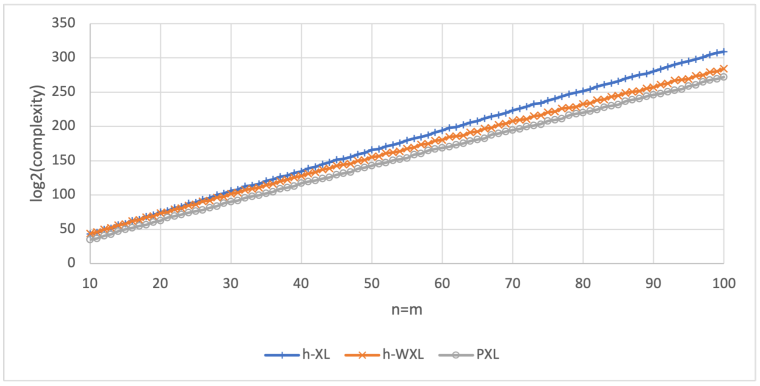
considering q^k times iterations.

Rough Estimations of Time Complexity Here, we present a more compact formula for the time complexity of PXL. Comparing the estimations $C_{(d)2}$ and $C_{(d)3}$, we can easily confirm that the value of $C_{(d)3}$ is larger than that of $C_{(d)2}$. Furthermore, comparing the estimations $C_{(d)1}$ and $C_{(d)3}$, we experimentally confirmed that, for the case where $10 \le n \le 100$, $m = n$, $1.5n$, $2n$, and k is the value minimizing the sum of the above five estimations, the value of $C_{(d)3}$ is always much larger than that of $C_{(d)1}$ (e.g., $C_{(d)1}$ and $C_{(d)3}$ on the case of $n = m = 100$ with $q = 2^8$ is approximately 2^{210} and 2^{259}, respectively). These facts indicate that the complexity of the **Linearize(1)** step is dominated by $C_{(d)3}$ for the practical cases, and it is estimated as follows:

$$O\left(k^2 \cdot \alpha \cdot \tbinom{n-k+D}{D} \cdot \tbinom{n+D}{D}\right). \tag{4.6}$$

By using this fact, the time complexity of PXL is roughly estimated by $C_{(d)3} + C_{\mathsf{fix}} + C_{\mathsf{li2}}$, say

$$O\left(k^2 \cdot \alpha \cdot \tbinom{n-k+D}{D} \cdot \tbinom{n+D}{D} + q^k \cdot \left(\alpha^2 \cdot \tbinom{k+D}{D} + \alpha^{\omega}\right)\right). \tag{4.7}$$



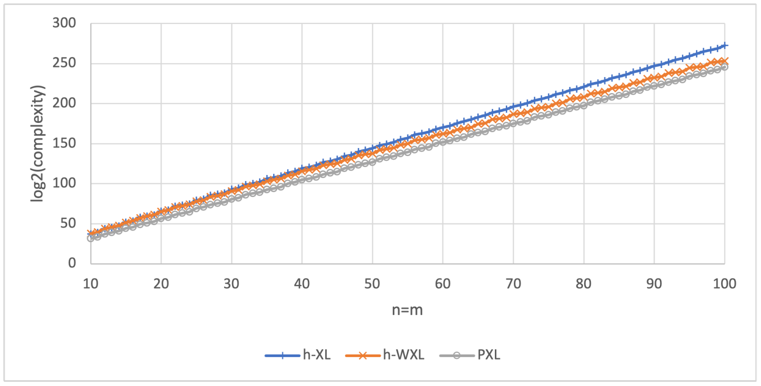


Fig. 1. Comparison of complexities approximated by power of 2 between PXL (4.7), h-XL (2.7), and h-WXL (2.8) on the MQ system with $10 \leq m = n \leq 100$ over $\mathbb{F}_{2^8}$ (above) and over $\mathbb{F}_{31}$ (below)

4.3 Comparison

We compare the complexity of our PXL with those of h-XL and h-WXL, with our motivation towards contribution of PXL to evaluating the security of MPKCs. Following the security estimation of [11], we choose h-WXL among the XL family as a target for comparison. We also refer the complexity of h-XL on which h-WXL is originally based (in fact, h-XL is the most basic method in the framework of the hybrid approaches with XL). Recall that the complexities of h-XL, h-WXL, and PXL are estimated by (2.7), (2.8), and (4.7), respectively. Note that, for fixed n, m, and q, each of the three approaches chooses the number k of guessed variables so that its complexity estimation becomes the smallest value, and thus the value of k depends on each approach. Furthermore, we here use $\omega = 2.37$ following [16]. As we will see below, PXL is theoretically more efficient than h-XL and h-WXL in the case of $n = m$ (this is the case that hybrid approaches for the MQ problem works most efficiently).

Table 1 and Figure 1 compare the bit complexities of PXL, h-XL, and h-WXL on the MQ system of m equations in n variable with $n = m$ over $\mathbb{F}_{2^8}$ and $\mathbb{F}_{31}$. These orders of the finite fields are chosen following the MQ challenge [30], and in particular, $q = 2^8 = 256$ is also suggested as a parameter of [11]. Note also that we do not choose $q = 2$ since exhaustive searches are known to be effective in this case. Furthermore, Table 1 shows the bit complexities of the three approaches, the optimal k of PXL minimizing the value of (4.7), the solving degree D of PXL obtained from (2.3), and the estimated size α of the resulting matrix of **Linearize(1)** obtained from (4.2) for the case where $n = m$ is set to be 20, 40, 60 and 80. For example, in the case where $q = 2^8$ and $n = m = 80$, the complexities of h-XL, h-WXL, and PXL are approximately estimated as 2^{252}, 2^{234}, and 2^{220}, respectively. As a result, we see that PXL has the less complexity than those of h-XL and h-WXL in the case of $n = m$; we expect that the similar results will be obtained in other finite fields from the form of the complexity estimation (4.7).
On the other hand, we confirmed that PXL is not efficient in highly overdetermined cases. This is because, in such overdetermined cases, k is set to be a very small value for efficiency.

Remark 5 (Space Complexity). The space complexities of h-XL and h-WXL are estimated by $O\left(\binom{n-k+D}{D}^2\right)$ and $O\left(\binom{n-k}{2} \cdot \binom{n-k+D}{D}\right)$, respectively. The memory space consumed by our PXL is upper-bounded by $O\left(\binom{k+D}{D} \cdot \binom{n-k+D}{D}^2\right)$, since the degree of every element of the Macaulay matrix and its transformed matrices in **Linearize(1)** is at most D through an execution of PXL from Lemma 3. These estimations cannot be directly compared to each other, since the values of the following two parameters depend on one's choice of an algorithm: The solving degree D and the number k of fixed values.
On the other hand, focusing on the sparsity/density of matrices, we predict that PXL is not efficient compared with h-WXL in terms of the space complexity for the following reason: Through the elimination process of Macaulay matrices, WXL can deal with a Macaulay matrix as a sparse matrix due to Wiedemann's algorithm, whereas PXL holds some dense submatrices. Considering this together with the time complexities for practical parameters, we conclude that the relationship between PXL and h-WXL would be a trade-off between time and memory.

5 Experimental Results

We implemented the proposed algorithm PXL in the Magma computer algebra system (V2.26-10) [5], in order to examine that it behaves as our complexity estimation provided in Section 4. (As it will be described below, note that our current implementation is not optimized one, see also Remark 6.) We also confirmed in our experiments that PXL outputs a solution correctly at the solving degree of PXL given in Subsection 3.4. First, we confirmed that the **Linearize(1)** step behaves as in (4.6). The reason why we focus on the behavior of the **Linearize(1)** step is the

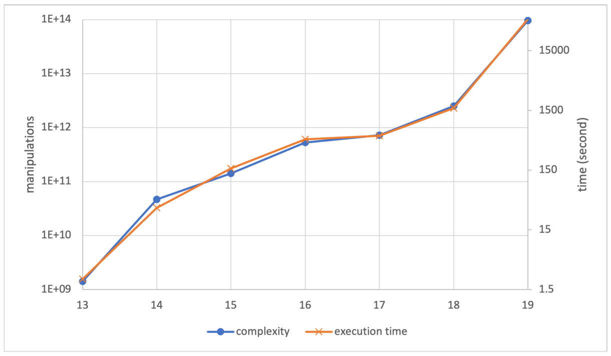


Fig. 2. Comparison between the estimation of complexity by (4.6) and the execution time of the **Linearize(1)** step on an $\mathcal{MQ}$ system with $n = m$ over $\mathbb{F}_{2^4}$

following: In the estimation (4.7) of the total time complexity, only $C_{(d)3}$ is specific to our estimation in theory, while the later parts C_{fix} and C_{li2} for the **Fix** and **Linearize(2)** steps just come from the known complexity estimation. Figure 2 compares the execution time of the **Linearize(1)** step and the bit complexity (4.6) on the system with $n = m$ from $n = 13$ to $n = 19$ over $\mathbb{F}_{2^4}$, and the number k of fixed variables is chosen so as to minimize the value of (4.7). As a result, Figure 2 shows that the execution time and our estimation (4.6) have almost the same behavior, which indicates that the estimation (4.6) would be reliable.

On the other hand, our current Magma implementation of the **Fix** and **Linearize(2)** steps does not show the similar behavior as our complexity estimation, due to the use of unoptimized implementation. For example, in the case of $n = m = 16$ with $k = 5$, **Linearize(1)**, **Fix**, and **Linearize(2)** took 10 min., 40 hr., and 30 min., respectively, whereas the estimated numbers of manipulations of these three steps from (4.6), (4.4), and (4.5) are 2^{39}, 2^{44}, and 2^{39}, respectively. We observe that this inefficiency of the latter two steps (in particular **Fix** with a lot of for-loops) is due to the use of Magma's interpreter language. Using compiler languages such as C instead could be a solution to resolve this problem, but we must newly implement the arithmetic of matrices and polynomials efficiently, which is not the topic of this paper. We leave such an efficient implementation with compiler languages to future work.

Remark 6. We remark that here we do not compare the execution time of our PXL with that of any other variant of XL, since the practical behavior deeply depends on how one implements the arithmetic of matrices (and polynomials) efficiently, which is not the topic of this paper. For a fair comparison, providing optimized implementations of several variants including PXL is required, and it is a very important task for practical cryptanalysis.

6 Conclusion

We presented a new variant of XL, which is a major approach for solving the MQ problem. Our proposed polynomial XL (PXL) eliminates the linearized monomials in polynomial rings to solve the system efficiently, and this paper estimates its complexities.
Given an MQ system of m equations in n variables, the proposed algorithm first regards each polynomial in n variable as that in $n-k$ variables $x_{k+1}, \ldots, x_n$, whose coefficients belong to the polynomial ring $\mathbb{F}_q[x_1, \ldots, x_k]$. We then generate a Macaulay matrix over $\mathbb{F}_q[x_1, \ldots, x_k]$, and partly perform the row reduction (Gaussian elimination). Finally, random values are substituted for the k variables, and the remaining part of the (partly-reduced) Macaulay matrix is transformed into the reduced row echelon form. Partly reducing the (polynomial) Macaulay matrix is done mainly on submatrices over $\mathbb{F}_q$ (not over $\mathbb{F}_q[x_1, \ldots, x_k]$) with arithmetic of polynomials in $\mathbb{F}_q[x_1, \ldots, x_k]$ of bounded degree, and the remaining part has size much smaller than the original one. This construction reduces the amount of manipulations for each guessed value compared to h-XL. This paper also presents an asymptotic estimation of the time complexity, which shows that the proposed algorithm solves the system faster in theory for the case of $n \approx m$ than both h-XL and h-WXL. On the other hand, PXL might be less efficient than h-WXL with respect to the space complexity.
This paper discusses only the quadratic case, but, as in the plain XL, the proposed algorithm can be also generalized to higher degree cases. Therefore, one considerable future work is to analyze the complexity of PXL on such higher degree systems. Furthermore, for a comparison of the practical time-efficiencies of our PXL and other XL variants, it is important to implement PXL (and the other variants) efficiently. In our experiments, we implemented PXL over Magma, but this can be more optimized by using an alternative (compiler) programming language, e.g., C. Therefore, to provide such an optimized code for PXL is a challenging task.

Acknowledgements The authors thank Tsuyoshi Takagi and Kazuhiro Yokoyama for many helpful comments and suggestions on this work. The authors are also grateful to Kosuke Sakata for his advice on the implementation of our proposed algorithm.

References

1. M.-R. Albrecht, C. Cid, J.-C. Faugère, and L. Perret. On the relation between the mxl family of algorithms and gröbner basis algorithms. *J. Symb. Comput.*, 47(8):926–941, 2012.
2. G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between xl and gröbner basis algorithms. In *ASIACRYPT 2004*, pages 338–353. Springer, 2004.
3. M. Aschenbrenner and A. Leykin. Degree bounds for gröbner bases in algebras of solvable type. *J. Pure Appl. Algebra*, 213:1578–1605, 2009.

4. L. Bettale, J.-C. Faugère, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.*, 3:177–197, 2009.
5. W. Bosma, J. Cannon, and C. Playoust. The magma algebra system. I. The user language. *J. Symb. Comput.*, 24(3-4):235–265, 1997.
6. B. Buchberger. *Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal.* PhD thesis, Universität Innsbruck, 1965.
7. A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. Gemss signature schemes proposal for nist pqc project (round 3 version), 2020.
8. N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT 2000*, pages 392–407. Springer, 2000.
9. D.-A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry.* Springer, second edition edition, 2005.
10. D.-A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms.* Springer, fourth edition edition, 2015.
11. J. Ding, M.-S. Chen, M. Kannwischer, J. Patarin, A. Petzoldt, D. Schmidt, and B.-Y. Yang. Rainbow signature schemes proposal for nist pqc project (round 3 version), 2020.
12. T.-W. Dubé. The structure of polynomial ideals and gröbner bases. *SIAM J. Comput.*, 19(4):750–773, 1990.
13. J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f4). *J. Pure Appl. Algebra*, 139(1-3):61–88, 1999.
14. J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *ISSAC 2002*, pages 75–83. ACM, 2002.
15. J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.
16. F.L̃. Gall. Algebraic complexity theory and matrix multiplication. In *ISSAC 2014*, page 23. ACM, 2014.
17. M.-R. Garey and D.-S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman, 1979.
18. J. v.z̃. Gathen and V. Shoup. Computing frobenius maps and factoring polynomials. *Comput. Complexity*, 2(3):87–224, 1992.
19. E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comp.*, 67(223):1179–1197, 1998.
20. A. Kipnis and A. Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In *CRYPTO 1999*, pages 19–30. Springer, 1999.
21. M. Kudo, S. Harashita, and H. Senda. Automorphism groups of superspecial curves of genus 4 over $\mathbb{F}_{11}$. *J. Pure Appl. Algebra*, 224(9), 2020.
22. D. Lazard. Systems of algebraic equations. In *EUROSAM 1979*, pages 88–94. Springer, 1979.
23. Wael Said Abdelmageed Mohamed. *Improvements for the XL algorithm with applications to algebraic cryptanalysis.* PhD thesis, TU Darmstadt, 2011.

24. D.H̃. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theor.*, 32(1):54–62, 1986.
25. M. Wiesinger-Widi. *Gröbner bases and generalized Sylvester matrices.* PhD thesis, Johannes Kepler University Linz, 2015.
26. W.-T. Wu. Basic principles of mechanical theorem proving in elementary geometries. *J. Autom. Reason.*, 2(3):221–252, 1986.
27. B.-Y. Yang and J.-M. Chen. All in the xl family: theory and practice. In *ICISC 2004*, pages 67–86. Springer, 2004.
28. B.-Y. Yang, J.-M. Chen, and N. Courtois. On asymptotic security estimates in xl and gröbner bases-related algebraic cryptanalysis. In *ICICS 2004*, pages 401–413. Springer, 2004.
29. B.-Y. Yang, O.C.-H. Chen, D.J. Bernstein, and J.-M. Chen. Analysis of quad. In *FSE 2007*, pages 290–308. Springer, 2007.
30. T. Yasuda, X. Dahan, Y.-J. Huang, T. Takagi, and K. Sakurai. Mq challenge: hardness evaluation of solving multivariate quadratic problems, 2015. NIST Workshop on Cybersecurity in a Post-Quantum World.
31. K. Yokoyama, Masaya Yasuda, Yasushi Takahashi, and Jun Kogure. Complexity bounds on semaev' s naive index calculus method for ecdlp. *J. Math. Cryptol.*, 14(1):460–485, 2020.
32. D. Y.Ỹ. Yun. On square-free decomposition algorithm. In *ISSAC 1976*, pages 26–35. ACM, 1976.

A Correctness of the XL algorithm

The correctness of XL (Algorithm 1) means here the following: For sufficiently large D, XL definitely finds one root of the input system F. This correctness holds if F is zero-dimensional, and it can be proved by using a fact that the set G of polynomials computed in Step (2) of Algorithm 1 is a Gröbner basis of $\langle F \rangle$ for D larger than a certain degree bound (a typical bound is Dubé's one [12], see below). Although the proof of this fact might be well-known (see e.g., [22]), let us write down it in this appendix, for the reader's convenience.
In the following, let K be a field, and let $K[x]$ denote the polynomial ring $K[x_1, \ldots, x_n]$ of n variables over K. As in Subsection 2.1, let $\mathrm{Mon}(K[x])$ denote the set of all monomials in $K[x]$ and, for $f \in K[x]$, let $\mathrm{supp}(f)$ denote the supporting set of f, that is, $\mathrm{supp}(f) := \{t \in \mathrm{Mon}(K[x]) : \mathrm{coeff}(f,t) \neq 0\}$. For $f \in K[x] \setminus \{0\}$, we denote by $\mathrm{LT}_{\succ}(f)$ and $\mathrm{LM}_{\succ}(f)$ the leading term and the leading monomial of f with respect to $\succ$, respectively. For a subset $F \subset K[x]$, we set $\mathrm{LT}_{\succ}(F) := \{\mathrm{LT}_{\succ}(f) : f \in F\}$ and $\mathrm{LM}_{\succ}(F) := \{\mathrm{LM}_{\succ}(f) : f \in F\}$. For simplicity, we denote $\mathrm{LT}_{\succ}$ as LT and so on, if $\succ$ is clear from the context.
The following theorem provides a criterion for a reduced row echelon form computed in XL (Algorithm 1) to yield a Gröbner basis of the input system:

Theorem 1 ([25, Theorem 2.3.3]). *Let $\succ$ be a monomial order on $\mathrm{Mon}(K[x])$, and $F = \{f_1, \ldots, f_m\} \subset K[x]$ a set of ordered polynomials. Let H be a Gröbner basis of the ideal $\langle F \rangle \subset K[x]$ with respect to $\succ$,* *and T a finite subset of $\mathrm{Mon}(K[x])$ such that for all $h \in H$, there exist $q_1, \ldots, q_m \in K[x]$ with $h = \sum_{i=1}^{m} q_i f_i$ and $\mathrm{supp}(q_i) \cdot \{f_i\} \subset S := T \cdot F$ for all $1 \leq i \leq m$. Put $A := \mathrm{Mac}_{\succ}(S, T)$, and let B be its reduced row echelon form. Then, $G := \mathrm{Mac}_{\succ}^{-1}(B, T)$ is a Gröbner basis of $\langle F \rangle$ with respect to $\succ$.*

Proof. Put $H = \{h_1, \ldots, h_\ell\}$ with $h_i \in K[x]$ for $1 \leq i \leq \ell$. By our assumption, for each $1 \leq i \leq \ell$, there exist $q_{ij} \in K[x]$ with $1 \leq j \leq m$ such that $h_i = \sum_{j=1}^{m} q_{ij} f_j$ and $\mathrm{supp}(q_{ij}) \cdot \{f_j\} \subset S$ for all $1 \leq j \leq m$. Thus we have

$$\bigcup_{i=1}^{\ell} \bigcup_{j=1}^{m} \mathrm{supp}(q_{ij}) \cdot \{f_j\} = \bigcup_{i=1}^{\ell} \bigcup_{j=1}^{m} \{t \cdot f_j : t \in \mathrm{supp}(q_{ij})\} \subset S,$$

and hence

$$h_i = \sum_{j=1}^{m} \sum_{t \in \mathrm{supp}(q_{ij})} \mathrm{coeff}(q_{ij}, t) \cdot t \cdot f_j \in \sum_{g \in S} K \cdot g,$$

where $\sum_{g \in S} K \cdot g$ denotes the set of all K-linear combinations of finite elements in S. Regarding the Macaulay matrix of $\{h_i\}$ with respect to T as a row vector in $K^{\#T}$, we have that it belongs to the linear space generated by row vectors of $\mathrm{Mac}_{\succ}(S, T)$. Thus, putting $G = \{g_1, \ldots, g_{\ell'}\}$, we can write $h_i = \sum_{k=1}^{\ell'} a_{i,k} g_k$ for some $a_{i,k} \in K$. It follows from the definition of a reduced row echelon form that $\mathrm{LT}_{\succ}(g) \neq \mathrm{LT}_{\succ}(g')$ for $g, g' \in G$ with $g \neq g'$. This implies that for each $1 \leq i \leq \ell$, there exists k such that $\mathrm{LT}_{\succ}(h_i) = a_{i,k} \mathrm{LM}_{\succ}(g_k)$. Therefore $\mathrm{LM}_{\succ}(H) \subset \mathrm{LM}_{\succ}(G)$, by which we have $\langle \mathrm{LT}_{\succ}(\langle F \rangle) \rangle = \langle \mathrm{LT}_{\succ}(H) \rangle \subset \langle \mathrm{LT}_{\succ}(G) \rangle$. From the construction of G, we also have $\langle \mathrm{LT}_{\succ}(G) \rangle \subset \langle \mathrm{LT}_{\succ}(\langle F \rangle) \rangle$, and thus $\langle \mathrm{LT}_{\succ}(\langle F \rangle) \rangle = \langle \mathrm{LT}_{\succ}(G) \rangle$. □

Dubé [12] showed an upper bound on the maximal degree of the reduced Gröbner basis of a homogeneous polynomial ideal. His bound depends on the number of variables n and the maximal degree d of the initial generators for the ideal, but not on any monomial order. In the following theorem (without proof), we state Dubé's bound:

Theorem 2 ([12, Theorem 8.2] or [3, Proposition 5.1]). *Let $\succ$ be an arbitrary monomial order on $\mathrm{Mon}(K[x])$, and $F \subset K[x]$ a finite set of homogeneous polynomials. Put $d := \max\{\deg(f) : f \in F\}$. Then, we have*

$$\mathrm{max.GB.deg}_{\succ}(F) \leq D(n-1, d) := 2\left(\frac{d^2}{2} + d\right)^{2^{n-2}},$$

where $\mathrm{max.GB.deg}_{\succ}(F)$ denotes the maximal degree of elements in the reduced Gröbner basis of $\langle F \rangle$ with respect to $\succ$.

The following corollary deduced from Theorem 2 provides a degree bound of Gröbner bases for the in-homogeneous case, and consequently it follows that XL (Algorithm 1) can compute a Gröbner basis of the input system:

Corollary 1 (cf. [3, Corollary 5.4]). *Let* $F = \{f_1, \ldots, f_m\} \subset K[x]$ *be a finite set of (possibly in-homogeneous) polynomials, and put* $d := \max\{\deg(f) : f \in F\}$*. Then, for every monomial order* $\succ$*, there exists a Gröbner basis* H *of* $\langle F \rangle$ *with respect to* $\succ$ *such that:*

– *For every* $h \in H$*, there exist* $q_1, \ldots, q_m \in K[x]$ *with* $h = \sum_{i=1}^{m} q_i f_i$*, and*

$$\deg(q_i f_i) \leq D(n,d) := 2\left(\frac{d^2}{2} + d\right)^{2^{n-1}}$$

for all $1 \leq i \leq m$*.*

Hence, for every D *with* $D \geq D(n,d)$*, the set* G *of polynomials computed in Step (2) of Algorithm 1 with the input* (F, D) *is a Gröbner basis of* $\langle F \rangle$*.*

Proof. For each $f \in K[x]$, we denote by f^h its homogenization by an extra variable y, that is,

$$f^h(x_1, \ldots, x_n, y) := y^{\deg(f)} f(x_1/y, \ldots, x_n/y) \in K[x, y],$$

and put $F^h := \{f^h : f \in F\} \subset K[x, y]$. Applying Theorem 2 to the ideal $\langle F^h \rangle$, we have

$$\text{max.GB.deg}_{\succ}(F^h) \leq D(n,d) = 2\left(\frac{d^2}{2} + d\right)^{2^{n-1}}.$$

It is well-known (e.g., [3, Corollary 3.5] or [31, Proposition 9]) that, for the reduced Gröbner basis G for $\langle F^h \rangle$ with respect to a suitable extension of $\succ$, the set

$$G|_{y=1} := \{g(x_1, \ldots, x_n, 1) : g \in G\}$$

is a Gröbner basis for the original ideal $\langle F \rangle$. For every $g \in G$, there exist $q_{g,i} \in K[x, y]$ with $1 \leq i \leq m$ such that $g = \sum_{i=1}^{m} q_{g,i} f_i^h$. Since g and $f_1^h, \ldots, f_m^h$ are all homogeneous, we may suppose that $q_{g,1}, \ldots q_{g,m}$ are also homogeneous, and

$$\deg(q_{g,i} f_i^h) \leq \deg(g) \leq D(n,d).$$

for any $1 \leq i \leq m$. Here we set $H := G|_{y=1}$, and let h be an arbitrary element in H. Writing $h = g|_{y=1}$ for some $g \in G$, we then have

$$h = g|_{y=1} = \sum_{i=1}^{m} (q_{g,i})|_{y=1} f_i,$$

with

$$\deg((q_{g,i})|_{y=1} f_i) \leq \deg(q_{g,i} f_i^h) \leq D(n,d).$$

Thus, the assertion holds by putting $q_i := q_{g,i}$. □

We finally prove the correctness of XL (Algorithm 1) with Corollary 1:

Proposition 2 (Correctness of the XL algorithm). *Let* K *be a finite field, and* F *be a finite subset of* $K[x_1, \ldots, x_n]$*. If the ideal* $\langle F \rangle$ *is zero-dimensional, i.e.,* $V(F) = \{(a_1, \ldots, a_n) \in \overline{K}^n : f(a_1, \ldots, a_n) = 0\ (\forall f \in F)\}$ *is finite, then the XL algorithm (Algorithm 1) with inputs*

F and D finds a partial solution $a_n \in V(\langle F \rangle \cap K[x_n])$ to F for every D with $D \geq D(n,d)$, where $d := \max\{\deg(f) : f \in F\}$. Moreover, if one uses a lexicographical order where x_n is the lowest among $x_1, \ldots, x_n$, then the XL algorithm (Algorithm 1) can compute a solution (in fact all solutions) over K to F (if exists).

Proof. By Corollary 1, the set G of polynomials computed in Step (2) of Algorithm 1 is a Gröbner basis G with respect to the elimination monomial order which one adopts. Since $\langle F \rangle$ is zero-dimensional, it is known (e.g., [10, Chapter III, Section 1, Exercise 5]) that G contains a univariate polynomial $g(x_n)$ in $K[x_n] \smallsetminus \{0\}$, and thus a partial solution $a_n \in V(\langle F \rangle \cap K[x_n])$ to F can be obtained by factoring $g(x_n)$. If the order is lexicographical, then it follows from e.g., [21, Lemma 2.3.2] that all solutions over K to F are computed by substituting each root of g to polynomials in $G \smallsetminus \{g\}$. □