On the Number of Linearly Independent Equations Generated by XL

Sondre Rønjom and Håvard Raddum

Department of Informatics, University of Bergen, N-5020 Bergen, Norway
sondrer@ii.uib.no, haavardr@ii.uib.no

Abstract. Solving multivariate polynomial equation systems has been the focus of much attention in cryptography in the last years. Since most ciphers can be represented as a system of such equations, the problem of breaking a cipher naturally reduces to the task of solving them. Several papers have appeared on a strategy known as *eXtended Linearization* (XL) with a view to assessing its complexity. However, its efficiency seems to have been overestimated and its behaviour has yet to be fully understood. Our aim in this paper is to fill in some of these gaps in our knowledge of XL. In particular, by examining how dependencies arise from multiplication by monomials, we give a formula from which the efficiency of XL can be deduced for multivariate polynomial equations over $\mathbb{F}_2$. This confirms rigorously a result arrived at by Yang and Chen by a completely different approach. The formula was verified empirically by investigating huge amounts of random equation systems with varying degree, number of variables and number of equations.

Keywords: XL, Gröbner bases, Stream Ciphers.

1 Introduction

The problem of solving multivariate polynomial equations is encountered in many different fields. This problem has in particular received a great deal of attention in cryptography. The problem of breaking a cipher is reformulated as a problem of solving a (very large) system of polynomial equations. Solving multivariate polynomial equations over $\mathbb{F}_2$ is known to be NP-hard.

XL was proposed in [10], as a new algorithm for solving multivariate polynomial equations. A parameter D is associated with XL, and the complexity of XL is exponential in D. In [10] and [11], the authors try to evaluate for which D XL works. For a system with m equations in n variables they use the estimation $D \approx \frac{n}{\sqrt{m}}$, which seems to work fine for special cases, but for general systems this approximation becomes very inaccurate. For the AES, we estimate that $D \approx \frac{n}{\sqrt{m}} \cdot 2.5$, and for Serpent it is more like $D \approx \frac{n}{\sqrt{m}} \cdot 4.9$, which makes a huge difference, as running time is exponential in D. So it is evident that the early estimations of D are inaccurate. This is also shown in [13] where a formula for estimating D is given. This formula applies to quadratic systems over large fields and is proved to be correct given one more assumption.

S.W. Golomb et al. (Eds.): SETA 2008, LNCS 5203, pp. 239–251, 2008.

The link between XL and computing Gröbner bases was established in [12]. Their computer simulations show that XL do not seem to follow the proposed bound $D \approx \frac{n}{\sqrt{m}}$, but behave much worse. They also compare Faugères F_4 Gröbner basis algorithm with XL, and show that F_4 computes a Gröbner basis in less time, using less space. This conclusion is not very surprising, as XL tries to compute a Gröbner basis, but spends huge amount of time computing dependent equations, such that the reduction step spends an equal amount of time computing zero. The difference from other Gröbner basis algorithms is that XL contains no strategies nor avoid linear dependencies. So XL can be said to be a worst kind of Gröbner basis algorithm.

In [5], Yang and Chen describe a formula for estimating D, but as they mention, their formula is neither trivial to use nor proved to be correct.

2 Preliminaries

Let $\lambda = \mathbb{F}_2[x_1, ..., x_n]/\{x_i^2 + x_i\}_{1 \leq i \leq n}$ denote the ring of Boolean functions in n variables $x_1, x_2, \ldots, x_n$ which satisfy the relation $x_i^2 + x_i = 0$ since $x_i \in \mathbb{F}_2$. Then an $f \in \lambda$ defines a function $\mathbb{F}_2^n \xrightarrow{f} \mathbb{F}_2$, on the n-dimensional vector space $\mathbb{F}_2^n$, with values in $\mathbb{F}_2$. We define a set of polynomials by

$$F = \{f_1(x_1, ..., x_n), ..., f_m(x_1, ..., x_n)\} \subseteq \lambda$$

and the associated system of equations

$$E = \{f_1(x_1, ..., x_n) = 0, ..., f_m(x_1, ..., x_n) = 0\}.$$

We associate a function over $\mathbb{F}_2$ with its coefficient vector, so to avoid confusion we introduce notation for this.

Definition 1. *Let $f \in \lambda$. We denote by $[f]$ the vector indexed by the 2^n monomials in λ, which contains a 1 at indices where the corresponding monomial appear in f and 0 otherwise, with respect to some monomial ordering. We call $[f]$ the coefficient vector of f. When writing $\mathfrak{m}[f]$ for a monomial $\mathfrak{m}$, we will mean $[\mathfrak{m}f]$. This is well-defined, so for a polynomial g, $g[f]$ (a sum of vectors) is equal to $[gf]$.*

2.1 The XL Algorithm

Let F be a system of m polynomials of degree D_e. We associate a number $D_\mathfrak{m} \in \mathbb{N}$, such that XL is constrained by $1 \leq \deg(\mathfrak{m}) \leq D_\mathfrak{m}$, where $\mathfrak{m}$ is a monomial. Further let $D = D_e + D_\mathfrak{m}$, and let $\lambda_{\leq d}$ denote the span of the monomials of degree less than or equal to d. XL will construct a new system U_D, formed by multiplying each polynomial in F with all monomials of degree less than or equal to $D_\mathfrak{m}$, that is, $U_D = \lambda_{\leq D_\mathfrak{m}} \otimes F$. Then U_D will contain polynomials satisfying $0 \leq \deg(f_i) \leq D$. The version of XL defined in [10] consider only quadratic systems, but we generalize our analysis and consider polynomial systems of any degree D_e. XL constructs a set of polynomials U_D, satisfying the following properties:

- $F \subseteq U_D \subseteq \lambda_{\leq D}$: F is contained in U_D which is a subset of all polynomials of degree less or equal to D.
- The total number of polynomials in U_D after an execution of XL is $|U_D| = m \cdot \sum_{i=0}^{D_{\mathfrak{m}}} \binom{n}{i}$, but these are not necessarily linearly independent.

The XL algorithm can now be described as follows (adapted from original paper):

Algorithm (The XL algorithm). For a positive integer $D_{\mathfrak{m}}$ and a set of polynomial equations $F = \{f_1, \ldots, f_m\}$, execute the following steps:

1. *Multiply*: Generate all the products $\mathfrak{m} \cdot f_i \in U_D, 1 \leq \deg(\mathfrak{m}) \leq D_{\mathfrak{m}}, f_i \in F$
2. *Linearize*: Consider each monomial $\mathfrak{m}$ with $deg(\mathfrak{m}) \leq D$ as a new variable and perform Gaussian elimination on the equations obtained in Step 1.
3. *Solve or Repeat*: If the system is not solved, increase $D_{\mathfrak{m}}$ and redo step 1 and 2.

Remark 1. Setting $D_e = 2$ gives the algorithm from [10].

The idea behind the algorithm is that new linearly independent equations are in fact generated. But this comes at the expense of increasing the total degree D of the system U_D. In [12] it is shown that XL is basically the construction of the Macaulay matrix. Lazard (see [17] and [18]) describes the link between linear algebra and computing Gröbner bases and proves that Gaussian elimination on the Macaulay matrix for D large enough returns a Gröbner basis. Thus the problem presented above can be reformulated as:

Problem 1. For which D will the XL algorithm, after a row reduction of U_D with respect to a monomial ordering, return a matrix with full rank?

It should be noted that the fastest Gröbner basis algorithm, Faugeres F_5 algorithm, is based on the same type of computation as XL. However, F_5 differs from XL in that it avoids computing unnecessary linear dependencies. But our result applies also to the case of F_5, as both algorithms reach the same degree D before a linear basis can be computed.

2.2 Restrictions on F

Exact analysis becomes very hard, if not impossible, if we assume nothing about the polynomials in F. The best approach is to define a sufficiently general restriction on the system F in which the algorithm can be analyzed in fair terms. In this section we define some restrictions on F, which is almost always fulfilled for systems coming from cryptography.

First, we put the following restrictions on F:

- Total degree: $\deg(f_i) = D_e, \forall f_i \in F$.
- All $[f_i]$ are linearly independent.

We will also restrict F by defining the types of relations which occur between the polynomials.

Construct the coefficient matrix H_0 of the set U_D, and assume $D_{\mathfrak{m}} \geq D_e$. The rows of H_0 consist of all vectors $\mathfrak{m}[f_j]$, with $1 \leq j \leq m$ and $0 \leq \deg(\mathfrak{m}) \leq D_{\mathfrak{m}}$. Since we have formed products of f_j with all monomials of degree $0 \leq \deg(\mathfrak{m}) \leq \deg(f_i)$, some of the rows in H_0 will add together to $[f_j]f_i$. We can do the same starting with f_i and construct some coefficient vectors that add up to $[f_i]f_j$. Since $[f_i]f_j = [f_j]f_i$ we have identified a linear dependency among some rows of H_0.

During the execution of XL, as long as $D_{\mathfrak{m}} \geq D_e$, we will always create linear dependencies of the form

$$[f_j] \cdot f_i + [f_i] \cdot f_j = 0.$$

The relation

$$f_i \cdot [f_i] + [f_i] = 0$$

also occurs since we have that $x_i^2 + x_i = 0$. These relations are trivial in that they will always exist for the equation systems over λ.

3 Equation Systems from LFSR-Based Stream Ciphers

Equations coming from LFSR-based stream ciphers are examples of equations that are very interesting with respect to an XL-type algorithm. Assume we are given a set of equations $F = \{f_1 = 0, f_2 = 0, \ldots, f_m = 0\}$ coming from a regularly clocked filter generator. Let $g(x) \in \mathbb{F}_2[x]$ denote a primitive generator of $\mathbb{F}_{2^n}$ such that the binary sequence $s_t = \sum_{i=0}^{n-1} s_{t-n+i}c_i$ is a sequence with maximal period $2^n - 1$ (an m-sequence). Then let $f(x_1, \ldots, x_n)$ denote a Boolean filter function of degree d. Then the keystream-sequence $z = \{z_t\}_{t=0,1,\ldots}$ is generated by

$$z_t = f(s_t, s_{t+1}, \ldots, s_{t+n-1}),$$

which can be represented by the equations $F = f_t(s_0, \ldots, s_{n-1}) + z_t = 0_{t=0,1,\ldots}$, where $f(s_t, \ldots, s_{t+n-1}) = f_t(s_0, \ldots, s_{n-1})$, since s_t can be written in terms of the initial state bits $s_0, s_1, \ldots, s_{n-1}$. Let $W_d = \sum_{i=1}^{d} \binom{n}{i}$ denote the number of monomials of degree less or equal to d. Then the coefficient vector $[f]$ may be restricted to a length W binary vector. Notice that in the direct algebraic attack we need W equations in order to solve the system using a naive linear algebra attack. Following [RonHel], there exist a $W \times W$ linear matrix operator T, which is invariant of the filtering function $f(x_1, \ldots, x_n)$, but variant of the polynomial $g(x)$. Using their notation, we may instead write the sequence z_t by

$$\begin{aligned} z_t &= f(s_t, s_{t+1}, \ldots, s_{t+n-1}) \\ &= s_t^*[f_0] \\ &= s_0^* T^t[f] \\ &= s_0^*[f_t], \end{aligned}$$

where s_t^* is the expanded state vector $[s_t, \ldots, s_{t+n-1}, s_t s_{t+1}, \ldots, s_{t+n-1} \cdots s_{t+n-d}]$. Thus, the equations in F are simply generated by $F = \{T^t f\}_{0 \leq t \leq m}$ for some $m \leq W$. In [19], the exact rank of the equation system F is determined. If $l(z)$ denotes the degree of the minimal polynomial of the sequence $z = \{z_t\}_{t=0,1,\ldots}$, then we have that $\dim(F) = m$ where $0 \leq m \leq ls(z) \leq W$. There are basically two cases to consider:

- A. if $m = l(z) \leq W$, then the system can be solved using a linear subspace attack or variations (see for instance [20]).
- B. if $m < l(z) \leq W$, then the system of m linearly independent equations may be solved using an XL-type algorithm.

This just tells us that the systems coming from the filter generator satisfy the two first restrictions in our analysis; that they are linearly independent and have the same degree. But there is a possibility that the system may contain nonlinear dependencies to begin with. For instance, the sequence z may satisfy a nonlinear recursion $z_{t+r} = h(z_t, z_{t+1}, \ldots z_{t+r-1}), 0 \leq r \leq m$, meaning that $f_{t+r} = h(f_t, f_{t+1}, \ldots, f_{t+r-1}), 0 \leq r \leq m$. If such a relation exist, we do not need to use a method like XL, since we may generate as many equations as we need using the nonlinear recursion h. On the other hand, determining such relations on practical systems seems to be a very hard problem.

But for a D_m and D_e where $D = D_m + D_e$, then if there are no nonlinear relations between the m equations in F of degree $\lfloor \frac{D_m}{D_e} \rfloor$, our estimate of the number of linearly independent equations will be exact since the system will only contain relations that we construct ourselves. Thus if the smallest degree of a nonlinear relation between the equations in F is k, then our analysis will be exact when applying XL with $D_m = k \cdot D_e - 1$, but not for $D_m = k \cdot D_e$.

4 Preliminary Observations

Before we present an explanation of XLs behaviour, we give an intuitive presentation of how to count the dependencies which occur in the application of an algorithm such as XL.

We assume that the systems we study behave according to our assumptions in Section 2 and use the notation introduced in Section 2. We will look at the analysis of XL on quadratic systems presented in [11], and correct a mistake made there. This will help to see how to systematically count the number of linear dependencies created by XL.

In the following, we set $D_e = 2$ and $D = D_{\mathfrak{m}} + D_e$. In [11] they use R to count the total number of equations, and set S to be the number of dependencies and I the number of linearly independent equations generated by XL, such that

$$I = R - S.$$

The authors do computer simulations on quadratic polynomial equations for $3 \leq D \leq 6$ and identify some relations. For instance, for $D = 4$ they identify two dependencies:

1. $f_i[f_j] + f_j[f_i] = 0$,
2. $f_i[f_i] + [f_i] = 0$.

For $D = 5$ they identify additionally two types of dependencies:

3. $f_i[f_j]x_k + [f_i]f_j x_k = 0$
4. $f_i[f_i]x_k + [f_i]x_k = 0$

The authors conclude that these are the only existing dependencies for $D = 4$ and $D = 5$, and verify that their estimations are coherent with computer simulations. For $D = 4$ there are $\binom{m}{2}$ ways of constructing relations on the form 1 and m ways of constructing relations of the form 2. For $D = 5$ these numbers are multiplied with the $n+1$ monomials of degree ≤ 1 to form additional types of dependencies:

– Case $D = 4$: $I = R - \binom{m}{2} - m$
– Case $D = 5$: $I = R - (n+1)\binom{m}{2} - (n+1)m$.

For $D = 6$ they state that the number of linearly independent equations is

$$I = R - \left(\binom{n}{2} + \binom{n}{1} + 1\right) \cdot \left(\binom{m}{2} + m\right). \tag{1}$$

At this point we step in and show where their analysis becomes wrong. They conclude that the only relations will be multiples of $f_i[f_j] + f_j[f_i] = 0$. It seems reasonable to assume that they have drawn this conclusion based on Buchberger's two criterion, but this is not correct and will turn out fatal in further analysis for larger D. Their formulas are indeed correct for $3 \leq D \leq 5$, but for $D = 6$ they forget to count the *dependencies among the dependencies*. By a slight abuse of notation, which will be clarified in the next section, these dependencies may be expressed as follows (the number of such dependencies is indicated in the brackets to the right):

5. $f_i[f_j[f_k]] + [f_i[f_j]]f_k + [[f_i]f_k]f_j = 0$ $(\binom{m}{3})$
6. $[[f_i]f_j]f_j + [[f_i]f_j] = 0$ $(2 \cdot \binom{m}{2})$
7. $[[f_i]f_i]f_i + [[f_i]] = 0$ $(\binom{m}{1})$

This means we count $\binom{m+2}{3}$ of the dependent equations twice, so we need to balance this by calculating from an inclusion/exclusion point of view. Using the authors notation, the correct bound should have been:

$$I = R - \left(\binom{n}{2} + \binom{n}{1} + 1\right) \cdot \left(\binom{m}{2} + m\right) + \left(\binom{m}{3} + 2\binom{m}{2} + \binom{m}{1}\right) \tag{2}$$

Note that the formulas above for $3 \leq D \leq 6$ works only for quadratic equations. The dependencies behave with respect to the degree D_e of the initial system E. If for instance $D_e = 7$ there would be constructed no dependencies applying XL with $D_{\mathtt{m}} \leq 6$. If we work with linear equations, XL will introduce new dependencies for each time we increase $D_{\mathtt{m}}$.

5 The Number of Linearly Independent Equations in XL

In this section we will estimate the number of linearly independent equations one generates with the XL-method. As we will see, the linear dependencies we can identify is governed by products of polynomials from F, and the number of dependencies is calculated by counting the number of such products. We multiply each f_i with all monomials of degree $\leq D_{\mathfrak{m}}$ (including the monomial 1, which keeps the original polynomials) to form the set U_D.

Let H_0 be the matrix whose rows are the coefficient vectors of all polynomials in U_D. The columns of H_0 are indexed by all monomials of degree $\leq D_{\mathfrak{m}} + D_e$. To avoid confusion in the generalization that follows, the rows of H_0 are indexed by $\mathfrak{m} \cdot r(f_i)$, instead of $\mathfrak{m}[f_i]$. The entry $(\mathfrak{m} \cdot r(f_i), \mathfrak{m}')$ is 1 if $\mathfrak{m}'$ occurs as a term in $\mathfrak{m} \cdot f_i$ and 0 otherwise.

We will now recursively construct a sequence of binary matrices $H_i, i \geq 1$. The rows of H_i will be indexed by $\mathfrak{m} \cdot r(f_{i_1}^{e_1} f_{i_2}^{e_2} \cdots f_{i_s}^{e_s})$, where $\sum_{j=1}^{s} e_j = i+1$ and $\mathfrak{m}$ is a monomial with $\deg(\mathfrak{m}) \leq D_{\mathfrak{m}} - i \cdot D_e$. The columns of H_i will have the same indices as the rows of H_{i-1}. The degree of $\mathfrak{m}$ needs to be non-negative, so the final H_i we construct is for $i = \lfloor \frac{D_{\mathfrak{m}}}{D_e} \rfloor$. When writing $g \cdot r(\cdots)$ for a polynomial $g = \mathfrak{m}_1 + \ldots + \mathfrak{m}_k$, we will mean the sum $\mathfrak{m}_1 \cdot r(\cdots) + \ldots + \mathfrak{m}_k \cdot r(\cdots)$. The row $\mathfrak{m} \cdot r(f_{i_1}^{e_1} f_{i_2}^{e_2} \cdots f_{i_s}^{e_s})$ will contain a 1 in all columns that occur as terms in the following sum:

$$\mathfrak{m} \cdot \sum_{j=1}^{s} (f_{i_j} + (e_{i_j} - 1 \mod 2)) r(f_{i_1}^{e_1} \cdots f_{i_j}^{e_j - 1} \cdots f_{i_s}^{e_s}). \tag{3}$$

The rest of the entries in row $\mathfrak{m} \cdot r(f_{i_1}^{e_1} f_{i_2}^{e_2} \cdots f_{i_s}^{e_s})$ will contain 0.

Let $\mathbf{v}$ be a binary vector indexed by the rows of H_i. If $\mathbf{v} \cdot H_i = \mathbf{0}$, $\mathbf{v}$ identifies a linear dependency between the rows of H_i. With the matrices $H_i, i = 0, \ldots, \lfloor \frac{D_{\mathfrak{m}}}{D_e} \rfloor$ defined, we are ready to prove the first result.

Theorem 1. *$H_i \cdot H_{i-1} = [\mathbf{0}]$, the all-zero matrix for $i = 1, \ldots, \lfloor \frac{D_{\mathfrak{m}}}{D_e} \rfloor$. That is, each row of H_i identifies a linear dependency among the rows of H_{i-1}.*

Proof. The row $\mathfrak{m} \cdot r(f_{i_1}^{e_1} f_{i_2}^{e_2} \cdots f_{i_s}^{e_s})$ in H_i contains a 1 in the columns indexed by the terms in the following sum

$$\mathfrak{m} \cdot \sum_{j=1}^{s} (f_{i_j} + (e_{i_j} - 1 \mod 2)) r(f_{i_1}^{e_1} \cdots f_{i_j}^{e_j - 1} \cdots f_{i_s}^{e_s}). \tag{4}$$

If $e_j = 1$, it means that $f_{i_j}^{e_j - 1}$ vanishes from the expression $r(\cdots)$. Assume without loss of generality that $e_{t+1} = \ldots = e_s = 1$, and that $e_j > 1, j = 1, \ldots, t$. We need to show that (4) is the all-zero vector when the terms are regarded as rows in H_{i-1}. We substitute $r(f_{i_1}^{e_1} \cdots f_{i_j}^{e_j - 1} \cdots f_{i_s}^{e_s})$ with the expression given by (3) to find which columns in H_{i-1} that contain a 1 in the rows found in (4).

We then examine the parity of the number of 1's in these columns. After substituting, (4) can be written as

$$\mathfrak{m}\cdot\sum_{j=1}^{s}(f_{i_j}+(e_j-1 \mod 2))\sum_{k=1,k\neq j}^{s}(f_{i_k}+(e_k-1 \mod 2))r(\cdots f_{i_j}^{e_j-1}\cdots f_{i_k}^{e_k-1}\cdots)$$

$$+\mathfrak{m}\cdot\sum_{j=1}^{t}(f_{i_j}+(e_j-1 \mod 2))(f_{i_j}+(e_j-2 \mod 2))r(\cdots f_{i_j}^{e_j-2}\cdots). \quad (5)$$

Each term in (5) represents a 1 in the column in H_{i-1} corresponding to the same term. In the double sum where $j \neq k$, each $r(\cdots f_{i_j}^{e_j-1}\cdots f_{i_k}^{e_k-1}\cdots)$ will occur exactly twice, once when $j < k$ and once when $j > k$. Both times the polynomials to be multiplied with $r(\cdots f_{i_j}^{e_j-1}\cdots f_{i_k}^{e_k-1}\cdots)$ will be $\mathfrak{m}(f_{i_j}+(e_j-1 \mod 2))(f_{i_k}+(e_k-1 \mod 2))$, so the number of 1's in columns involving $r(\cdots f_{i_j}^{e_j-1}\cdots f_{i_k}^{e_k-1}\cdots)$ is even. For the remaining single sum, we note that $f_{i_j}f_{i_j}=f_{i_j}$ and that exactly one of e_j-1 and e_j-2 is $1 \mod 2$ and the other is $0 \mod 2$. Multiplying out the brackets in the single sum we get $\mathfrak{m}(f_{i_j}+f_{i_j})=0 \mod 2$ in front of each $r(\cdots f_{i_j}^{e_j-2}\cdots)$, so the number of 1's in columns involving $r(\cdots f_{i_j}^{e_j-2}\cdots)$ is also even. □

We are now ready to proceed to the main result, an estimation of the number of linearly independent equations generated by the XL-method.

Theorem 2. *Let I be the number of linearly independent equations generated by the XL-method on a system of m equations in n variables, where each equation has degree D_e and we multiply with all monomials of degree $\leq D_{\mathfrak{m}}$. If the only linear dependencies among the rows of H_{i-1} are the ones indicated by H_i, then*

$$I=\sum_{i=0}^{\lfloor\frac{D_{\mathfrak{m}}}{D_e}\rfloor}(-1)^i\binom{m+i}{i+1}\sum_{j=0}^{D_{\mathfrak{m}}-i\cdot D_e}\binom{n}{j}.$$

Proof. By the construction of the matrices H_i, we have $I=\mathrm{rank}(H_0)$. Let the number of rows in H_i be b_i. If all the rows of H_i are linearly independent we will have $\mathrm{rank}(H_{i-1})=b_{i-1}-b_i$. However, there will in general be linear dependencies also between the rows of H_i, so the correct expression will be

$$\mathrm{rank}(H_{i-1})=b_{i-1}-\mathrm{rank}(H_i), i=1,\ldots\lfloor\frac{D_{\mathfrak{m}}}{D_e}\rfloor. \quad (6)$$

The matrix $H_{\lfloor\frac{D_{\mathfrak{m}}}{D_e}\rfloor}$ will have full rank $b_{\lfloor\frac{D_{\mathfrak{m}}}{D_e}\rfloor}$ since there is no H-matrix coming after it. By recursively using (6) for substituting the expressions for $\mathrm{rank}(H_i)$ we have the following formula

$$I=\mathrm{rank}(H_0)=\sum_{i=0}^{\lfloor\frac{D_{\mathfrak{m}}}{D_e}\rfloor}(-1)^i b_i. \quad (7)$$

To finish, we need to compute b_i, the number of rows in H_i. The rows of H_i are indexed by $\mathfrak{m} \cdot r(f_{i_1}^{e_1} f_{i_2}^{e_2} \cdots f_{i_s}^{e_s})$, where $\deg(\mathfrak{m}) \leq D_{\mathfrak{m}} - i \cdot D_e$ and $\sum_{j=1}^{s} e_j = i+1$. The number of choices for $\mathfrak{m}$ is $\sum_{j=0}^{D_{\mathfrak{m}} - i \cdot D_e} \binom{n}{j}$. The number of ways to make a product of $i+1$ equations by picking equations from the total set of m equations is the number of ways to throw $i+1$ balls into m bins. This number is $\binom{m+i}{i+1}$ by [9]. We then get

$$b_i = \binom{m+i}{i+1} \sum_{j=0}^{D_{\mathfrak{m}} - i \cdot D_e} \binom{n}{j},$$

and substituting this into (7) gives us the desired expression. □

We restrict our analysis to systems which only contain trivial dependencies. This means that our formula is not correct for initial systems containing other types of dependencies. One example of non-trivial dependencies are systems containing nonlinear dependencies.

6 Linking Theorem 2 to Theorem/Conjecture from Related Work

We are aware of three papers by Yang and Chen (one also with Courtois) [5], [6] and [7] which, among other things, try to estimate the number of linearly independent equations generated by the XL-method. Their result uses the notation $[t^D]p(t)$, which represents the coefficient of the D'th-degree term in the polynomial (or series) $p(t)$. For an instance of the XL algorithm where the initial equations all have degree k, let T be the number of monomials generated, and let I be the number of linearly independent equations. For $\mathbb{F}_2$, their result from [5] is as follows.

Theorem 3

$$T - I \geq [t^D] \left(\frac{1}{1-t} \left(\frac{1-t^2}{1-t} \right)^n \left(\frac{1-t^k}{1-t^{2k}} \right)^m \right)$$

when $D < D_{reg}$, where D_{reg} is the degree of the first term with a negative coefficient in the series.

In [5] there is a proof of the theorem for $k = 2$, but this proof has been shown to be flawed. In [6] they write "As pointed out by C. Diem, the [5] proof is inaccurate.... In any event, since it is also confirmed by many simulations we will henceforth assume Theorem 3 holds in general..." In [7] they write "The [5] proof was faulty..."

Theorem 3 seeks to give an upper bound on the number of linearly independent equations we can get from XL. With the extra assumption in Theorem 2, saying that the only linear dependencies occurring are the ones we can identify, Theorem 3 could be stated with equality. Below we will find the link between Theorems 2 and 3, showing that these two results indeed say the same things

when assuming only trivial linear dependencies. However, we thing that it is easier to use Theorem 2 as one can plug in the numbers for a particular system and do the simple arithmetic to get the expected number of linearly independent equations. To find the same thing using Theorem 3, one needs to expand a complicated series to find the coefficient of a particular term. We start by computing the D'th degree coefficient in the series from Theorem 3

Proposition 1

$$[t^D]p(t) = [t^D]\left(\frac{1}{1-t}\left(\frac{1-t^2}{1-t}\right)^n\left(\frac{1-t^k}{1-t^{2k}}\right)^m\right)$$
$$= \sum_{i=0}^{\lfloor \frac{D}{k} \rfloor}(-1)^i\binom{m+i-1}{i}\sum_{j=0}^{D-ik}\binom{n}{j}.$$

Proof The first fraction in $p(t)$ can be written as $\frac{1}{1-t} = \sum_{l=0}^{\infty} t^l$. The second fraction can be expressed as $\left(\frac{1-t^2}{1-t}\right)^n = (1+t)^n = \sum_{j=0}^{n}\binom{n}{j}t^j$. The third fraction can be expressed as $\left(\frac{1-t^k}{1-t^{2k}}\right)^m = (1-(-t^k))^{-m}$. By [8], this is equal to $\sum_{i=0}^{\infty}(-1)^i\binom{m+i-1}{i}t^{ik}$. Since we are only interested in $[t^D]p(t)$ where $D \leq n$, we can cut away terms of degree higher than D in the three sums to get

$$[t^D]p(t) = [t^D]\left(\sum_{l=0}^{D} t^l\right)\left(\sum_{j=0}^{D}\binom{n}{j}t^j\right)\left(\sum_{i=0}^{\lfloor \frac{D}{k} \rfloor}(-1)^i\binom{m+i-1}{i}t^{ik}\right) =$$

$$[t^D]\left(\sum_{l=0}^{D} t^l\sum_{j=0}^{l}\binom{n}{j}\right)\left(\sum_{i=0}^{\lfloor \frac{D}{k} \rfloor}(-1)^i\binom{m+i-1}{i}t^{ik}\right).$$

We are looking for the coefficient of the D'th degree term, so we need only multiply the two sums together with the constraint $l = D - ik$. Taking away the sum over l, substituting $D - ik$ for l in the rest of the first sum and multiplying together we get the desired expression for the D'th degree coefficient. □

The number D is the maximum degree of a monomial when running the XL algorithm. Relating this to our notation we have $D = D_e + D_{\mathtt{m}}$ and $k = D_e$. We then get the following result.

Corollary 1. *Let $D = D_e + D_{\mathtt{m}}$ and $k = D_e$, then*

$$T - I = [t^D]\left(\frac{1}{1-t}\left(\frac{1-t^2}{1-t}\right)^n\left(\frac{1-t^k}{1-t^{2k}}\right)^m\right)$$

is equivalent to

$$I = \sum_{i=0}^{\lfloor \frac{D_{\mathtt{m}}}{D_e} \rfloor}(-1)^i\binom{m+i}{i+1}\sum_{j=0}^{D_{\mathtt{m}}-ik}\binom{n}{j}.$$

Proof. Rearranging the first expression, using the proposition and substituting for D and k we get

$$I = T - \sum_{i=0}^{\lfloor \frac{D_{\mathtt{m}}}{D_e} \rfloor + 1} (-1)^i \binom{m+i-1}{i} \sum_{j=0}^{D_{\mathtt{m}} - (i-1)D_e} \binom{n}{j}.$$

The T can be expressed as $T = \sum_{j=0}^{D_e + D_{\mathtt{m}}} \binom{n}{j}$, which is the term for $i = 0$ in the sum. Canceling these terms we are left with

$$I = - \sum_{i=1}^{\lfloor \frac{D_{\mathtt{m}}}{D_e} \rfloor + 1} (-1)^i \binom{m+i-1}{i} \sum_{j=0}^{D_{\mathtt{m}} - (i-1)D_e} \binom{n}{j}.$$

We can let the sum start at $i = 0$ by subtracting 1 from the upper limit for i and increasing each i in the rest of the expression by 1. Compensating for the minus sign in front of the sum we get

$$I = \sum_{i=0}^{\lfloor \frac{D_{\mathtt{m}}}{D_e} \rfloor} (-1)^i \binom{m+i}{i+1} \sum_{j=0}^{D_{\mathtt{m}} - iD_e} \binom{n}{j}. \qquad \square$$

The corollary shows that Theorems 3 and 2 are basically the same result. With the proof of Theorem 2 we have also proved Theorem 3 with equality when assuming that only trivial linear dependencies occur. Many computer experiments on random systems (with no restrictions on the dependencies), counting the number of linearly independent equations have been done, both by us and the authors of Theorem 3. In these experiments it has never occurred that Theorem 3 fails, so we believe that the result is correct and end this section with the following conjecture.

Conjecture 1. Let I be the number of linearly independent polynomials in n variables generated by step 1 of the XL algorithm, where all m initial equations have degree D_e and we multiply with all monomials up to degree $D_{\mathtt{m}}$. Then

$$I \leq \sum_{i=0}^{\lfloor \frac{D_{\mathtt{m}}}{D_e} \rfloor} (-1)^i \binom{m+i}{i+1} \sum_{j=0}^{D_{\mathtt{m}} - ik} \binom{n}{j}.$$

7 Conclusions

The work in this paper comes from the field of cryptography, in particular algebraic attacks on symmetric key ciphers. The complexity of such attacks has been hard to estimate, but this paper shows the XL-algorithm generates a lot of linearly dependent equations and is not as efficient as initially hoped.

Using the formula in Theorem 2, we can predict the smallest D for which the XL algorithm will work over $\mathbb{F}_2$. The formula is easier to use than that of Yang and Chen, since no multiplication of polynomials or series is involved, but only simple arithmetic.

References

1. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
2. Courtois, N.: Higher Order Correlation Attacks,XL Algorithm and Cryptanalysis of Toyocrypt. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 182–199. Springer, Heidelberg (2003)
3. Courtois, N.: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
4. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
5. Yang, B.-Y., Chen, J.-M.: Theoretical Analysis of XL over Small Fields. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 277–288. Springer, Heidelberg (2004)
6. Yang, B.-Y., Chen, J.-M., Courtois, N.: On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 401–413. Springer, Heidelberg (2004)
7. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
8. Anderson, I.: A First Course in Combinatorial Mathematics, Theorem 2.6, 2nd edn., p. 16. Oxford University Press, Oxford (1989)
9. van Lint, J.H., Wilson, R.M.: A Course in Combinatorics, Theorem 13.1, 2nd edn., p. 119. Cambridge University Press, Cambridge (2001)
10. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
11. Courtois, N., Patarin, J.: About the XL Algorithm over GF(2). In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 141–157. Springer, Heidelberg (2003)
12. Ars, G., Faugre, J.C., Imai, H., Kawazoe, M., Sugita, M.: Comparison Between XL and Gröbner Basis Algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 338–353. Springer, Heidelberg (2004)
13. Moh, T.: On The Method of "XL" And Its Inefficiency to TTM, IACR eprint server (2001), `http://eprint.iacr.org/2001/047`
14. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
15. Macaulay, F.S.: On some formula in elimination. In: Proceedings of London Mathematical Society, pp. 3–27 (1902)
16. Bardet, M., Faugre, J.C., Salvy, B.: Complexity of Gröbner basis computation for Semi-regular Overdetermined sequences over F_2, with solutions in F_2 Rapport de recherche de l'INRIA, No. 5049 (2003)
17. Lazard, D.: Gröbner-Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In: van Hulzen, J.A. (ed.) EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)

18. Lazard, D.: Rèsolution des systémes d'équations algébriques. Theoretical Computer Science 15(1) (1981)
19. Rønjom, S., Helleseth, T.: The Linear Vector Space Spanned by the Nonlinear Filter Generator. In: Golomb, S.W., Gong, G., Helleseth, T., Song, H.-Y. (eds.) SSC 2007. LNCS, vol. 4893, pp. 169–183. Springer, Heidelberg (2007)
20. Rønjom, S., Gong, G., Helleseth, T.: On attacks on filtering generators using linear subspace structures. In: Golomb, S.W., Gong, G., Helleseth, T., Song, H.-Y. (eds.) SSC 2007. LNCS, vol. 4893, pp. 204–217. Springer, Heidelberg (2007)