Solving Systems of Multivariate Quadratic Equations over Finite Fields or: From Relinearization to MutantXL

Enrico Thomae, Christopher Wolf

Horst Görtz Institute for IT-security
Faculty of Mathematics
Ruhr-University of Bochum, 44780 Bochum, Germany
{enrico.thomae,christopher.wolf}@ruhr-uni-bochum.de,
chris@Christopher-Wolf.de

Abstract. In this article we investigate algorithms for solving non-linear multivariate equations over finite fields and the relation between them. For non binary fields usually computing the Gröbner basis of the corresponding ideal is the best choice in this context. One class of algorithms is based on Buchberger's algorithm. Today's best algorithms like F_4 and F_5 belong to this class. Another strategy to solve such systems is called *eXtended Linearization (XL)* from Eurocrypt 2000. In the past both strategies were treated as different ideas and there was a heated discussion which of them to prefer. Since Ars *et al.* proved in 2004 that XL is a redundant version of F_4, the latter seemed to be the winner. But that was not the end of the line as piece for piece the idea emerged that both classes are only different views on the same problem. We even think that they are just different time-memory optimizations. One indication to that can be found in the PhD of Albrecht, who introduced MatrixF$_5$, a F_5 version of XL. A second indication can be found in the PhD of Mohamed, who introduced a memory-friendly version of XL using Wiedemanns algorithm. We want to give further evidence by providing a theoretical analysis of MutantXL. We show that MutantXL solves at the same degree of regularity as its competitors F_4 and F_5 for most instances. Thereby we also confirm recent results of Albrecht, who showed that MutantXL is a redundant version of F_4, *i.e.* it never solves below the degree of regularity of F_4. We show that MutantXL has, compared to WiedemannXL, to pay its gain in efficiency with memory.
To enhance the understanding of the whole XL-family of algorithms we give a full overview from Relinearization over XL to MutantXL and provide some additional theoretical insights.

1 Introduction

Solving systems of $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic ($\mathcal{MQ}$) equations is hard in general. More precisely, the associated $\mathcal{MQ}$-problem is known to be $\mathcal{NP}$-complete [25]. The security of many cryptosystems relies directly or indirectly on this problem, what makes solving systems of $\mathcal{MQ}$ equations an established tool in cryptanalysis.

Todays best known algorithms to solve such systems are based on a breakthrough result due to Bruno Buchberger [7] in 1965. He used so-called S-polynomials to compute the Gröbner Bases of the ideal spanned by the $\mathcal{MQ}$-system. This algorithm can be seen as a generalization of the well known Gaussian Elimination for linear systems. In 1999 Jean-Charles Faugère proposed a more efficient variant of Buchberger's algorithm by using the Macaulay matrix as well as sparse matrix algebra and called it F_4-algorithm [17]. In 2002 he also got rid of the reductions to zero and published the F_5-algorithm [19]. In practice F_5 and its variants have an impressive track record in bringing down cryptographic systems and challenges [18, 20, 21, 23, 22, 6, 24]. On the theoretical site a complexity analysis of both algorithms was unknown until Bardet *et al.* came up with a solution in 2005 [5]. Before then it was hard to theoretically analyze the complexity of attacks using F_4 or F_5. This might be the reason why Kipnis and Shamir developed a simpler but slower algorithm called *Relinearization* to analyze their cryptanalysis of the HFE public key cryptosystem in 1999 [26]. One year later Courtois *et al.* came up with a similar but scalable algorithm named *eXtended Linearization* or *XL* for short [12]. As the idea is very simple, it turned out that Lazard proposed a very similar algorithm already decades before [27]. Unfortunately the latter provided only a bad upper bound on the complexity and Moh [29] pointed out that the complexity analysis of XL was flawed and way more difficult as for Relinearization. So, at this time both XL and F_5 suffered from a missing complexity analysis. Nevertheless Courtois and Pieprzyk claimed to have broken AES [14] using a variant of XL, called XSL, in 2002. At least since they were disproved by Cid and Leurent [9] only two years later, the community of cryptographers became increasingly reserved against this method, even if the complexity analysis is understood quit well today. This is mainly due to Moh [29], Diem [15] and Yang and Chen [35].

During the last years XL and F_5 were treated as two different algorithms and there was a heated discussion which of them is the better choice for cryptographic problems. For example Ars *et al.* [3] showed 2004 that XL is a redundant version of F_4, *i.e.* it produces more equations than necessary. In 2011 Albrecht *et al.* [2] even showed that MutantXL, one of the most promising variants of XL, is a redundant version of F_4. On the one hand these results suggest that F_4/F_5 is always faster than the XL-family. On the other hand Mohamed *et al.* introduced a variant of XL using Wiedemann's algorithm [32, 31] and thus consuming less memory than F_4.

This immediately suggests one of the questions we want to discuss in this article: Might it be possible that both XL and F_5 are just different views of the same problem? One attempt in this direction is due to Albrecht [1] who started with

the polynomial view of F_5 and ended with the matrix view of XL by introducing Matrix-F_5. Essentially this algorithm applies the F_5 criteria to XL and provides a Gröbner basis as output. Unfortunately Matrix-F_5 is neither faster nor needs less memory than F_5.

1.1 Organization and Achievement

This article gives an overview of the XL-family with a strong link to Gröbner basis algorithms and the most promising variant of XL, namely MutantXL or also called XL2. Our work can be seen in line with the overview of XL-like algorithms of Yang and Chen [35], the summary of XL-algorithms with focus on applying Wiedemanns algorithm in the PhD of Mohamed [31] and the F_5 view on XL in the PhD of Albrecht [1].
First we shortly recap the well known technique of Relinearization in section 3.1. Second we proof that this is a special case of the XL algorithm in section 3.3. Third we shortly introduce most of the variants of XL, with a special focus on MutantXL in section 3.4. Section 4 deals with the question how random systems of $\mathcal{MQ}$equations look like. Section 5 will repeat the complexity analysis of XL. In section 6 we give a complexity analysis of MutantXL and observe that most of the time it solves at the same degree as F_4 and F_5, what somehow confirms the results of Albrecht *et al.* [2]. Section 7 investigates the memory consumption of F_4/F_5, MutantXL and WiedemannXL and thus gives rise to the assumption that they are different time-memory optimizations of the same class of algorithms.

2 Notation

Here we introduce the notation we use throughout the article. As mentioned above, solving non-linear systems of m equations and n unknowns is a difficult problem in general. Restricting to the seemingly easy case of degree 2 equations is equally difficult, as there is a polynomial time reduction. Actually the associated problem to decide if some system is solvable or not, also known as $\mathcal{MQ}$-problem, is proven to be NP-complete [25].

Let $P : \mathbb{F}_q^n \to \mathbb{F}_q^m$ be an $\mathcal{MQ}$-system of the form

$$\begin{aligned} p^{(1)}(x_1, \ldots, x_n) &= 0 \\ p^{(2)}(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ p^{(m)}(x_1, \ldots, x_n) &= 0, \end{aligned} \tag{1}$$

with

$$p^{(k)}(x_1, \ldots, x_n) := \sum_{1 \le i \le j \le n} \gamma_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} \beta_i^{(k)} x_i + \alpha^{(k)}. \tag{2}$$

We call equation $p^{(k)} = 0$ with $p^{(k)}$ defined by (2) inhomogeneous. The homogeneous case consists only of quadratic terms and is thus defined by

$$p^{(k)}(x_1, \ldots, x_n) := \sum_{1 \leq i \leq j \leq n} \gamma_{ij}^{(k)} x_i x_j. \tag{3}$$

We need the classification into homogeneous and inhomogeneous polynomials later on, because it is not always easy to see that formulas for both cases are equal after transforming an inhomogeneous system in a homogeneous one.
Let $\pi^{(k)}$ be the coefficient vector of $p^{(k)}(x_1, \ldots, x_n)$ in lexicographic order, *i.e.*

$$\pi^{(k)} = (\gamma_{11}^{(k)}, \gamma_{12}^{(k)}, \ldots, \gamma_{1n}^{(k)}, \gamma_{22}^{(k)}, \gamma_{23}^{(k)}, \ldots, \gamma_{nn}^{(k)}, \beta_1^{(k)}, \ldots, \beta_n^{(k)}, \alpha^{(k)}),$$

then $\mathit{\Pi}$ denote the coefficient matrix of P

$$\mathit{\Pi} := \begin{pmatrix} \pi^{(1)} \\ \vdots \\ \pi^{(m)} \end{pmatrix}.$$

Denote by $I_{\mathit{\Pi}}$ the number of linearly independent equations of $\mathit{\Pi}$. More formal $I_{\mathit{\Pi}}$ is the dimension of the vector space generated by $\{\pi^{(k)} | 1 \leq k \leq m\}$. Sometimes we just write I instead of $I_{\mathit{\Pi}}$. The number of different monomials occurring in P is denoted by T.

Note that the problem of solving non-linear equations becomes easier if m exceeds n or vice versa. Even if not proven yet, experience suggest that $m = n$ is the hardest instance. The naive algorithm is to solve (1) by linearization, *i.e.* to substitute every monomial in $p^{(k)}$ by a new variable and to solve the obtained linear system of equations $\mathit{\Pi}$ via Gaussian Elimination. This leads to a correct solution if we have $m = \frac{n(n+1)}{2} + n$ linearly independent equations, *i.e.* if the number of linearly independent equations I is equal to the number of monomials T. Otherwise we obtain an exponential number of parasitic solutions. With the technique of Relinearization, introduced in [26], we can solve P (asymptotically) if we have $m \geq 0.09175 \cdot n^2$ linearly independent equations. Lowering the trivial factor of $\frac{1}{2}$ to roughly $\frac{1}{10}$ was a big leap and sufficient to cryptanalyze HFE.

3 From Relinearization to MutantXL

3.1 Relinearization

The idea of Relinearization is very clear and simple. Given a random $\mathcal{MQ}$-system P we first linearize, *i.e.* introduce new variables $y_k := x_i x_j$. For the simplicity of the analysis we assume P to be homogeneous. That means the number of unknowns $x_i x_j$ is $\binom{n+1}{2} = \frac{n(n+1)}{2}$. Notice that this is no restriction as we can express any non-homogeneous system in form of a homogeneous system by introducing one more variable. For random systems it is very likely that all

of the m equations are linearly independent, cf. section 4. This underdetermined system of linear equations is now solved by Gaussian Elimination, see figure 1 for illustration. As we can see there, we obtain $q^{\frac{n(n+1)}{2}-m}$ parasitic solutions in $y_{m+1}, \ldots, y_{\frac{n(n+1)}{2}}$, *i.e.* a number exponential in n.

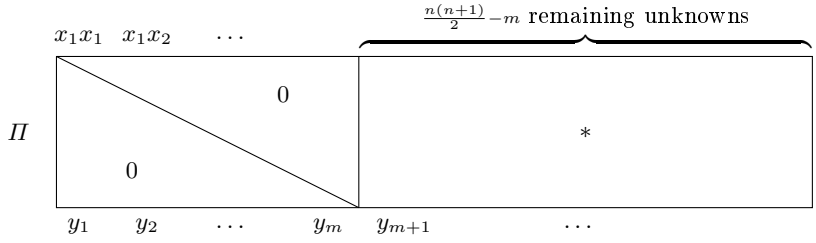

Fig. 1. Coefficient Matrix Π of P after Gaussian elimination

After linearization both $y_1 := x_1x_1$ and $y_2 := x_1x_2$ are two independent linear variables. But from an algebraic point of view this is not true as both y_1 and y_2 depend on x_1. Relinearization exploits this structure to eliminate parasitic solutions, *i.e.* to fix the remaining variables $y_{m+1}, \ldots, y_{\frac{n(n+1)}{2}}$ implicitly via new equations. The following equations are trivially true and also linearly independent for some $y_a = x_ix_j$:

$$\begin{aligned} & x_ix_jx_kx_l = x_ix_kx_jx_l = x_ix_lx_jx_k \qquad (4)\\ \Leftrightarrow \quad & y_{i_1}y_{i_2} = \quad y_{i_3}y_{i_4} \quad = y_{i_5}y_{i_6} \end{aligned}$$

Kipnis and Shamir required $i < j < k < l$ in the above equation. There are $\binom{n}{4}$ possible choices for $x_ix_jx_kx_l$ and thus we get $2\binom{n}{4}$ linearly independent equations by (4). If this is larger than the number of unknowns in the remaining variables y's we are done and can solve the system, *i.e.* for

$$2\binom{n}{4} \geq \binom{\frac{n(n+1)}{2} - m + 1}{2}.$$

For m in the same magnitude as n this is not the case in general. For $m = \varepsilon n^2$ and only considering the n^4 part, we get the following asymptotic equation

$$0 \leq -\varepsilon^2 + \varepsilon - \frac{1}{12}$$

and hence $\varepsilon \geq 0.09175$.

Note that for inhomogeneous equations the overall analysis is the same but with a bigger number of unknowns. By

$$2\binom{n}{4} \geq \binom{\frac{n(n+1)}{2} + n - m + 1}{2},$$

we obtain the same asymptotic result. But later in the exact analysis we will need to distinguish between these two cases, as Relinearization in the homogeneous case is exactly the same as XL of degree 2.

3.2 The XL algorithm

The idea of XL [12] is simpler but not as easy to analyze. Basically we multiply every equation of P by every monomial of a certain degree D. Obviously this preserves the original solution. At some point D is large enough to obtain roughly as many linearly independent equations as monomials and thus we can solve the system by linearization. We call this D *solving degree* and the corresponding degree $D+2$ of the polynomials *saturation degree*. The crucial question is, how many of the produced equations are linearly independent. We will look at this in section 5. First let us define the XL algorithm in a rigorous way.

Definition 1. *Let $P^{inh} := \{p^{(k)} \mid 1 \leq k \leq m\}$ be the set of inhomogeneous quadratic polynomials $p^{(k)}$ as defined in (2) and $P^{hom} := \{p^{(k)} \mid 1 \leq k \leq m\}$ the set of homogeneous quadratic polynomials $p^{(k)}$ as defined in (3). We define the set of all monomials of degree $D \in \mathbb{N}_0$ by*

$$\mathit{Mon}_D(n) := \{\prod_{j=1}^{D} x_{i_j} \mid 1 \leq i_1 \leq i_2 \leq \ldots \leq i_D \leq n\}.$$

Multiplying P^{inh} by all monomials of degree D yields the set

$$\mathit{Blow}_D^{inh}(n) := \{ab \mid a \in \mathit{Mon}_D(n) \text{ and } b \in P^{inh}\}.$$

The set $\mathit{Blow}_D^{hom}(n)$ is defined analogously. The following set describes what is commonly used as XL algorithm of degree D.

$$\mathit{XL}_D^{inh}(n) := \bigcup_{i=0}^{D} \mathit{Blow}_i^{inh}(n).$$

If n is clear out of the context, we just write $\mathit{Mon}_D, \mathit{Blow}_D^{inh}$ or XL_D^{inh}.

Notice that $\mathtt{XL}_D^{\mathtt{inh}}(n)$ and $\mathtt{Blow}_D^{\mathtt{hom}}(n+1)$ are equivalent sets of polynomials and thus we can restrict our analysis to the homogeneous case (see section 5 for a proof). Furthermore we describe the XL algorithm for $\mathcal{MQ}$-systems only. First this is the most important case for cryptanalysis and second substituting the term $D+2$ by $D+\deg(P)$ easily generalizes the algorithm.

Definition 2 (XL algorithm). *First we generate XL_D^{inh} and check if the number of linearly independent equations I is equal to the number of produced monomials T subtracted by $D+2$. In this case we linearize the system and solve it by Gaussian elimination. Notice, if $T - I \leq D+2$ we can choose the order of the monomials such that we obtain at least one univariate equation after Gaussian elimination, which can be solved,* e.g. *by Berlekamp's algorithm. If $T-I > D+2$ we set $D := D+1$ and try again.*

Another way of looking at this algorithm is to first homogenize P and then just produce $\mathtt{Blow}_D^{\mathtt{hom}}(n+1)$. Notice that the corresponding coefficient matrix of $\mathtt{Blow}_D^{\mathtt{hom}}(n+1)$ is also known as Macaulay matrix. We speak of XL of degree $D+2$, referring to the highest total degree of all polynomials and not the solving degree. This degree $D+2$ is called the saturation degree of XL.

3.3 Relinearization as special case of XL

The claim of Relinearization being a special case of XL was already made in the paper that introduced XL [12]. Due to many technical details the proof was omitted and referred to the extended version of the paper, what is to the best of our knowledge not publicly available. But as we use Moh's detailed analysis of Relinearization, *i.e.* $i \leq j \leq k \leq l$ [29], to compare Relinearization with XL, we think our proof differs from what Courtois *et al.* had in their mind.

For $i \leq j \leq k \leq l$ we get

$$2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2}$$

equations by Relinearization, instead of $2\binom{n}{4}$ in the case $i < j < k < l$. To allow to distinguish different cases in the proof we assume m to be of the form $\sum_{i=0}^{\gamma-1}(n-i) = \gamma n + \frac{\gamma-\gamma^2}{2}$ for $\gamma = \varepsilon n$ and thus $m = (\varepsilon - \frac{\varepsilon^2}{2})n^2 + \frac{\varepsilon}{2}n$. Through this $y_{m+1} = x_{\gamma+1}x_{\gamma+1}$ holds and due to the graded lexicographical order for all indices of not specified monomials x_ix_j in the $*$ block, see figure 2, it holds $i, j > \gamma$. This allows us to analyse $x_ix_jx_kx_l$ in the two cases $i \leq \gamma$ and $i > \gamma$.

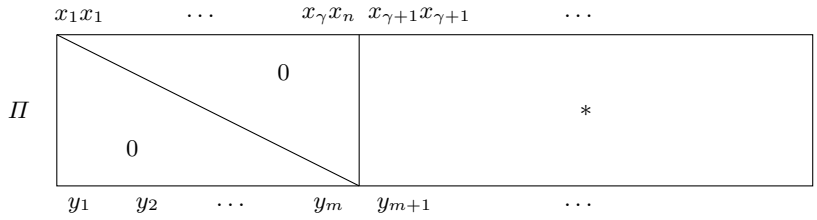

Fig. 2. Coefficient Matrix Π of P after Gaussian elimination

We want to show that multiplying by special monomials is exactly the same as Relinearization. Due to the choice of m we can distinguish two cases.

Case 1, $i \leq \gamma$. For $i \leq \gamma$ Relinearization behaves exactly as XL, as we will show now. Remember Relinearization uses

$$\underbrace{x_i x_j}_{y_{i_1}} x_k x_l = \underbrace{x_i x_k}_{y_{i_2}} x_j x_l = \underbrace{x_i x_l}_{y_{i_3}} x_j x_k \text{ with } i_1, i_2, i_3 \in \{1, \ldots, m\}. \tag{5}$$

As XL do not make use of additional variables y_i, equations (5) are trivially true and thus of no use for solving the original system. But instead we can produce all monomials of (5) by multiplying the row containing y_{i_1}, y_{i_2} and y_{i_3} in figure 2 with $x_k x_l, x_j x_l$ and $x_j x_k$. This way we obtain 3 equations which are obviously linearly independent. To calculate the gain of this, *i.e.* the difference between the number of linearly independent equations and monomials, we have to distinguish 3 more cases.

Case 1.1, $i < j < k < l$. There are $\sum_{i=1}^{\gamma} \binom{n-i}{3}$ possibilities for $x_i x_j x_k x_l$, as well as for $x_i x_k x_j x_l$ and $x_i x_l x_j x_k$ for $i < j < k < l$. So we produce $3 \sum_{i=1}^{\gamma} \binom{n-i}{3}$ equations with XL by multiplying y_i by $x_k x_l$. But we also produce $\sum_{i=1}^{\gamma} \binom{n-i}{3}$ new monomials containing variables x_i with $i \leq \gamma$ and so the number of remaining new equations is $2 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{3}$.

Case 1.2, ($j = i$ and $k \neq l$) or ($j = k$ and $i \neq l$) or ($k = l$ and $i \neq j$). In the case of two equal and two different indices we have the following 3 possibilities of monomials: $x_i x_i x_k x_l$, $x_i x_j x_j x_l$ and $x_i x_j x_k x_k$. Any of them produces 2 equations due to $x_i x_i x_k x_l = x_i x_k x_i x_l = x_i x_l x_i x_k$. Notice that the last equality is not used by Relinearization, because it is trivial. So we ignore them too. Since $x_i x_i x_k x_l$ introduce one new monomial containing variables x_i with $i \leq \gamma$, only $x_i x_k x_i x_l$ gives us a new equation. So we have $3 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{2}$ new equations in total.

Case 1.3, $i = j$ and $k = l$ and $i \neq k$. In this case Relinearization uses $x_i x_i x_k x_k = x_i x_k x_i x_k$. The left monomial produces one new monomial containing variables x_i with $i \leq \gamma$ for every equation in XL and thus only the right monomial produces $\sum_{i=1}^{\gamma} (n - i)$ new equations.

To sum up cases 1.1 - 1.3, we produced

$$2 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{3} + 3 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{2} + \sum_{i=1}^{\gamma} (n-i)$$

new linearly independent equations by adapting Relinearization to XL. Notice that we produced more equations than this, but used them to eliminate the newly introduced monomials of degree 4 with variables x_i and $i \leq \gamma$. So the number of unknowns in XL is only the number of degree 4 monomials containing variables x_i with $i > \gamma$, *i.e.* $\binom{n-\gamma+3}{4}$.

Case 2, $i > \gamma$. For $i > \gamma$ Relinearization uses the equations

$$x_i x_j x_k x_l = x_i x_k x_j x_l = x_i x_l x_j x_k.$$

These equations cannot be rebuilt by XL. But the difference between both methods is that Relinearization introduce more variables after the second linearization step and XL does not. So we do not need these equations for XL because they are only needed in Relinearization to eliminate variables we do not have in XL.

The following equations sum up the number of unknowns and equations in both methods. The terms on the left hand side are the number of unknowns and on the right hand side the number of equations.

Relinearization:

$$\begin{array}{ccc} \frac{(\binom{n+1}{2} - m)(\binom{n+1}{2} - m + 1)}{2} & \leq & 2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} \\ \uparrow & & \uparrow \\ \Delta_1 & & \Delta_2 \\ \downarrow & & \downarrow \end{array}$$

XL:

$$\binom{n-\gamma+3}{4} \leq 2 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{3} + 3 \cdot \sum_{i=1}^{\gamma} \binom{n-i}{2} + \sum_{i=1}^{\gamma} (n-i)$$

To show that both are equal, we have to show that the difference Δ_1 between the terms on the left is equal to the difference Δ_2 of the terms on the right. We us $m = \gamma n + \frac{\gamma - \gamma^2}{2}$ $(*)$ and the following equality for $k \in \mathbb{N}_{>0}$

$$\binom{n}{k} - \binom{n-\gamma}{k} = \sum_{i=1}^{\gamma} \binom{n-i}{k-1}.$$

We get

$$\begin{aligned}
\Delta_1 &= \frac{(\binom{n+1}{2}-m)(\binom{n+1}{2}-m+1)}{2} - \binom{n-\gamma+3}{4} \\
&\overset{(*)}{=} 2\binom{n-\gamma}{4} + 3\binom{n-\gamma}{3} + \binom{n-\gamma}{2} \\
&= 2\left(\binom{n-\gamma}{4} - \binom{n}{4} + \binom{n}{4}\right) \\
&\quad +3\left(\binom{n-\gamma}{3} - \binom{n}{3} + \binom{n}{3}\right) \\
&\quad +\binom{n-\gamma}{2} - \binom{n}{2} + \binom{n}{2} \\
&= 2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} \\
&\quad -2\cdot\sum_{i=1}^{\gamma}\binom{n-i}{3} - 3\cdot\sum_{i=1}^{\gamma}\binom{n-i}{2} - \sum_{i=1}^{\gamma}(n-i) \\
&= \Delta_2 \qquad \square
\end{aligned}$$

To conclude, if we use XL and multiply not by all quadratic monomials, but only by special ones, we do the same as Relinearization and thus Relinearization is a special case of XL. Now we want to show that it is equal in the homogeneous case of degree two.

In section 5 we will explain that the number of linearly independent equations produced by $\mathtt{Blow}_2^{\mathtt{hom}}$ is $m\binom{n+1}{2} - \binom{m}{2}$. Using this we can analyze if XL outperforms Relinearization or not. In the homogeneous case the following must hold for $\mathtt{Blow}_2^{\mathtt{hom}}$ to obtain a solution.

$$m\binom{n+1}{2} - \binom{m}{2} - \binom{n+3}{4} \geq -D-2 \tag{6}$$

The following must hold for Relinearization to obtain a solution.

$$2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} - \binom{\frac{(n+1)n}{2} - m + 1}{2} \geq -D-2 \tag{7}$$

Because of the following equality, inequations (6) and (7) are equal.

$$\begin{aligned}
& m\binom{n+1}{2} - \binom{m}{2} - \binom{n+3}{4} \\
&= \frac{n^4}{24} + \frac{n^3}{4} - \frac{n^2m}{2} + \frac{11n^2}{24} - \frac{nm}{2} + \frac{n}{4} + \frac{m^2}{2} - \frac{m}{2} \\
&= 2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} - \binom{\frac{(n+1)n}{2} - m + 1}{2}
\end{aligned}$$

In the inhomogeneous case, $\texttt{Blow}_2^{\texttt{inh}}$ is slightly better than Relinearization. We gain a factor of $\frac{1}{12}$ instead of 0.09175 in the asymptotic analysis. We can also derive this from the inequations above. If we homogenize the inhomogeneous system, we have to substitute n by $(n+1)$ in inequation (6). Relinearization does not depend on whether equations are homogeneous or not, *i.e.* inequation (7) remains the same and thus both are not longer equal.

3.4 Variants of XL

Inspired by Gröbner bases and some other observations there is a whole family of XL-like algorithms, which try to use some additional ideas to speed up the original XL algorithm. We revisit the most important ones, namely FXL, XFL, XLF, XL', XSL, MutantXL/XL2 and WXL. [6, 11, 12, 13, 16, 30, 35] and give some reasons if and under which circumstances they are useful. See the PhD thesis of Mohamed [31] for a more extensive list of XL variants.

FXL
FXL, or *fixing extended linearization*, was suggested in the original paper of Courtois *et al.* [12] and is XL with guessing some variables beforehand. This is quite a good idea also for the Gröbner base algorithm F_4 [6]. An empirical confirmation for XL was given in [36]. They showed that D is exponential, namely 2^m, in the case $m = n$ and already polynomial in the case $m = n+1$. So it helps to guess at least one variable. The optimal number of guessed variables is given by Yang and Chen [35, section 5.2].

XFL
XFL is a variant of FXL. We choose f variables, but do not guess them in the beginning. We choose the order of the monomials in a way that all monomials containing any of the f variables are eliminated last. Now we linearize the system and apply Gaussian elimination. Because the system was underdetermined, we obtain no unique solution. To do so, we guess one of the f variables and apply Gaussian elimination again. Why is this stepwise guessing better than FXL in some case? First we have to do the most work, i.e. the first Gaussian elimination, only once. In FXL we have to do this after every wrong guessing. But notice, that there the number of monomials is smaller, so we carefully have to calculate the right tradeoff between the two variants. Second XFL may use dependencies among the f variables and thus succeed.

XLF
XLF just takes the *field equations* $(x^q - x) = 0$ in $\mathbb{F}_q$ into account and was first mentioned in [11]. XLF makes sense in the inhomogeneous case, if D gets larger than $(q-2)$. In this case the analysis becomes slightly different, because the number of produced monomials decrease, i.e. monomials x_i^{D+2} reduce to the already existing x_i. This means we need less linearly independent equations to succeed.

XL'
Introduced by Courtois and Patarin in [13] this variant solves the equation system by XL until there are only $\binom{r+D+2}{D+2}$ equations in r variables left. This remaining system of equations is solved by brute force or other algorithms like Gröbner bases.

Claim. For practical purposes, FXL is better than XL'.

Proof. We call FXL better than XL', *i.e.* FXL $\geq$ XL', if $(T-I)_{\text{FXL}}$ is smaller than $(T-I)_{\text{XL}'}$. With section 5 and $D = 2k$ we can write

$$\begin{aligned}(T-I)_{\text{FXL}} &= \binom{n-r+D+1}{D+2} - \sum_{i=0}^{k}(-1)^i\binom{m}{i+1}\binom{n-r+D-2i-1}{n-r-1} \\ &= \sum_{i=0}^{2k+2}(-1)^i\binom{m-n+r}{i}\binom{m}{2k-i+2}\end{aligned}$$

and

$$\begin{aligned}(T-I)_{\text{XL}'} &= \binom{n+D+1}{D+2} - \sum_{i=0}^{k}(-1)^i\binom{m}{i+1}\binom{n+D-2i-1}{n-1} - \binom{r+2k+2}{2k+2} \\ &= \sum_{i=0}^{2k+2}(-1)^i\binom{m-n}{i}\binom{m}{2k-i+2} - \binom{r+2k+2}{2k+2} + 1.\end{aligned}$$

If we plotted formula $(T-I)_{\text{XL}'} - (T-I)_{\text{FXL}}$ we would see that this is greater than zero, *i.e.* FXL is better than XL', for r less than some bound depending on k. For increasing k the bound on r decrease. It seems very hard to calculate this bound in an analytical way. But for real world parameter $k < 10$ and $r \ll n$ we are below this bound. W.l.o.g. we can assume $m = n$, otherwise we substitute r. See table 1 for the upper bound on r depending on m and k. With F_5 we can solve $\mathcal{MQ}$-systems up to $m = 20$ in 2^{66} operations, so we stopped the table at $m = 30$ for practical purpose. Even $k > 6$ is of no practical interest because the workload without considering guessing would be larger than $\binom{n+2k+2}{2k+2}^{\omega}$ for $2 \leq \omega \leq 3$. Note that the cases marked gray are always solvable by $\texttt{XL}^{\texttt{inh}}_{2k}$

$m\backslash k$	1	2	3	4	5	6
5	1	0	0	0	0	0
10	6	3	1	0	0	0
15	11	8	6	5	1	1
20	15	13	12	10	8	6
25	20	18	17	15	12	10
30	25	23	22	19	17	15

Table 1. Upper bound on r such that FXL $\geq$ XL'.

without guessing. In any other case the bound on r is high enough to guess as many variables as we need to solve the equation system with FXL. So we claim that FXL is always better than XL' for practical purpose.

XSL
Courtois and Pieprzyk [14] published this method at Asiacrypt 2002 and claimed to have broken AES using it. This was disproved in 2005 by Leurent and Cid [9]. The idea of XSL is to use the special structure of the equation system. If some equations are sparse you might introduce more new monomials than equations by multiplying them by all monomials of a special degree. So in some case it might be better to multiply some equations only by some monomials. It still is an open question how to do this. The idea of XSL is connected to Coppersmiths lattice based method to solve modular equations. Like in XL you multiply the equation by so called shift polynomials. Choosing the correct shift set is a difficult problem. In the case of two unknowns, we can plot the Newton polytope and get an intuition. But in multivariate cryptography you deal with a lot more unknowns. So it even is an open problem to find the correct shift set for some given equation.

MutantXL
One of the most efficient variants of XL is called MutantXL [16, 30] respectively XL2 [35]. It is claimed to be as fast as F_4 in some cases. This claim was derived from experiments on HFE [8].

Let I be the number of linearly independent equations produced by $\mathtt{XL}_D^{\mathtt{inh}}$ and $T = \binom{n+D+2}{D+2}$ the number of monomials of degree $\leq D+2$. If $(T-I) > (D+2)$ it is highly unlikely that XL finds a univariate polynomial and thus solves the system. As outlined above, XL will continue with $D := D+1$. MutantXL is a step in between. Instead of doing a full extension from D to $D+1$ it uses equations that would be produced by $\mathtt{XL}_{D+k}^{\mathtt{inh}}$ with $k > 0$ as long as they do not introduce new monomials. To this aim we use polynomials of degree $< D+2$ that are produced in the Gaussian elimination step of $\mathtt{XL}_D^{\mathtt{inh}}$. These polynomials are called *mutants*. For example multiplying these polynomials by all monomials of $\mathtt{Mon}_1$ leads to new equations without generating new monomials. However, this strategy is only useful for *inhomogeneous* equations. In the *homogeneous* case all monomials have same degree and thus mutants simply never occur.

Definition 3. *Let* $f = \sum_{i=1}^{m} g_{j_i} h^{(i)}$ *with* $h^{(i)} \in P^{\mathtt{inh}}$ *and* g_{j_i} *some polynomial of degree* $\leq D$ *be a representation of* f*. This representation is not unique. The set* J *denotes all representations* $(j_1, \ldots, j_m)$ *of* f*. The* level (lev) *of this representation* $j \in J$ *is defined by*

$$lev\left(\sum_{i=1}^{m} g_{j_i} h^{(i)}\right) := max\left\{ deg\left(g_{j_i} h^{(i)}\right) \mid 1 \leq i \leq m \right\}.$$

The level (Lev) *of f is defined by the minimum level of all its representations.*

$$Lev\,(f) := min\{lev\left(\sum_{i=1}^{m} g_{ji}h^{(i)}\right) \mid j \in J\}$$

We call g a mutant if $deg(f) < Lev(f)$.

We will give a detailed complexity analysis of MutantXL in section 6.

WXL, PWXL and WMXL
In his PhD thesis [31] Mohamed used Wiedemann's algorithm instead of Gaussian Elimination to decrease the amount of memory needed for XL and called this algorithm WXL. He showed experimentally that his variant always consumes less memory than F_4. As Wiedemanns algorithm allows easy parallelization he also introduced a parallel version called PWXL. The claim that this version is faster than F_4 is not completely fair as they used several processors in parallel but compared to F_4 running on only one processor. In cryptanalysis it is very common to guess some variables before solving a system of equations. Due to this guessing you can also easily run F_4 in parallel. Combining Wiedemanns algorithm with MutantXL is called WMXL.

4 The generic case of random systems

To analyze the complexity of XL in section 5 we need to count the dimension of the vector space spanned by $\{ap^{(k)} \mid 1 \leq k \leq m \text{ and } a \in \mathtt{Mon}_D\}$ or to put it simpler, the number of linearly independent equations I generated by $\mathtt{Blow}_D^{\mathtt{hom}}$. Obviously if two rows of Π are linearly dependent, all their multiples are, too. Moreover, even if two polynomials share a common factor, we get a nontrivial dependency in $\mathtt{XL}_1^{\mathtt{inh}}$ (see section 5 for details). Thus it seems infeasible to derive *one* formula covering all $\mathcal{MQ}$-systems. Instead we concentrate our analysis on the *generic* case of random $\mathcal{MQ}$-systems. The question to deal with in this section is: Which properties does a random $\mathcal{MQ}$-system typically have? A first attempt is due to Macaulay [28], who defined regular sequences as early as 1916.

Definition 4 (regular sequence). *A sequence of m polynomials $(p_1, \ldots, p_m)$ is regular if for all $i = 1, \ldots, m$, p_i is not a zero-divisor in the quotient ring $\mathbb{F}[x_1, \ldots, x_n]/(p_1, \ldots, p_{i-1})$. In other words if there exists g such that $gp_i \in \langle p_1, \ldots, p_{i-1}\rangle$ then $g \in \langle p_1, \ldots, p_{i-1}\rangle$ also holds.*

According to this definition regular sequences can be viewed as sequences without any special internal structure, *i.e.* the only relations holding are the trivial ones. More precisely $gp_i \in \langle p_1, \ldots, p_{i-1}\rangle$ means that there is a linear combination of multiples of $p_1, \ldots, p_{i-1}$ that equals gp_i and thus gp_i is linearly dependent to the equations produced by $p_1, \ldots, p_{i-1}$. For regular sequences this implies $g \in \langle p_1, \ldots, p_{i-1}\rangle$ which means that $p_i g \in \langle p_1, \ldots, p_{i-1}\rangle$ is trivially true and thus there only exist the trivial dependency $gp_i = p_i g$.

Definition 4, *i.e.* $gp_i \in \langle p_1, \ldots, p_{i-1} \rangle \Leftrightarrow g \in \langle p_1, \ldots, p_{i-1} \rangle$ can also be written as

$$gp_i = \sum_{j=1}^{i-1} h_j p_j \Leftrightarrow g = \sum_{j=1}^{i-1} l_j p_j \Leftrightarrow gp_i = \sum_{j=1}^{i-1} l_j p_j p_i, \tag{8}$$

for some polynomials h_j and l_j.

If we denote the *linear closure* of degree k of a polynomial f or a set P, respectively, as

$$\begin{aligned} \mathrm{Lin}(f,k) &:= \mathrm{span}\left(\{\mu f : \mu \in \mathtt{Mon}_k\}\right) \\ \mathrm{Lin}(P,k) &:= \mathrm{span}\left(\{\mu p : \mu \in \mathtt{Mon}_k, p \in P\}\right) , \end{aligned}$$

and define with $\#\mathrm{Lin}(P,k)$ the dimension of $\mathrm{Lin}(P,k)$ or to put it simpler the number of linearly independent equations, then condition (8) can, due to Moh [29, section 4], be equivalently formulated as follows.

$$\#(Lin(\{p_1, \ldots, p_{i-1}\}, k) \cap Lin(p_i, k)) = \#Lin(\{p_1, \ldots, p_{i-1}\}, k-2) \tag{9}$$

We will need this equation to proof lemma 1 later on.

Bardet *et al.* mentioned in their complexity analysis of Gröbner basis computations [4, 5] that regular systems only exists if the number of equations m is less or equal the number of variables n. Thus they introduced the definition of semi-regular systems to cover the overdetermined case $m > n$. Therefore they first needed the notion of the degree of regularity, which is the smallest degree d such that the dimension of the vector space spanned by all polynomials of an ideal $\mathfrak{I} = \langle p_1, \ldots, p_m \rangle$ with degree d equals the number of monomials of degree d. Or to put it simpler, the number of linearly independent equations I equals the number of monomials T.

Definition 5 (degree of regularity). *The degree of regularity of a homogeneous ideal $\mathfrak{I} = \langle p_1, \ldots, p_m \rangle$ is defined by*

$$d_{reg} := min\left\{ d \geq 0 \,:\, dim\left(\{p \in \mathfrak{I} \mid deg(p) = d\}\right) = \binom{n+d-1}{d} \right\}.$$

Definition 6 (semi-regular sequence). *A homogeneous sequence of m polynomials $(p_1, \ldots, p_m)$ is semi-regular if for all $i = 1, \ldots, m$ and g such that $gp_i \in \langle p_1, \ldots, p_n \rangle$ and $deg(gp_i) < d_{reg}$ then $g \in \langle p_1, \ldots, p_{i-1} \rangle$ also holds.*

Unfortunately it is not proven yet that semi-regular sequences are generic and thus all the proofs are built on this assumption. Diem [15] reduced this assumption to the more common MinRank conjecture or also known as Fröbergs conjecture.

To give some intuition on the behavior of random $\mathcal{MQ}$-systems, we will now calculate the probability for such a system with $m = n = 2$ to be regular. In that special case definition 4 is equivalent to the condition of both polynomials being co-prime. Let $f = h_1h_2$ and $g = h_1h_3$ be two quadratic polynomials sharing a common factor h_1, then obviously $h_3f = h_2g$ is a nontrivial dependency. Conversely if $h_1g = h_2f$ and $h_1 \notin \langle f \rangle$ then obviously $\gcd(f, g) \neq 1$. The following corollary gives the probability of two random quadratic polynomials f, g being not co-prime.

Corollary 1. *Two randomly chosen $\mathcal{MQ}$-polynomials $f, g \in \mathbb{F}_q[x_1, x_2]$ are not co-prime with probability*

$$\frac{(q-1)^2 + (q^3-q)^3}{(q^6-q^3)^2} \approx \mathcal{O}(q^{-3}).$$

Proof. Two randomly chosen quadratic polynomials f and g are not co-prime iff they share a common factor. There are $(q-1)^2$ possibilities for $\lambda_1, \lambda_2 \in \mathbb{F}_q^*$ with $\lambda_1 g = \lambda_2 f$. Let $g = ab$ and $f = ac$ with $a, b, c \in \mathbb{F}[x_1, x_2]$ and $\deg(a) = \deg(b) = \deg(c) = 1$. There exist $(q^3 - q)$ possibilities each for a, b, c. The total amount of quadratic polynomials in $\mathbb{F}_q[x_1, x_2]$ is $(q^6 - q^3)$ and thus we have $(q^6 - q^3)^2$ possibilities to choose f and g. □

For the common setting $q = 2^8$, the probability of a system with $m = n = 2$ to be not regular is 2^{-24}. Note that this probability quickly decrease if n increase.

5 Complexity Analysis of XL revisited

The crucial point when using XL is to determine the number of linearly independent equations I produced by $\mathtt{Blow}_D^{\mathtt{hom}}$ or $\mathtt{XL}_D^{\mathtt{inh}}$. This is needed to calculate D through $T - I < D + 2$ and therefore implies the complexity of the whole algorithm. For random equation systems (see section 4) we will now revisit the formulas derived theoretically by Moh [29], Yang and Chen [35] or by experiments for D between 0 and 5 over $\mathbb{F}_2$ by Courtois and Patarin [13].

We run own experiments to confirm previous results. All equations in table 2, 3 and 4 were obtained by a total of several 10,000 experiments. We omitted discordant values which occurred with very low probability every time the random system did not match the conditions of section 4. All experiments were performed on a Intel Xeon X33502.66GHz (Quadcore) with 8 GB of RAM using only one core and the software system Magma V2.16-1 [10]. Parameters were running for various tuples (n, m, D) in the range $3 \leq n \leq 15$, $3 \leq m \leq 50$, $1 \leq D \leq 8$. Notice, for a ground field $\mathbb{F}_q$ and $D + 2 < q$, the formulas are independent of the ground field. If $D + 2 \geq q$ we have to take the field equations $x^q = x$ into account and things get messy—at least from a theoretical perspective. For example if $q = 2$ the number of monomials of degree D decreases from $\binom{n+D-1}{D}$ to $\binom{n}{D}$ and besides of the trivial dependency $fg = gf$ there is an additional

dependency due to $f^2 = f$ for f, g quadratic polynomials. The important case of $\mathbb{F}_2$ was treated by Rønjom and Raddum in [34]. See table 5 for their results restricted to the homogeneous case.

Table 2. Number of linearly independent equations produced by $\texttt{Blow}_D^{\texttt{hom}}$, experimentally derived.

D	Number of linearly independent equations
0	m
1	mn
2	$m\binom{n+1}{2} - \binom{m}{2}$
3	$m\binom{n+2}{3} - \binom{m}{2}n$
4	$m\binom{n+3}{4} - \binom{m}{2}\binom{n+1}{2} + \binom{m}{3}$
5	$m\binom{n+4}{5} - \binom{m}{2}\binom{n+2}{3} + \binom{m}{3}n$

Table 3. Number of linearly independent equations produced by $\texttt{Blow}_D^{\texttt{inh}}$, experimentally derived.

D	Number of linearly independent equations
0	m
1	mn
2	$m\binom{n+1}{2}$
3	$m\binom{n+2}{3} - \binom{m-1}{3}$
4	$m\binom{n+3}{4} - \binom{m-1}{3}n + \binom{m-1}{4}$
5	$m\binom{n+4}{5} - \binom{m-1}{3}\binom{n+1}{2} + \binom{m-1}{4}n - \binom{m-1}{5} + \binom{m-1}{4}$
6	$m\binom{n+5}{6} - \binom{m-1}{3}\binom{n+2}{3} + \binom{m-1}{4}\binom{n+1}{2} - \binom{m-1}{5}n + \binom{m-1}{4}n + \binom{m-1}{6} - \binom{m-1}{5}$

Table 4. Number of linearly independent equations produced by $\texttt{XL}_D^{\texttt{inh}}$, experimentally derived.

D	Number of linearly independent equations
0	m
1	$m + mn$
2	$m + mn + m\binom{n+1}{2} - \binom{m}{2}$
3	$m\binom{n+3}{3} - \binom{m}{2}(n+1)$
4	$m\binom{n+4}{4} - \binom{m}{2}\binom{n+2}{2} + \binom{m}{3}$
5	$m\binom{n+5}{5} - \binom{m}{2}\binom{n+3}{3} + \binom{m}{3}(n+1)$

Table 5. Number of linearly independent equations produced by $\mathtt{Blow}_D^{\mathtt{hom}}$ over $\mathbb{F}_2$.

D	Number of linearly independent equations
0	m
1	mn
2	$m\binom{n}{2} - (\binom{m}{2} + m)$
3	$m\binom{n}{3} - (\binom{m}{2} + m)n$
4	$m\binom{n}{4} - (\binom{m}{2} + m)\binom{n}{2} + (\binom{m}{3} + 2\binom{m}{2} + m)$
5	$m\binom{n}{5} - (\binom{m}{2} + m)\binom{n}{3} + (\binom{m}{3} + 2\binom{m}{2} + m)n$

We restrict our analysis to $D + 2 < q$ in the whole paper. Replacing formulas will easily cover the other cases.

In section 3.2 we claimed that $\mathtt{Blow}_D^{\mathtt{hom}}(n+1)$ and $\mathtt{XL}_D^{\mathtt{inh}}(n)$ are equivalent due to homogenization. With the formulas of table 2 and 4 we proof this claim exemplarily for $D = 2$:
Note that for $\mathtt{Blow}_D^{\mathtt{hom}}(n+1)$, the number of monomials is $\binom{n+4}{4} - 1$ because we know x_{n+1}^4 by the choice of $x_{n+1} = 1$ for homogenization. Considering the formula $I - T$ we get the following.

$$
\begin{aligned}
&\mathtt{Blow}_2^{\mathtt{hom}}(n+1): \\
&m\binom{n+2}{2} - \binom{m}{2} - \binom{n+4}{4} + 1 \\
&= m\binom{n+1}{2} + m\binom{n+1}{1} - \binom{m}{2} - \binom{n+3}{4} - \binom{n+3}{3} + 1 \\
&= m + mn + m\binom{n+1}{2} - \binom{m}{2} - \binom{n+3}{4} - \binom{n+2}{3} - \binom{n+2}{2} + 1 \\
&= m + mn + m\binom{n+1}{2} - \binom{m}{2} - \binom{n+3}{4} - \binom{n+2}{3} - \binom{n+1}{2} - n \\
&: \mathtt{XL}_2^{\mathtt{inh}}(n)
\end{aligned}
$$

Now we proof the formulas given in table 2 theoretically. The following result was given and proven inductively by Moh [29]. We want to formulate this proof in more detail and show were the systematic linear dependencies arise.

Lemma 1. *If P^{hom} is a semi-regular sequence, then the number of linearly independent equations produced by $\mathtt{Blow}_D^{\mathtt{hom}}$ with $D = 2k+b$ and $b \in \{0, 1\}$ is given by*

$$
I_{\mathtt{Blow}_D^{\mathtt{hom}},n} := \sum_{i=0}^{k} (-1)^i \binom{m}{i+1} \binom{n + 2(k-i) - 1 + b}{2(k-i) + b}. \tag{10}
$$

Before proving this lemma at the end of this section, we need some intermediate results. First we concentrate on $\mathtt{Blow}_2^{\mathtt{hom}}$ and search for the $\binom{m}{2}$ linear dependent

equations out of all $m\binom{n+1}{2}$ produced equations. Let f, g be two $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic polynomials in n variables each. Denote $\mathtt{Mon}_f, \mathtt{Mon}_g$ the set of monomials in f and g, respectively. Assume the existence of some admissible ordering for multivariate polynomials f, g, *e.g. degrev-lex* or *lex*.

Lemma 2. *Let f, g be a pair of co-prime $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic polynomials. Moreover, let $F := \{bf : b \in \mathtt{Mon}_g\}$ and $G := \{ag : a \in \mathtt{Mon}_f\}$ be the sets of cross-wise monomial multiplication of f and g, respectively. Then these two sets produce $|F| + |G| - 1$ linearly independent equations.*

Proof. We denote our two polynomials by $f := \sum_{i=1}^{\sigma} \alpha_i a_i$ and $g := \sum_{i=1}^{\tau} \beta_i b_i$ for non-zero field elements $\alpha_i, \beta_j \in \mathbb{F}^*$ and monomials a_i, b_j for $1 \leq i \leq \sigma$ and $1 \leq j \leq \tau$. All monomials have degree 2, *i.e.* we have $\deg(a_i), \deg(b_i) = 2$. The important property of the two sets F, G is that each monomial ab for $a \in \mathtt{Mon}_f$ and $b \in \mathtt{Mon}_g$ exists twice, namely once in $bf \in F$ and once in $ag \in G$. The following equation shows that adding all equations of F multiplied by coefficients β_i is equal to adding all equations of G multiplied by coefficients α_i and thus the set $F \cup G$ is linear dependent.

$$\sum_{i=1}^{\tau} \beta_i b_i f = \sum_{i=1}^{\tau} \beta_i b_i \sum_{j=1}^{\sigma} \alpha_j a_j = \sum_{j=1}^{\sigma} \alpha_j a_j \sum_{i=1}^{\tau} \beta_i b_i = \sum_{j=1}^{\sigma} \alpha_j a_j g$$

For short we write $fg = gf$ and call this relation *trivial syzygy*. On the other hand assume the existence of a nontrivial syzygy $h_1 f = h_2 g$. As g and f are co-prime this directly implies $f \,|\, h_2$ and $g \,|\, h_1$ which contradicts that $h_1 f = h_2 g$ is a nontrivial syzygy . □

Corollary 2. *The largest linearly independent subset of $\mathtt{Blow}_2^{\mathtt{hom}}$ for regular or semi-regular sequences is of size $\binom{n+1}{2}m - \binom{m}{2}$.*

Proof. By its definition, we have at most $\binom{n+1}{2}m$ distinct elements in $\mathtt{Blow}_2^{\mathtt{hom}}$. This explains the first part of the sum and also gives an upper bound. Considering all pairs $(f, g) \in P \times P$ with $f < g$ and also Lemma 2, we obtain $\binom{m}{2}$ linear dependencies. □

Corollary 3. *The largest linearly independent subset of $\mathtt{XL}_2^{\mathtt{inh}}$ is of size $\binom{n}{2}m + nm + m - \binom{m}{2}$.*

Proof. This corollary works similar to corollary 2. By its definition, we have at most $\binom{n}{2}m + nm + m$ elements in $\mathtt{XL}_2^{\mathtt{inh}}$. This explains the first part of the sum and also gives an upper bound. Considering all pairs $(f, g) \in P \times P$ with $f < g$ and also Lemma 2, we obtain $\binom{m}{2}$ linear dependencies. □

Lemma 3. *Let f, g be a pair of linearly independent, homogeneous $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic polynomials. For $n \geq k > 2$, the set $\mathtt{Blow}_k^{\mathtt{hom}} = \{\mu f, \mu g : \mu \in \mathtt{Mon}_k\}$ contains at most $2\binom{n+k-1}{k} - \binom{n+k-3}{k-2}$ linearly independent equations.*

Proof. The first part of the sum is a result of the $\binom{n+k-1}{k}$ choices of the monomial μ. We fix some monomial $v \in \mathtt{Mon}_{k-2}$ and study the two sets $F_v := \{vfb : b \in \mathtt{Mon}_g\}$ and $G_v := \{vga : a \in \mathtt{Mon}_f\}$. For a given pair F_v, G_v, we can now apply lemma 2. We have $|\mathtt{Mon}_{k-2}| = \binom{n+k-3}{k-2}$ individual choices for v. □

Extending this lemma from pairs to sets is kind of tricky, because since $D \geq 4$ we obtain new linear dependencies between 3 and more equations. Thus we are counting linear dependencies twice if we only consider pairs f, g. To count all equations only once, we need (9) cf. section 4.

Proof (lemma 1). First we reformulate the formula of lemma 1. The number of linearly independent equations $\#\mathrm{Lin}(P^{\mathtt{hom}}, D)$ there is given by

$$\sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i+1} \binom{n+D-2i-1}{n-1}. \tag{11}$$

We proof this by induction via m. The case $m = 1$ is trivial.
Let us assume (11) holds for m. We have to show that it also holds for $m+1$.

We have $P^{\mathtt{hom}}_{m+1} := P^{\mathtt{hom}}_m \cup \{p_{m+1}\}$ and write

$$\begin{aligned}\#\mathrm{Lin}(P^{\mathtt{hom}}_{m+1}, D) = {} & \#\mathrm{Lin}(P^{\mathtt{hom}}_m, D) + \#\mathrm{Lin}(p_{m+1}, D) \\ & - \#(\mathrm{Lin}(P^{\mathtt{hom}}_m, D) \cap \mathrm{Lin}(p_{m+1}, D)).\end{aligned}$$

The last term simplifies to $\#\mathrm{Lin}(P^{\mathtt{hom}}_m, D-2)$ using (9). Using the induction hypothesis we obtain the following formula for $\#\mathrm{Lin}(P^{\mathtt{hom}}_{m+1}, D)$.

$$\begin{aligned}& \sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i+1} \binom{n+D-2i-1}{n-1} \\ & + \binom{n+D-1}{D} \\ & - \sum_{0 \leq 2i \leq D-2} (-1)^i \binom{m}{i+1} \binom{n+D-2i-3}{n-1} \\ & = \sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i+1} \binom{n+D-2i-1}{n-1} \\ & + \sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i} \binom{n+D-2i-1}{n-1} \end{aligned} \tag{12}$$

Exploiting $\binom{m}{l} = \binom{m-1}{l} + \binom{m-1}{l-1}$ yields

$$(12) = \sum_{0 \leq 2i \leq D} (-1)^i \binom{m+1}{i+1} \binom{n+D-2i-1}{n-1}.$$

□

The overall complexity of the XL algorithm is essentially the workload of the Gaussian Elimination step and thus given by $\binom{n+D+2}{D+2}^{\omega}$ for $2 \leq \omega \leq 3$. For plain Gaussian Elimination we have $\omega = 3$. Using sparse matrix algebra, we can assume $\omega = 2$ [32]. Table 6 give the saturation degree $D+2$ for some practical important choices of m equations and $n = m - r$ variables, *i.e.* r variables are guessed. The corresponding $\log_2$ complexity, including guessing over $\mathbb{F}_{2^8}$, can be found in table 7.

$m\backslash r$	0	1	2	3	5
5	-	5	3	2	2
10	-	10	6	5	3
15	-	15	8	7	5
20	-	20	11	9	7
25	-	25	13	11	9
30	-	30	16	14	11

Table 6. Degree $D+2$ of XL.

$m\backslash r$	0	1	2	3	5
5	-	22	25	29	40
10	-	41	39	43	52
15	-	60	51	55	63
20	-	80	66	67	75
25	-	100	78	79	87
30	-	119	93	94	98

Table 7. Complexity of XL over $\mathbb{F}_{2^8}$

6 Complexity Analysis of MutantXL

Again, the crucial question is how many equations produced by mutants are linearly independent to the previous ones. We give a nontrivial upper bound on this number. Assuming this bound to be tight we are able to calculate the saturation degree of MutantXL, which matches empirical data. To conclude we compare MutantXL to Gröbner basis algorithms like F_4 and show that it solves at the degree of regularity more often than not, but never below.

First we observe that not all mutants are useful, as some of them trivially produce linearly dependent elements.

Definition 7 (trivial mutant). *Let $D+2$ be the saturation degree of MutantXL. We call mutants (cf. Def. 3) in the linear hull of $\mathtt{XL}^{\mathtt{inh}}_{D-1}$ trivial.*

The definition of trivial mutants is motivated by the following observation. Let g be a *trivial mutant*, *i.e.* $g \in \mathrm{span}(\mathtt{Blow}^{\mathtt{inh}}_d)$, $\deg(g) < (d+2)$ and $d < D$. For every $x \in \mathtt{Mon}_{D-d}$ we obtain by xg a mutant of $\mathtt{Blow}^{\mathtt{inh}}_D$. Thus all the linearly independent equation produced by trivial mutants are produced twice by *non-trivial mutants* of $\mathtt{Blow}^{\mathtt{inh}}_D$.

Let $\mathcal{D}_{\mathtt{Blow}^{\mathtt{inh}}_D,n} := \dim(\mathrm{span}(\mathtt{XL}^{\mathtt{inh}}_D)) - \dim(\mathrm{span}(\mathtt{XL}^{\mathtt{inh}}_{D-1}))$ denote the difference between the dimensions of the vector spaces generated by $\mathtt{XL}^{\mathtt{inh}}_D$ and $\mathtt{XL}^{\mathtt{inh}}_{D-1}$ or to put it simpler the number of new linearly independent equations produced by $\mathtt{Blow}^{\mathtt{inh}}_D$. We calculate $\mathcal{D}_{\mathtt{Blow}^{\mathtt{inh}}_D,n}$ using (11) by

$$\mathcal{D}_{\mathtt{Blow}^{\mathtt{inh}}_D,n} = I_{\mathtt{XL}^{\mathtt{inh}}_D,n} - I_{\mathtt{XL}^{\mathtt{inh}}_{D-1},n} = I_{\mathtt{Blow}^{\mathtt{hom}}_D,n+1} - I_{\mathtt{Blow}^{\mathtt{hom}}_{D-1},n+1}\,. \tag{13}$$

Recall the strategy of MutantXL (cf. section 3.4). In step 0 we produce $\mathtt{XL}_D^{\mathtt{inh}}$ similar to the XL algorithm. In step 1 we determine all nontrivial mutants through $\mathtt{Blow}_D^{\mathtt{inh}}$ and multiply them by all monomials with degree at most k, with $k \in \mathbb{N}$ fullfilling the following condition.

$$\sum_{j=0}^{k-1} |\mathtt{Mon}_{D+2-j}| \leq \mathcal{D}_{\mathtt{Blow}_D^{\mathtt{inh}},n} \leq \sum_{j=0}^{k} |\mathtt{Mon}_{D+2-j}| \,. \tag{14}$$

For the sake of simplicity we use $k = 1$, *i.e.* $|\mathtt{Mon}_{D+2}| \leq \mathcal{D}_{\mathtt{Blow}_D^{\mathtt{inh}},n} \leq |\mathtt{Mon}_{D+2}| + |\mathtt{Mon}_{D+1}|$ illustrated in figure 3, in our explanation. Note that neither we found a set of parameters m, n with $k > 1$ in step 1 nor could formally proof this fact. But as we only distinguish the cases $k = 0$ and $k > 0$ in our final analysis that question is of no effect. For $k = 1$ mutants will produce at most $n(\mathcal{D}_{\mathtt{Blow}_D^{\mathtt{inh}},n} -$

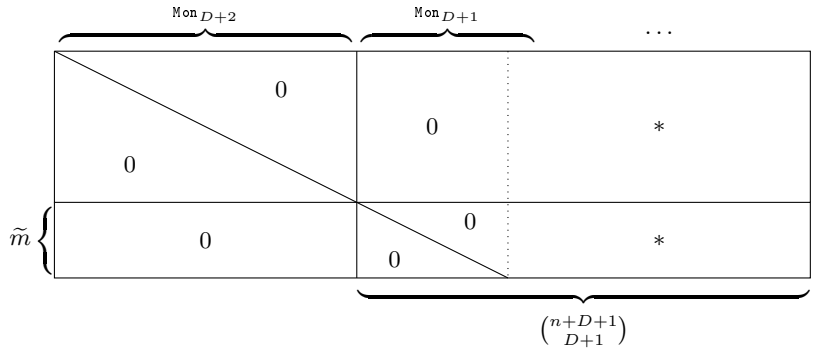

Fig. 3. Coefficient Matrix Π of $\mathtt{Blow}_D^{\mathtt{inh}}$ after Gaussian elimination. Here $\tilde{m}$ indicates the number of mutants for the corresponding system P.

$|\mathtt{Mon}_{D+2}|$) equations or $n\widetilde{m}$ using the notation of figure 3, as we multiply all $\widetilde{m}$ mutants by all n monomials of degree one. See corollary 4 for arbitrary k. Experiments for $2 \leq n \leq 7$ and $n \leq m \leq 9$ over $\mathbb{F}_{2^8}$ show that this trivial bound is far above the correct number of new linearly independent equations.

Corollary 4. *The maximal number of equations produced by nontrivial mutants is given by*

$$\sum_{i=1}^{k-1} \binom{n+i-1}{i} |\mathtt{Mon}_{D+2-i}| + \binom{n+k-1}{k} \left(\mathcal{D}_{\mathtt{Blow}_D^{\mathtt{inh}},n} - \sum_{i=0}^{k-1} |\mathtt{Mon}_{D+2-i}| \right).$$

Our nontrivial upper bound uses the fact that all $n\widetilde{m}$ equations produced by mutants are implicit equations of $\mathtt{Blow}_{D+1}^{\mathtt{inh}}$. Exactly $\mathcal{D}_{\mathtt{Blow}_{D+1}^{\mathtt{inh}},n}$ (see eqn. 13) of them are linearly independent to $\mathtt{XL}_D^{\mathtt{inh}}$. But they all contain monomials of

$\mathtt{Mon}_{D+3}$. Equations produced by mutants have maximal degree $D+2$ and thus first all $|\mathtt{Mon}_{D+3}|$ monomials have to be reduced. Therefore $\mathcal{D}_{\mathtt{Blow}^{\mathtt{inh}}_{D+1},n} - |\mathtt{Mon}_{D+3}|$ is an upper bound on the number of linearly independent equations produced by mutants. For arbitrary k this leads to the following upper bound.

Corollary 5. *A nontrivial upper bound on the number of linearly independent equations produced by mutants is given by*

$$\begin{aligned} &\sum_{i=1}^{k} \left(I_{\mathit{IL}^{\mathtt{inh}}_{D+i},n} - I_{\mathit{IL}^{\mathtt{inh}}_{D+i-1},n} - |\mathit{Mon}_{D+2+i}| \right) \\ &= I_{\mathit{IL}^{\mathtt{inh}}_{D+k},n} - I_{\mathit{IL}^{\mathtt{inh}}_{D},n} - \sum_{j=1}^{k} |\mathit{Mon}_{D+2+j}| \,. \end{aligned} \tag{15}$$

In step 2 we would reduce the equations obtained in step 1 and thus receive new mutants which we would multiply by all monomials of a certain degree to obtain new equations and so on and so forth. We can iterate this process and thus get an upper bound on the number of new linearly independent equations produced by mutants, if we consider k such that equation 15 is maximal. We do not even have do go that far, as obviously $T-I$ is a trivial upper bound and we can show that for $k \to \infty$ equation 15 matches these trivial bound. Let therefore transform equation 15 as follows.

$$\begin{aligned} &I_{\mathtt{XL}^{\mathtt{inh}}_{D+k},n} - I_{\mathtt{XL}^{\mathtt{inh}}_{D},n} - \sum_{j=1}^{k} |\mathtt{Mon}_{D+2+j}| \\ &= \binom{n+D+2}{D+2} - I_{\mathtt{XL}^{\mathtt{inh}}_{D},n} + I_{\mathtt{XL}^{\mathtt{inh}}_{D+k},n} - \binom{n+D+k+2}{D+k+2} \\ &= \binom{n+D+2}{D+2} - I_{\mathtt{XL}^{\mathtt{inh}}_{D},n} - \underbrace{\sum_{i=0}^{D+k} (-1)^i \binom{m-n-1}{i} \binom{m}{D+k+2-i}}_{=:\Omega} \end{aligned}$$

Note, the second equality is due to Yang and Chen [35]. Obviously Ω is zero if $D+k+2-m > i > m-n-1$ and thus for $k > 2m-n-3-D$ the number of new linearly independent equations produced by mutants is upper bounded by $T-I$, what is exactly the range left until we obtain maximal rank. Note that depending on the solution of the $\mathcal{MQ}$-system $T=I$ might be impossible. But as $T-I < D+2$ suffices to find a solution using MutantXL the upper bound might not be matched but sufficiently tight to obtain a solution. Even if we are not able to proof that mutants produce sufficiently many equations to solve the $\mathcal{MQ}$-system, there are three good arguments. First all the equations are from different spaces $\mathtt{Blow}^{\mathtt{inh}}_{D+i}$ for $i > 0$. Second, if we multiply equations with monomials of degree one that are implicitly from $\mathtt{Blow}^{\mathtt{inh}}_{D+k}$ and $D+k$ is even, than all multiples should be linearly independent as new trivial syzygies are only introduced from odd to even degree (cf. sec. 5). And last but not least experimental evidence. We

did experiments for $2 \leq n \leq 7$ and $n+1 \leq m \leq (n+1)n/2$ which all confirmed the saturation degree for MutantXL given in the following lemma.

Lemma 4. *If we assume $T-I$ to be a sufficiently tight upper bound on the number of new linearly independent equations produced by mutants, then MutantXL solves at saturation degree $D+2$ as soon as nontrivial mutants occur,* i.e.

$$\mathcal{D}_{Blow_D^{\mathtt{inh}},n} > \binom{n+D+1}{D+2}.$$

We want to mention that a similar result was given by Yang and Chen [35, Prop.4] looking at the problem from a different perspective. They also state that MutantXL (respectively XL2) solves as soon as Mutants occur. This happens as soon as all top-degree monomials are eliminated. If we separate the top-degree monomials of every equation, we can think of an homogeneous system with n variables or equivalently an inhomogeneous system of $n-1$ variables. Thus Yang and Chen concluded that Mutants first occur at the saturation degree of $\mathtt{XL}_D^{\mathtt{inh}}(n-1)$. Due to the following equality this is equal to lemma 4.

$$\mathcal{D}_{\mathtt{Blow}_D^{\mathtt{inh}},n} = I_{\mathtt{XL}_D^{\mathtt{inh}},n} - I_{\mathtt{XL}_{D-1}^{\mathtt{inh}},n} = I_{\mathtt{XL}_D^{\mathtt{inh}},n-1}$$

By having a closer look at the problem, especially by corollary 5, we gave additional intuition and hopefully did a step to proof lemma 4 without assumptions. In figures 4 and 5 we calculated the saturation degree $D+2$ of MutantXL with the degree of regularity d_{reg} [5] of Gröbner basis algorithms like F_4 or F_5 for random $\mathcal{MQ}$-systems with $n \in [1, 10]$ respectively $n \in [1, 30]$ variables and $m \in \left[n+1, \frac{n(n+1)}{2}\right]$ equations. Note, the case $m = n$ is an exception, as $D+2 = 2^m$ both for XL and MutantXL.

In a nutshell, MutantXL almost always solves at the degree of regularity. Only in very few cases, except $m = n$, it solves at most one degree higher than $\mathrm{F}_4/\mathrm{F}_5$. Table 8 and 9 give numerical values. Table 10 show the corresponding difference of the saturation degree of MutantXL and d_{reg}. To mention the gain of MutantXL over XL, table 11 shows the difference between the saturation degrees of both.

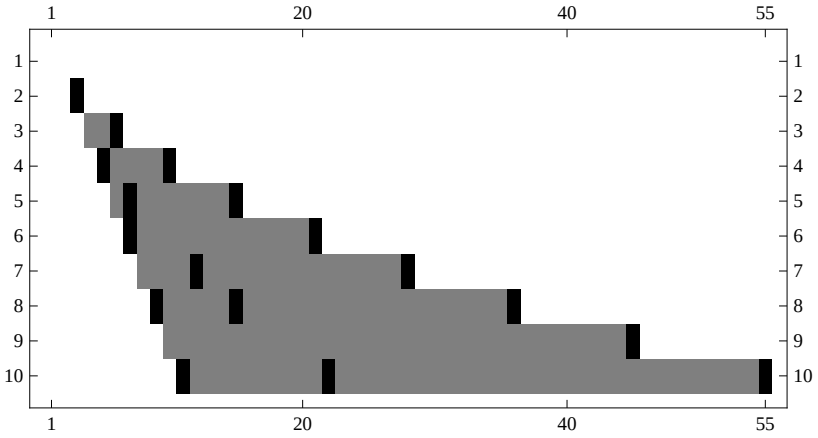

Fig. 4. Visualization $\Delta := D + 2 - d_{reg}$ for $n = 1 \ldots 10$. Black: $\Delta = 1$, gray: $\Delta = 0$. Y-Axis is number of variables n, x-axis the number of equations m.

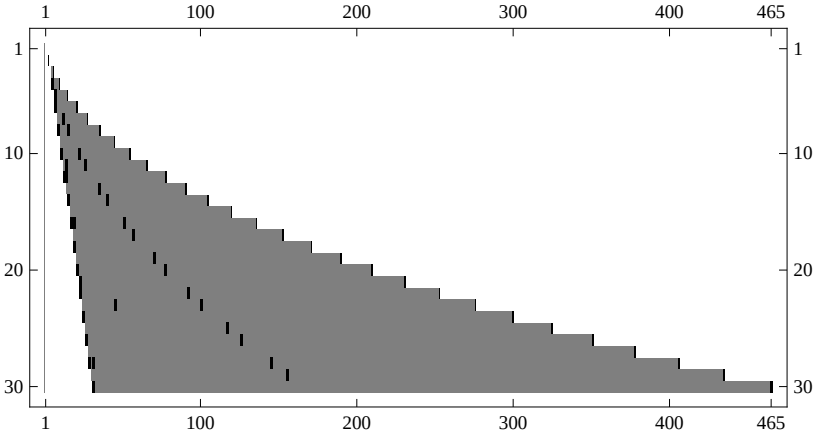

Fig. 5. Visualization $\Delta := D + 2 - d_{reg}$ for $n = 1 \ldots 30$. Black: $\Delta = 1$, gray: $\Delta = 0$. Y-Axis is number of variables n, x-axis the number of equations m.

The complexity of MutantXL is determined by the Gaussian elimination step on all $m\binom{n+D}{D}$ equations produced by XL and the number of mutants—if any. The latter is captured by $\max\left\{0, \binom{n+D+2}{D+2} - I - D - 2\right\}$. Thus if $m = n$ and we guess r variables over $\mathbb{F}_q$ beforehand, the complexity of MutantXL given in table 13 is calculated by

$$\left(m\binom{n-r+D}{D} + \max\left\{0, \binom{n-r+D+2}{D+2} - I - D - 2\right\}\right)^{\omega} \cdot q^r.$$

As Bettale *et al.* we use $\omega = 2$ for the linear algebra constant, to make their hybrid approach algorithm comparable to our analysis of MutantXL. Table 12 gives the complexity of the HybridF$_5$ algorithm, which is computed through $\mathcal{O}\left(\left(m\binom{n+d_{reg}-1}{d_{reg}}\right)^2\right)$. Note for $m = 20$ and guessing one or two variables over $\mathbb{F}_{2^8}$ we obtain the same results as in [6, table 4]. The values in the tables are rounded Log_2 complexities. The exact value for $m = 20$, $r = 1$ and $\mathbb{F}_{2^8}$ is 66.73 respectively 67.79 for $r = 2$.

Comparing values of table 12 and 13 suggest that MutantXL is faster than F$_5$ in many cases. Due to [2] that should be impossible. We think the reason for that phenomena is the complexity given in [6], which should be a upper bound that is not tight. If we use $\mathcal{O}\left(\binom{n+d_{reg}}{d_{reg}}^2\right)$ instead (cf. [5]), F$_5$ indeed is faster than MutantXL as long as the degree of regularity is smaller or equal to the saturation degree (see table 14).

7 Comparison of Memory Consumption

Using XL, MutantXL or F$_{4/5}$ in practice brings new challenges. In particular, the memory consumption of these algorithms seriously hinders practical applications. Therefore, we outline the overall memory consumption of F$_4$ (upper and lower bound), of the rather memory friendly WiedemannXL, and MutantXL. In all cases, we assume that we need $\lceil \log_2 q \rceil$ bit to store one field element.
Note that all memory bounds can be improved by spending an extra workload of q^r for some small $r \in \mathbb{N}_{>0}$. In particular for $r = 1, 2$ this is usually feasible.

$m\backslash r$	0	1	2	3	5
5	6	3	3	2	2
10	11	6	5	4	3
15	16	8	7	6	4
20	21	11	9	8	6
25	26	13	11	10	8
30	31	16	14	12	10

Table 8. Degree of Regularity d_{reg}.

$m\backslash r$	0	1	2	3	5
5	32	4	3	2	2
10	-	6	5	4	3
15	-	9	7	6	4
20	-	11	9	8	6
25	-	14	11	10	8
30	-	16	14	12	10

Table 9. Saturation Degree ($D+2$) of MutantXL.

$m\backslash r$	0	1	2	3	5
5	26	1	0	0	0
10	-	0	0	0	0
15	-	1	0	0	0
20	-	0	0	0	0
25	-	1	0	0	0
30	-	0	0	0	0

Table 10. Difference of $D + 2$ of MutantXL and d_{reg}.

$m\backslash r$	0	1	2	3	5
5	0	1	0	0	0
10	-	4	1	1	0
15	-	6	1	1	1
20	-	9	2	1	1
25	-	11	2	1	1
30	-	14	2	2	1

Table 11. Difference of $D + 2$ of XL and MutantXL

$m\backslash r$	0	1	2	3	5
5	20	21	27	32	40
10	41	38	42	46	57
15	62	51	55	59	67
20	83	67	68	72	79
25	103	79	80	84	91
30	123	95	96	96	104

Table 12. Complexity of F_5 over $\mathbb{F}_{2^8}$ using [6].

$m\backslash r$	0	1	2	3	5
5	-	21	25	32	40
10	-	34	37	41	52
15	-	50	50	54	60
20	-	62	62	66	73
25	-	78	75	78	85
30	-	90	91	91	97

Table 13. Complexity of MutantXL over $\mathbb{F}_{2^8}$.

For some parameter sets, this can also improve the overall attack (*e.g.* FXL or HybridF_5).

Lemma 5 (Memory Bounds for $\mathrm{F}_{4/5}$). *The memory requirements* (in bits) *for $F_{4/5}$ are bounded from above by*

$$\mathtt{MemUpperF}_{4/5}(q, n, d_{reg}) = \lceil \log_2 q \rceil \binom{n + d_{reg} - 1}{d_{reg}}^2$$

The lower bound is given by

$$\mathtt{MemLowerF}_{4/5}(q, n, d_{reg}) = \lceil \log_2 q \rceil \binom{n + d_{reg} - 1}{d_{reg}} \frac{n(n+1)}{2}$$

Proof. $\mathtt{MemUpperF}_{4/5}$ is given by the number of monomials. As the number of monomials equals the number of rows, this is the size of the corresponding matrix. Without assuming sparsity, we need as many elements to store this matrix. In contrast, $\mathtt{MemLowerF}_{4/5}$ assumes the same number of rows/columns, but only the minimal number of coefficients, *i.e.* $n(n+1)/2$.

From a practical point of view, both bounds are too vague: Neither are the rows of the coefficient matrix Π in $\mathrm{F}_{4/5}$ dense, nor fully sparse. Unfortunately, we are not aware of a treatment of this question in the open literature.

$m\backslash r$	0	1	2	3	5
5	18	18	25	29	40
10	37	33	37	41	52
15	56	45	48	52	60
20	76	59	60	64	71
25	96	71	72	76	83
30	115	86	87	88	95

Table 14. Complexity of F_5 over $\mathbb{F}_{2^8}$ using [5].

Lemma 6 (Memory Bounds for XL). *XL needs at least a total of*

$$\mathtt{MemLowerXL}(q,n,D) = \lceil \log_2 q \rceil \binom{n+D+1}{D+2} \frac{n(n+1)}{2}$$

bits for saving the coefficient matrix, and at most

$$\mathtt{MemUpperXL}(q,n,D) = \lceil \log_2 q \rceil \binom{n+D+1}{D+2}^2$$

Proof. As for lemma 5, we consider the case of extremely sparse matrices (lower bound) and dense matrices.

Corollary 6. *WiedemannXL needs a total of*

$$\mathtt{MemWiedXL}(q,n,D) = \mathtt{MemLowerXL}(q,n,D)$$

memory for its coefficient matrix.

Motivation for this corollary: As we do not need to perform row or column operations on the coefficient matrix, we can preserve the sparsity. Consequently, we achieve the lower memory bound of XL.

Lemma 7 (WiedemannMutantXL). *Combining the ideas of Wiedemann and MutantXL, we achieve the following upper bound on the memory consumption:*

$$\mathtt{MemWiedMutantXL}(q,n,D) =$$
$$\mathtt{MemWiedXL}(q,n,D) + \frac{\log_2 q}{n} \binom{n+D+1}{D+1} \left(\binom{n+D+2}{D+2} - I_{\mathtt{XL}_D^{\mathrm{inh}}} \right)$$

Proof. To produce the mutants, we need to store the whole matrix in Wiedemann fashion. This explains the first part of the sum. Secondly, we obtain mutants. For these, we know that their first $\binom{n+D+1}{D+2}$ columns must be all-zero. Accordingly, we only need to save the following $\binom{n+D+1}{D+1}$ coefficients for each degree. We have to produce at most $T - I$ new equations by multiplying mutants by monomials. We need at most $(T-I)/n$ mutants to achieve this goal.

Note that we need to call Wiedemann's algorithm for our current coefficient matrix around $(T-I)/n$ times (assuming to optimizations like Block-Wiedemann). Hence, our implementation will be slowed down by this factor. On the other hand, we can usually solve with a much smaller value D and hence gain over plain XL.

All lemmata are summarized in Figure 6. We see that XL consumes far less memory than $\mathrm{F}_{4/5}$—regardless if we assume the higher or lower memory consumption. WiedemannXL is a step in between: Here, we have less memory than for $\mathrm{F}_{4/5}$ (upper case), but more memory for $\mathrm{F}_{4/5}$ (lower case). Reason: WiedemannXL needs a higher saturation degree $D+2$, so it is outperformed by $\mathrm{F}_{4/5}$ under ideal conditions (all rows are sparse). At first sight, the memory consumption of WiedemannMutantXL looks surprising. However, we are kind of cheating here as we only need to store the mutants and then compute the intermediate rows on the fly.

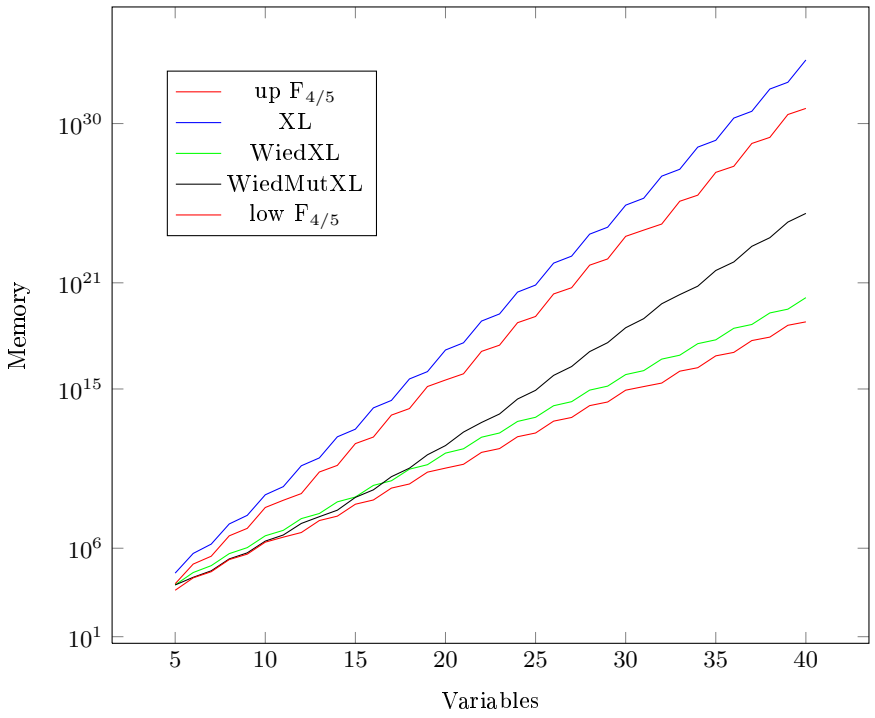

Fig. 6. Memory consumption of different algorithms for $m = n + 2$ and $q = 256$.

Acknowledgements

We want to thank Stanislav Bulygin (Cased Darmstadt, Germany) for helpful comments on an earlier version of this manuscript, Bo-Yin Yang who provided us with experimental experience and pointed out the connection of lemma 4 and his result [35, Prop.4], Ludovic Perret who helped us with the memory complexity of F_4 and J.B. Mohamed who corrected a typo in the formula of corollary 5 in one of our previous versions on ePrint [33].

The authors were funded via an DFG (German Research Foundation) Emmy Noether grant. In addition, the work described in this paper has been supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

Bibliography

[1] M. Albrecht. Algorithmic algebraic techniques and their application to block cipher cryptanalysis. PhD thesis, Royal Holloway, University of London, 2010. `http://martinralbrecht.files.wordpress.com/2010/10/phd.pdf`.

[2] M. Albrecht, C. Cid, J.-C. Faugère, and L. Perret. On the relation between the mutant strategy and the normal selection strategy in gröbner basis algorithms. Eprint Report 2011/164, 2011.

[3] G. Ars, J. charles Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between xl and gröbner basis algorithms. In *ASIACRYPT 2004, Lecture*, pages 338–353. Springer-Verlag, 2004.

[4] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.

[5] M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In P. Gianni, editor, *MEGA 2005 Sardinia (Italy)*, 2005.

[6] L. Bettale, J.-C. Faugère, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *In Journal of Mathematical Cryptology*, 3:177–197, 2009.

[7] B. Buchberger. A Theoretical Basis for the Reduction of Polynomials to Canonical Forms. *ACM SIGSAM Bull.*, 10(3):19–29, 1976.

[8] J. A. Buchmann, J. Ding, M. S. E. Mohamed, and W. S. A. E. Mohamed. Mutantxl: Solving multivariate polynomial equations for cryptanalysis. In H. Handschuh, S. Lucks, B. Preneel, and P. Rogaway, editors, *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[9] C. Cid and G. Leurent. An analysis of the xsl algorithm. In *Proceedings of Asiacrypt 2005, LNCS*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Bimal Roy, editor, Springer-Verlag, 2005. ISBN 3-540-30684-6.

[10] Computational Algebra Group, University of Sydney. *The MAGMA Computational Algebra System for Algebra, Number Theory and Geometry.* `http://magma.maths.usyd.edu.au/magma/`.

[11] N. Courtois. Algebraic attacks over GF(2^k), application to HFE challenge 2 and Sflash-v2. In *Public Key Cryptography — PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 201–217. Feng Bao, Robert H. Deng, and Jianying Zhou (editors), Springer, 2004. ISBN 3-540-21018-0.

[12] N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In

Advances in Cryptology — EUROCRYPT 2000, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Bart Preneel, editor, Springer, 2000. Extended Version: `http://www.minrank.org/xlfull.pdf`.

[13] N. T. Courtois and J. Patarin. About the XL algorithm over GF(2). In *CT-RSA'03: Proceedings of the 2003 RSA conference on The cryptographers' track*, pages 141–157, Berlin, Heidelberg, 2003. Springer-Verlag.

[14] N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Yuliang Zheng, editor, Springer, 2002.

[15] C. Diem. The XL-algorithm and a conjecture from commutative algebra. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*. Pil Joong Lee, editor, Springer, 2004. ISBN 3-540-23975-8.

[16] J. Ding, J. Buchmann, M. S. E. Mohamed, W. S. A. Moahmed, and R.-P. Weinmann. Mutantxl. In Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC08), Apr. 2008.

[17] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure anNational Institute of Standards and Technologyd Applied Algebra*, 139:61–88, June 1999.

[18] J.-C. Faugère. HFE challenge 1 broken in 96 hours. Announcement that appeared in `news://sci.crypt`, 19$^{\text{th}}$ of Apr. 2002.

[19] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *International Symposium on Symbolic and Algebraic Computation — ISSAC 2002*, pages 75–83. ACM Press, July 2002.

[20] J.-C. Faugère. Algebraic cryptanalysis of (HFE) using Gröbner bases. Technical report, Institut National de Recherche en Informatique et en Automatique, Feb. 2003. `http://www.inria.fr/rrrt/rr-4738.html`, 19 pages.

[21] J.-C. Faugère. Fast Gröbner. Algebraic cryptanalysis of HFE and filter generators. In *Workshop on Coding and Cryptography 2003*, pages 175–176. Daniel Augot, Pascal Charpin, and Grigory Kabatianski, editors, l'Ecole Supérieure et d'Appliction des Transmissions, 2003.

[22] J.-C. Faugère and G. Ars. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Rapport de recherche 4739, Feb. 2003. `www.inria.fr/rrrt/rr-4739.html`.

[23] J.-C. Faugère and A. Joux. Algebraic cryptanalysis of Hidden Field Equations (HFE) using Gröbner bases. In *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Dan Boneh, editor, Springer, 2003.

[24] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Henri Gilbert, editor, Springer, 2010. ISBN 978-3-642-13189-9.

[25] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979. ISBN 0-7167-1044-7 or 0-7167-1045-5.

[26] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem. In *Advances in Cryptology — CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Michael Wiener, editor, Springer, 1999. `http://www.minrank.org/hfesubreg.ps` or `http://citeseer.nj.nec.com/kipnis99cryptanalysis.html`.

[27] D. Lazard. Systems of algebraic equations. In *EUROSAM*, pages 88–94, 1979.

[28] F. Macauly. The algebraic theory of modular systems. In C. U. Press, editor, *Cambridge Mathematical Library*, 1916.

[29] T. Moh. On the method of "XL" and its inefficiency to TTM, 2000.

[30] M. S. Mohamed, W. S. Mohamed, J. Ding, J. Buchmann, S. Tohaneanu, R.-P. Weinmann, D. Carbarcas, and D. Schmidt. MutantXL and Mutant Gröbner Basis Algorithm. In *SCC '08: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography*, pages 16–22, 2008.

[31] W. S. A. Mohamed. *Improvements for the XL Algorithm with Applications to Algebraic Cryptanalysis*. PhD thesis, TU Darmstadt, Jun 2011. `http://tuprints.ulb.tu-darmstadt.de/2621/`.

[32] W. S. A. Mohamed, J. Ding, T. Kleinjung, S. Bulygin, and J. Buchmann. Pwxl: A parallel wiedemann-xl algorithm for solving polynomial equations over gf(2). In C. Cid and J.-C. Faugere, editors, *Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography (SCC 2010)*, pages 89–100, Jun 2010.

[33] J. B. M.S.E. Mohamed, J. Ding. The complexity analysis of the mutantxl family. Cryptology ePrint Archive, Report 2011/036, 2011. `http://eprint.iacr.org/`.

[34] S. Rønjom and H. Raddum. On the number of linearly independent equations generated by xl. In *Proceedings of the 5th international conference on Sequences and Their Applications*, SETA '08, pages 239–251, Berlin, Heidelberg, 2008. Springer-Verlag.

[35] B.-Y. Yang and J.-M. Chen. All in the XL family: Theory and practice. In *ICISC 2004*, pages 67–86. Springer, 2004.

[36] B.-Y. Yang and J.-M. Chen. Theoretical analysis of XL over small fields. In *ACISP 2004*, volume 3108 of *LNCS*, pages 277–288. Springer, 2004.