MXL$_3$: An Efficient Algorithm for Computing Gröbner Bases of Zero-Dimensional Ideals

Mohamed Saied Emam Mohamed[1], Daniel Cabarcas[2], Jintai Ding[2], Johannes Buchmann[1], and Stanislav Bulygin[3]

[1] TU Darmstadt, FB Informatik
Hochschulstrasse 10, 64289 Darmstadt, Germany
{mohamed,buchmann}@cdc.informatik.tu-darmstadt.de

[2] Department of Mathematical Sciences, University of Cincinnati,
South China University of Technology
cabarcd@email.uc.edu, jintai.ding@uc.edu

[3] Center for Advanced Security Research Darmstadt (CASED)
Stanislav.Bulygin@cased.de

Abstract. This paper introduces a new efficient algorithm, called MXL$_3$, for computing Gröbner bases of zero-dimensional ideals. The MXL$_3$ is based on XL algorithm, mutant strategy, and a new sufficient condition for a set of polynomials to be a Gröbner basis. We present experimental results comparing the behavior of MXL$_3$ to F$_4$ on HFE and random generated instances of the MQ problem. In both cases the first implementation of the MXL$_3$ algorithm succeeds faster and uses less memory than Magma's implementation of F$_4$.

Keywords: Multivariate polynomial systems, Gröbner basis, XL algorithm, Mutant, MutantXL algorithm.

1 Introduction

The standard way to represent the polynomial ideals is to compute a Gröbner basis of it. One of the most useful applications of Gröbner bases is to compute efficiently the variety of the ideal. This leads to solving the polynomial system induced by the ideal.

The Buchberger algorithm [4] was the first algorithm for computing Gröbner bases. It is based on the computation of Gröbner bases using s-polynomials. F$_4$ [11] is an algorithm that uses linear algebra and Buchberger's s-polynomial techniques to compute Gröbner bases.

XL was introduced in [6] as an efficient algorithm for solving polynomial equations in case only a single solution exists. The MutantXL algorithm was proposed as a variant of XL that is based on the mutant strategy [9,8]. The MXL$_2$ algorithm [16] is an improvement to MutantXL that uses the partial enlargement technique [16] and a necessary number of mutants to solve.

As explained in [13,17] the XL algorithm calculates the reduced Gröbner basis in the case of a single solution. So, we wonder if a variant of the XL algorithm can compute Gröbner bases in a more general case.

© Springer-Verlag Berlin Heidelberg 2010

The comparison of XL and F_4 in [13,17] concluded that F_4 computes a Gröbner basis faster and uses less memory resources than XL. By combining the mutant strategy with the XL algorithm, it was shown in [9] that MutantXL outperforms XL in all cases. Moreover the results presented in [16] showed that MXL_2 outperforms F_4 for all the random systems and 57% of the HFE cases that are considered in that paper. Another indicator for the fact that a variant of MutantXL outperforms F_4 is in [15]. So this variant of the XL algorithm is a good candidate to be adapted for computing Gröbner bases.

In this paper we introduce a new efficient algorithm for computing Gröbner bases of zero-dimensional ideals that we call MXL_3. The MXL_3 algorithm uses the MutantXL strategy, MXL_2 improvements, and a new efficiently checkable condition to test whether a set of polynomials is a Gröbner basis. We give an experimental comparison between the first implementation of the MXL_3 algorithm and Magma's implementation of the F_4 algorithm on some HFE cryptosystems and some randomly generated instances of the MQ problem. We show that for the HFE systems MXL_3 can solve systems of univariate degree 288 that have number of variables up to 49 while Magma's F_4 can not solve any system with more than 39 variables under the same memory constraints. Moreover, we show that MXL_3 solves the HFE challenge 1 using a smaller matrix dimensions than Magma's F_4.

This paper is organized as follows. In Section 2 we give an overview of Gröbner bases and present the new condition to test whether a set of polynomials is a Gröbner basis. In Section 3 we review the XL algorithm, mutant strategy, and the MXL_2 improvements. In Section 4 we describe the MXL_3 algorithm. In Section 5 we give our experimental results on random and HFE systems and finally we conclude the paper in Section 6.

2 Gröbner Bases

We adopt the notation and use some of the results from [2]. Let K be the ground field, and let the polynomial ring $K[x_1, \ldots, x_n]$ over K be denoted by $K[\underline{x}]$. A term in the indeterminates $x_1, \ldots, x_n$ is a power product of the form $x_1^{e_1} \cdots x_n^{e_n}$ with $e_i \in \mathbb{N}$. We denote by T the set of all terms. A monomial is any product of a field element and a term. Let $\leq$ denote a term order on T. The degree of $t = x_1^{e_1} \cdots x_n^{e_n} \in T$ is defined by $\deg(t) := \sum_{i=1}^{n} e_i$. For $f = \sum_{t \in T} c_t t \in K[\underline{x}]$, where $c_t \in K$ is the coefficient of t in f, we define the terms of f by $T(f) := \{t \in T \mid c_t \neq 0\}$, the degree of f by $\deg(f) := \max\{\deg(t) \mid t \in T(f)\}$, the head term of f by $\mathrm{HT}(f) := \max_{\leq} \mathrm{T}(f)$, the head coefficient of f, denoted by $\mathrm{HC}(f)$, is the coefficient of the head term, and the head monomial of f is $\mathrm{HM}(f) := \mathrm{HC}(f)\,\mathrm{HT}(f)$. If $f, g \in K[\underline{x}]$ the s-polynomial of f and g is defined as $\mathrm{spol}(f, g) = \frac{t}{\mathrm{HM}(f)} f - \frac{t}{\mathrm{HM}(g)} g$, with $t := \mathrm{lcm}(\mathrm{HT}(f), \mathrm{HT}(g))$.

Given a subset P of $K[\underline{x}]$, we denote by $\langle P \rangle$ the ideal generated by P, and by $\mathrm{HT}(P)$ the set of head terms from elements in P. We denote by $\mathrm{span}_K(P)$ the K-linear span of P. We will denote by $P_{(op)d}$ the subset of all the polynomials of degree $(op)d$ in P, where (op) is any of $\{=, <, >, \leq, \geq\}$.

Definition 1. *A finite subset G of an ideal I of the polynomial ring $K[\underline{x}]$ is called a Gröbner Basis for I (w.r.t the term order $\leq$) if*

$$\langle \mathrm{HT}(G) \rangle = \langle \mathrm{HT}(I) \rangle \,.$$

A finite subset $\widetilde{H}$ of $K[\underline{x}]$ is a row echelon form *of H w.r.t. $\leq$ if span$(\widetilde{H})$ = span(H) and elements of $\widetilde{H}$ have pairwise different leading terms.*

Definition 2. *Let P be a finite subset of $K[\underline{x}]$, $0 \neq f \in \langle P \rangle$ and $t \in T$. A representation*

$$f = \sum_{i=1}^{s} a_i t_i p_i$$

with $a_i \in K$, $t_i \in T$, and $p_i \in P$ is called a t-representation of f w.r.t. P (and $\leq$) if $\mathrm{HT}(t_i p_i) \leq t$ *for $i = 1, \ldots, s$. A* $\mathrm{HT}(f)$*-representation of f w.r.t. P is called a* standard representation.

Proposition 1. *[2] Let G be a finite subset of $K[\underline{x}]$ with $0 \notin G$, and assume that for all $g_1, g_2 \in G$,* $\mathrm{spol}(g_1, g_2)$ *equals zero or has a standard representation w.r.t. G. Then G is a Gröbner basis.*

We recall a result commonly known as Buchberger's second criterion. We paraphrase it in the following proposition.

Proposition 2. *[5,2] let F be a finite subset of $K[\underline{x}]$ and $g_1, p, g_2 \in K[\underline{x}]$ be such that $HT(p) \mid \mathrm{lcm}(\mathrm{HT}(g_1), \mathrm{HT}(g_2))$, and for $i = 1, 2$ $\mathrm{spol}(g_i, p)$ has a standard representation w.r.t. F, then* $\mathrm{spol}(g_1, g_2)$ *also has a standard representation w.r.t. F.*

In the rest of this paper we will be working with the total-degree orderings of terms. So by "order" we mean "total-degree order" here. In the total-degree orderings, we compare total degree first. In case of the equality, there are many different orderings that can break the ties. The most commonly used are the graded lexicographic and the graded reverse lexicographic orderings.

Now we present our new result that establishes a sufficient condition for a finite set to be a Gröbner basis.

Proposition 3. *Let G be a finite subset of $K[\underline{x}]$ with D being the highest degree of its elements. Let $<$ be an order on $K[\underline{x}]$. Suppose that the following holds:*

1. *G contains all the terms of degree D as leading terms; and*
2. *if $H := G \cup \{t \cdot g \mid g \in G,\ t \text{ a term and } \deg(t \cdot g) \leq D + 1\}$, there exists $\widetilde{H}$, a row echelon form of H, such that $\widetilde{H}_{\leq D} = G$,*

then G is a Gröbner basis.

Note that condition 1 implies $\langle G \rangle$ is a zero-dimensional ideal. From now on we concentrate on zero-dimensional ideals.

Proof. Let $G = \{g_1, \ldots, g_s\}$ with $g_i \neq g_j$ for $i \neq j$. Suppose that the highest degree in G is D and that conditions 1 and 2 above hold. We want to show that for $i, j \in \{1, \ldots, s\}$, with $i \neq j$, $f := \mathrm{spol}(g_i, g_j)$ has a standard representation w.r.t. G. without loss of generality, it suffices to show it for $\mathrm{spol}(g_1, g_2)$.

If $d := \deg(\mathrm{lcm}(\mathrm{HT}(g_1), \mathrm{HT}(g_2))) \leq D + 1$, then by condition 2

$$f \in \mathrm{span}_K(H) = \mathrm{span}_K(\widetilde{H}) = \mathrm{span}_K(G) \oplus \mathrm{span}_K(\widetilde{H}_{=D+1}).$$

If $\deg(f) < D + 1$ then it is trivial to see that $f \in \mathrm{span}_K(G)$ and hence has a standard representation w.r.t. G. Suppose that $\deg(f) = D + 1$. By condition 1, every term of degree $D + 1$ appears as a head term in H. Choose $h_1 \in H$ such that $\mathrm{HT}(h_1) = \mathrm{HT}(f)$ and define f_1 by

$$f_1 := f - \frac{\mathrm{HC}(f)}{\mathrm{HC}(h_1)} h_1.$$

It is easy to see that $f_1 \in \mathrm{span}_K(\widetilde{H})$ and that $\mathrm{HT}(f_1) < \mathrm{HT}(f)$. If $\deg(f_1) = D + 1$ we can repeat the same argument for f_1 and by iterating the argument a finite number of times m, we obtain an expression

$$f = \sum_{i=1}^{m-1} a_i h_i + f_m \tag{1}$$

with $a_i \in K$, $h_i \in H$, for $1 \leq i < m-1$ $\mathrm{HT}(h_i) > \mathrm{HT}(h_{i+1})$ and $\deg(f_m) < D+1$. Since $f_m \in \mathrm{span}_K(\widetilde{H})$ and $\deg(f_m) < D + 1$, $f_m \in \mathrm{span}_K(G)$ thus clearly (1) yields a standard representation of f w.r.t G.

For $d > D + 1$, we proceed by induction. Suppose that $d > D + 1$ and that for $i \neq j$, if $\deg(\mathrm{lcm}(\mathrm{HT}(g_i), \mathrm{HT}(g_j))) < d$ then $\mathrm{spol}(g_i, g_j)$ has a standard representation w.r.t. G. Assume, without loss of generality, that $\deg(g_1) \geq \deg(g_2)$ and note that $\deg(g_1) > (D + 1)/2$. Let $t := \mathrm{lcm}(\mathrm{HT}(g_1), \mathrm{HT}(g_2))$ and let t_1, t_2 be terms such that for $i = 1, 2$, $t = t_i\,\mathrm{HT}(g_i)$. Note that $\deg(t_i) \geq 2$ and that t_1 and t_2 are disjoint. Choose any terms $t_{11}, t_{12}, t_{21}, t_{22}$ such that for $i = 1, 2$, $t_i = t_{i1}t_{i2}$ and $\deg(g_1) + \deg(t_{12}) = D + 1$ and $\deg(t_{21}) = 1$. These choices are possible because $(D + 1)/2 < \deg(g_1) \leq D$ thus $1 \leq D + 1 - \deg(g_1) < D + 1 - (D + 1)/2 = (D + 1)/2 < deg(g_1)$ and because $\deg(t_2) \geq 2$. It follows that $\deg(t_{11}), \deg(t_{12}), \deg(t_{21})$ and $\deg(t_{22})$ are all greater than or equal to 1. Also, if we let $t^* := \frac{t}{t_{11}t_{21}}$, by construction, for $i = 1, 2$, $\mathrm{lcm}(t^*, \mathrm{HT}(g_i))) = t/t_{i1}$ divides t properly, $\deg(t^*) = D$ and since t_1 and t_2 are disjoint, t^* is different from both $\mathrm{HT}(g_1)$ and $\mathrm{HT}(g_2)$. Then, by condition 1, there exist $g \in G \setminus \{g_1, g_2\}$ with $\mathrm{HT}(g) = t^*$. Also, for $i = 1, 2$, since $\deg(\mathrm{lcm}(\mathrm{HT}(g), \mathrm{HT}(g_i))) < \deg(t)$, by the inductive hypothesis, $\mathrm{spol}(g, g_i)$ has a standard representation w.r.t. G. Moreover, $\mathrm{HT}(g)$ divides t and therefore, by the Buchberger's second criterion, $\mathrm{spol}(g_1, g_2)$ has a standard representation w.r.t. G.

3 From XL to MXL$_2$

The MXL$_3$ algorithm adapts MXL$_2$ which in turn adapts XL [6]. Below we present a brief overview of the XL algorithm, the mutant strategy [8,9] and the MXL$_2$ improvements [16].

Let P be a finite set of polynomials in $K[\underline{x}]$. Given a degree bound D, the XL algorithm is simply based on extending the set of polynomials P by multiplying each polynomial in P by all the terms in T such that the resulting polynomials have degree less than or equal to D. Then, by using linear algebra, XL computes $\widetilde{P}$, a row echelon form of the extended set P. Afterwards, XL searches for univariate polynomials in $\widetilde{P}$.

In [8,9], it was pointed out that during the linear algebra step, certain polynomials of degrees lower than expected appear. These polynomials are called mutants. The mutant strategy aims at distinguishing mutants from the rest of polynomials and to give them a predominant role in the process of solving the system.

The precise definition of mutants is as follows.

Definition 3. *Let I be the ideal generated by the finite set of polynomials P. An element f in I can be written as*

$$f = \sum_{p \in P} f_p p \tag{2}$$

where $f_p \in K[\underline{x}]$. The maximum degree of $f_p p$, $p \in P$, is the level of this representation. The level of f is the minimum level of all of its representations. The polynomial f is called mutant with respect to P if $\deg(f)$ *is less than its level.*

The MutantXL algorithm [9] is a direct application of the mutant concepts to the XL algorithm. It was noted in [16] that there are two problems in the MutantXL algorithm that affect its performance. The first problem is, when the system generates a huge number of mutants. The second problem is, when the system generates an insufficient number of mutants to solve the system at a lower degree than XL. The MXL$_2$ algorithm handles the first problem by choosing the minimum number of mutants necessary to solve.

In [16], Mohamed et. al. also introduced a new technique for the space enlargement process to handle the second problem which is called the partial enlargement technique. In the process of space enlargement, MutantXL multiplies all the polynomials in P of degree D by all the terms of degree one such that each term is multiplied only once. In many cases, the last iteration of this process generates a very large number of dependent polynomials. These polynomials are reduced to zero. MXL$_2$ avoids this problem by using the partial enlargement technique. This means that, in the process of space enlargement MXL$_2$ multiplies only a subset of the polynomials of degree D in P and tries to solve. This step is repeated until the system is solved. The strategy for selecting a subset will be explained in the next section.

By these two improvements MXL_2 could outperform Magma's F_4 in terms of memory in all the cases of random systems and 57% of the cases of HFE systems that are considered in [16].

4 Description of the MXL_3 Algorithm

The main difference between MXL_2 and MXL_3 is that MXL_2 only works when the system of equations has a unique solution whereas MXL_3 can handle any system of equations with a finite number of solutions. Any XL-type algorithm eventually computes a Gröbner basis, however it is uncertain for which degree bound it occurs. Proposition 3 provides an easy to check condition that guarantees a Gröbner basis has been found. Experimental results show that in all the cases that we examined, using this alternative criterion reveals the Gröbner basis early. Another important difference is that MXL_3 multiplies only by some chosen monomials, while MXL_2 multiplies by all possible monomials. In addition to the notation of section 2, we also need the following notation.

4.1 Notation

Let $X := \{x_1, \ldots, x_n\}$ be a set of variables, upon which we impose the following order: $x_1 > x_2 > \ldots > x_n$. Let

$$R = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, ..., x_n^2 - x_n \rangle$$

be the Boolean polynomial ring in X with the monomials of R ordered by the graded lexicographical order $<_{glex}$. We consider elements of R as polynomials over $\mathbb{F}_2$ where degree of each term w.r.t any variable is 0 or 1. Let $P = (p_1, \ldots, p_m) \in R^m$ be an m-tuple of polynomials in R.

Throughout the operation of the algorithm described in this paper, a degree bound D will be used. This degree bound denotes the maximum degree of the polynomials contained in P. Also, we use ED as a degree bound for the eliminated subset of P, ($P_{\leq ED}$). Note that the content of P is changed throughout the operation of the algorithm. We define the leading variable $\mathrm{LV}(p)$ for $p \in P$ as the largest variable in $\mathrm{HT}(p)$, according to the order defined on the variables set. Also, we define the subset $\mathrm{LV}(P, x)$ as the set of all polynomials of P with leading variable x.

4.2 MXL_3 Algorithm

The algorithm performs the following steps:

- *Initialize*: Set $P = \{p_1, \ldots, p_m\}$, $D = \max\{\deg(p) : p \in P\}$, the elimination degree $ED = \min\{\deg(p) : p \in P\}$, the set of mutants $M = \emptyset$, the extension flag $newExtend = true$, and the partitioned variable $x = x_1$.

– *Repeat*
  Echelonize: Consider each term in $P_{\leq ED}$ as a new variable. Set $P_{\leq ED} = \widetilde{P}_{\leq ED}$, where $\widetilde{P}_{\leq ED}$ is the row echelon form of $P_{\leq ED}$.
  Here polynomials are identified with their coefficient vectors as explained in [11].
  ExtractMutants: Add all the new elements of $P_{<ED}$ to M.
  Gröbner: If ($ED < D$ or $newExtend = true$), $M_{<ED} = \emptyset$ and $|P_{=(ED-1)}| = |T_{=(ED-1)}|$ then set $G = P_{\leq(ED-1)}$, return G and terminate.
  Enlarge: If $M \neq \emptyset$, then **Multiply**(P, M, ED), otherwise **Extend**$(P, D, x, ED, newExtend)$.

Multiply(P, M, ED)

– Set $k = \min\{\deg(p)\colon p \in M\}$.
– Set $y = \max\{\mathrm{LV}(p) : p \in M_{=k}\}$.
– Select a necessary number of mutants of $M_{=k}$, multiply the selected mutants by all variables $\leq y$, remove the selected mutants from M, add the new polynomials to P.
  (The necessary number of mutants is numerically computed as in [16].)
– Set $ED = k + 1$.

Extend$(P, D, x, ED, newExtend)$

– If $newExtend = true$, then increment D by 1, set $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}\}$, and set $newExtend = false$. Otherwise, set $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}$ and $\mathrm{LV}(p) > x\}$ (the next-smallest leading variable).
– multiply all the polynomials of $\mathrm{LV}(P, x)$ (the current partition) by all the variables $\leq x$ without redundancy and add the newly obtained polynomials to P.
– If $x = x_1$, then set $newExtend = true$.
– Set $ED = D$.

We now explain the selection strategy that we use to avoid the redundancy produced from the extend step. During the multiplication process we keep the multiplier variable that gave rise to every new produced polynomial and we keep one for the original polynomials. When we extend the system, we multiply the polynomial p by all variables smaller than its previous multiplier variable. In case of the previous multiplier of p is one, we multiply by all variables. The target of this selection method is to speed up the extension process of the system. Only we multiply by monomials of degree one (variables) without any trivial redundancy.

Let for example $p \in P$, x_ip, x_jp be two polynomials in the extended system, and $x_i > x_j$. Then x_ip is extended by multiplying it with variables $< x_i$ one of them being x_jx_ip, while the redundant polynomial x_ix_jp can not be produced by x_jp since $x_i > x_j$ and x_jp is multiplied only by variables $< x_j$.

The multiplication of mutants process provides another important improvement to our algorithm. Let the system have mutants of degree $k < D$ and x is the greatest leading variable of the set of mutants. MXL$_3$ multiplies these mutants by all variables $\leq x$ instead of multiplying by all variables as in MXL$_2$. The target of this improvement is to solve with as small number of polynomials as we can.

Let for example M be a set of mutant polynomials of degree k and let x_i be the smallest leading variable of the elements in M. We multiply the elements of M by all variables $\leq x_i$. The resulting polynomials have smallest leading variable $\leq x_i$, then all the old polynomials of degree $k+1$ with leading variable $> x_i$ will not play any role during the Gaussian elimination process. This will decrease the dimension of the system.

The following theorem establishes the correctness of the algorithm.

Theorem 1. *The* MXL$_3$ *algorithm computes a Gröbner basis G of the ideal generated by the set $\{p_1, \ldots, p_m\}$ of R.*

Proof. Termination: MXL$_3$ terminates only when it enlarges all the polynomials of degree $< ED$ and when P contains all the terms of degree $ED - 1$ as leading terms, at a certain degree $ED \leq D$. The worst case is to satisfy these two conditions at $ED = D = n + 1$. Let the system is extended up to degree n without satisfying the termination conditions of the algorithm. In this case, MXL$_3$ extends the system to the next degree $D = n + 1$. P has only one polynomial of degree n. In the *Enlarge* step, this polynomial is extended, *newExtend* is set to *true*, and ED to $n + 1$. After the *Echelonize* step, P still contains only one polynomial of degree $n = ED - 1$ which is equal to the number of all terms in the Boolean ring R with degree n. If $M \neq \emptyset$, MXL$_3$ loops between *Eliminate* and *Enlarge* a finite number of times until the set M becomes empty. So all the conditions of *Gröbner* step are satisfied. Then MXL$_3$ returns $G = P$ with highest degree n and terminates.

Correctness: MXL$_3$ returns a set of polynomials G with elements of maximum degree $d = ED - 1$, where $ED \leq D$. The set G satisfies the first condition of Proposition 3 since It is in the row echelon form and It contains all the terms of degree d as leading terms. Also, the *Gröbner* step returns G only when *newExtend* $=$ *true* and $M_{<ED} = \emptyset$ which means that all the polynomials of degree $\leq d$ are enlarged. Then G satisfies the second condition of Proposition 3. Therefore G is a Gröbner basis for the ideal generated by the input system $\{p_1, \ldots, p_m\}$.

5 Experimental Results

We built our experiments to compare the efficiency of MXL$_3$ to the efficiency of F$_4$ in solving some random systems generated by Courtois [7] as well as some HFE systems generated by the code of John Baena. We run all the experiments on a Sun X4440 server, with four "Quad-Core AMD Opteron$^{\text{TM}}$ Processor 8356" CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz. We used only one out of the 16 cores.

Tables 1 and 2 show the results of dense random systems with many solutions and the results of HFE systems of univariate degree 288, respectively. In both

Table 1. Performance of MXL$_3$ versus F_4 for dense random system

	MXL$_3$				F_4			
n	D	max. matrix	Memory	Time	D	max. matrix	Memory	Time
25	6	66631×76414	698	704	6	248495×108746	5128	1341
26	6	88513×102246	1207	1429	6	298592×148804	8431	3325
27	6	123938×140344	2315	2853	6	354189×197902	13312	6431
28	6	201636×197051	4836	7982	6	420773×261160	20433	13810
29	6	279288×281192	9375	18796	6	499222×340254	30044	25631
30	6	332615×351537	15062	33331	6	1283869×374081	72258	92033
31	6	415654×436598	23078	94191	6	868614×489702	108738	162118

tables we denote the number of variables and equations by n and the highest degree of the iteration steps by D. The tables also show the maximum matrix size, the memory used in Megabytes, and the execution time in seconds. It is evident from Tables 1 and 2 that MXL$_3$ solves the random generated systems and HFE systems faster and consumes less memory than F$_4$.

Table 1 shows that both MXL$_3$ and F$_4$ solve random systems up to a system of 31 variables. The solutions of MXL$_3$ are consistent to the results of Magma. When MXL$_3$ and F$_4$ tried to solve a 32 variables system, both were able to enlarge the system up to degree 6. When the system was enlarged to degree 7, they ran out of memory. For the 30 variables system we get a strange matrix size from Magma. We created many 30 variables random system and we obtain approximately the same numbers.

Table 2. Performance of MXL$_3$ versus F_4 for HFE(288,n) systems

	MXL$_3$				F_4			
n	D	max. matrix	Memory	Time	D	max. matrix	Memory	Time
30	5	86795×130211	1389	3106	5	149532×136004	7105	3806
35	5	155914×296872	5737	10047	5	200302×321883	40480	11032
36	5	173439×344968	7310	14183	5	219438×382252	50846	15220
37	5	192805×399151	9288	20375	5	247387×444867	66623	20787
38	5	212271×459985	11351	27089	5	274985×512311	83445	27305
39	5	234111×528068	15070	36833	5	305528×588400	104135	38013
40	5	258029×604033	20881	63460≃17.6 hours	ran out of memory			
45	5	404940×1126819	55216	299355≃3.46 days	ran out of memory			
47	5	457691×1417468	77967	371088≃4.3 days	ran out of memory			
48	5	517642×1583807	98913	689235≃7.9 days	ran out of memory			
49	5	561972×1765465	120524	751965≃8.7 days	ran out of memory			

Table 2 shows that all the HFE systems of univariate degree 288 up to 49 variables are solved by using MXL$_3$, whereas F$_4$ could only solve HFE systems up to 39 variables with the same memory resources.

In Table 3 we compare the performance of the MXL$_3$ algorithm against the F$_4$ algorithm in computing a Gröbner basis of the random system $n = 30$. For

MXL$_3$, we give the elimination degree (D), the matrix size for each level, the rank of the matrix (Rank), the number of mutants found (NM), the number of used mutants (UM), and the lowest degree of mutants found (MD). For F$_4$, we give the step degree (D), the matrix size, and the step memory in MB.

Table 3. Results for the system Random-30

	MXL$_3$						F_4		
Step	D	Matrix Size	Rank	NM	UM	MD	D	Matrix Size	Memory
1	2	30×466	30	0	0	-	2	30×466	14.2
2	3	930×4526	930	0	0	-	3	937×4526	14.2
3	4	13980×31931	13515	0	0	-	4	13320×30551	207
4	5	131690×174437	121365	0	0	-	5	106603×143547	4318
5	6	**332615×351537**	329051	31060	665	5	6	588160×437262	42843
6	6	302981×309033	302981	3596,12340	0,191	5,4	6	**1283869×374081**	72258
7	5	172945×174437	172945	2480,3160,90	0,0,11	4,3,2	2	722×466	72258
8	3	4510×4526	4510	315,15	0,1	2,1	3	4864×3782	72258
9	2	480×466	465	15	0	1	4	22421×19736	72258
10							5	103919×62858	72258

Table 3 shows that by using the mutant strategy, MXL$_3$ can easily solve the 30 variables random system with a smaller matrix size compared to F$_4$. MXL$_3$ starts to generate mutants at step 5. In this step 31060 mutants of degree 5 are generated, out of which only 665 are multiplied. Due to the degree of the generated mutants, the elimination degree remains the same in the next step, i.e. , $D = 6$. Starting from step 7, D starts to decrease. In step 8, the system generates 315 quadratic mutants and 15 linear mutants. By using only one of the linear mutants, MXL$_3$ generates additional 15 linear mutants in the next step, which in turn leads to solving the system.

Also, Table 3 shows that the number of reductions to zero is less than 8% for each iteration step. This explains practically that our improved selection strategy has strictly increased the efficiency of the algorithm since it avoids the redundant computations.

In Appendix A Table 4 presents a comparison between the maximum matrix size constructed by MXL$_3$ and MXL$_2$ on some random systems that have only one solution. The results show that MXL$_3$ solves with smaller number of polynomial equations and smaller number of terms than MXL$_2$. This due to the selection strategy of the multiplied variables that used is by MXL$_3$. In Appendix B Table 5 presents another comparison between MXL$_3$ and Magma's F$_4$ when the HFE parameter is setting to true. In this case MXL$_3$ also solves with smaller number of polynomials than Magma's F$_4$. For example the HFE challenge 1 $n = 80$ that was first solved with maximum matrix size 307126×1667009 by Faugère and Joux [14] using F$_5$ /2 algorithm [12] in May 2002, can be solved by MXL$_3$ with maximum matrix size 268840×1666981, while Magma solves it with maximum matrix size 293287×1666981. In this case Magma is faster than our implementation since it uses a very fast linear algebra implementation.

For the comparison with Faugère's F$_4$ algorithm, we used Magma (version V2.13-10). We used the field equations $x_i^2 = x_i$ and using the polynomial ring type for Magma that is defined over $\mathbb{F}_2$ such that all the terms are reduced modulo the field equations. When we use the new version of Magma (V2.15) and the new Magma type BooleanPolynomialRing, we have worse results in terms of the matrix size and the memory, although we obtained better results in terms of the running times. For MXL$_3$, we also used the Boolean polynomial ring in our C++ implementation. For the *Echelonize* step, we used an adapted version of M4RI [1], a library for dense matrix linear algebra over $\mathbb{F}_2$. Our adaptation is in changing the strategy of selecting a pivot during Gaussian elimination to keep the old elements in the system intact.

We chose to compare MXL$_3$ only with Magma's implementation of F$_4$ because both Faugère's algorithm and Steel's implementation are widely recognized as benchmarks for efficient Gröbner basis computation. A comparison with Faugère's F$_5$ algorithm has been suggested by some researchers. We consider this infeasible due to the controversy about the algorithm and the lack of a well recognized implementation.

F$_5$ is primary intended for homogeneous regular sequences There is no detailed description of how to adapt it for non-homogeneous sequences and moreover, no analysis exist on the behavior for such sequences. In the non-homogeneous case the F$_5$ criteria can discover trivial syzygies of the leading forms but it says nothing about what to do with the non-trivial ones which in turn yield mutants. In [9,16,10] the importance of mutants in the computation of Gröbner Bases for non-homogeneous sequences has been shown. F$_5$ says nothing about how to take advantage of those. Therefore, in the absence of such algorithm one relies on Buchberger's or the F$_4$ algorithm to compute a Gröbner Basis once mutants appear.

6 Conclusion and Future Work

In this paper, the MXL$_3$ algorithm is introduced as a new and efficient method to compute Gröbner bases on the Boolean polynomial ring. The experiments showed that both in classical cryptographic challenges and random systems, this new algorithm performs better in terms of memory than the F$_4$ algorithm implemented in Magma, currently the best publicly available implementation of F$_4$. The growth of the complexity shown in the experiments suggests that the difference is not marginal.

These experimental results demonstrate the importance of mutants in the computation of Gröbner bases, which was explored in a different setting in [10]. In combination with the techniques derived from the concepts of necessary number of mutants and partial enlargement, this new strategy has shown to be very successful unfolding the underlying structure of systems of equations.

Also, the new criterion for determining the termination of the new algorithm, proved to be efficiently checkable and sharp to detect a Gröbner basis. The fact that the MXL$_3$ algorithm terminates at the same degree as the F$_4$ algorithm in all experiments, suggests a connection between this criterion and other criteria to establish a Gröbner basis, a very interesting new direction, we will study next.

This paper further demonstrates the great potential of the mutant strategy and much more is still needed to be done to realize its full potential. Comparison with PolyBoRi [3] are planed as well as some improvements in the enlargement technique.

Acknowledgments

We would like to thank partial support from NSF China (No.6073131) and Taft foundation. Also, we would like to thank John Baena for supporting us with his code of generating HFE systems.

References

1. Albrecht, M., Bard, G.: M4RI Linear Algebra over GF(2) (2008), `http://m4ri.sagemath.org/index.html`
2. Becker, T., Kredel, H., Weispfenning, V.: Gröbner bases: a computational approach to commutative algebra, April 1993. Springer, London (1993)
3. Brickenstein, M., Dreyer, A.: Polybori: A framework for gröbner-basis computations with boolean polynomials. Journal of Symbolic Computation 44(9), 1326–1345 (2009); Effective Methods in Algebraic Geometry
4. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal). PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965 (English translation in Journal of Symbolic Computation (2004)
5. Buchberger, B.: A criterion for detecting unnecessary reductions in the construction of gröbner bases. Johannes Kepler University Linz, London, UK, vol. 72, pp. 3–21. Springer, Heidelberg (1979)
6. 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)
7. Courtois, N.T.: Experimental Algebraic Cryptanalysis of Block Ciphers (2007), `http://www.cryptosystem.net/aes/toyciphers.html`
8. Ding, J.: Mutants and its impact on polynomial solving strategies and algorithms. Privately distributed research note, University of Cincinnati and Technical University of Darmstadt (2006)
9. Ding, J., Buchmann, J., Mohamed, M.S.E., Moahmed, W.S.A., Weinmann, R.-P.: MutantXL. In: Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China, April 2008, pp. 16–22. LMIB (2008)
10. Ding, J., Carbarcas, D., Schmidt, D., Buchmann, J., Tohaneanu, S.: Mutant Gröbner Basis Algorithm. In: Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China, April 2008, pp. 23–32. LMIB (2008)
11. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F4). Pure and Applied Algebra 139(1-3), 61–88 (1999)

12. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 international symposium on Symbolic and algebraic computation (ISSAC), Lille, France, July 2002, pp. 75–83. ACM, New York (2002)
13. Faugère, J.-C., Ars, G.: Comparison of XL and Gröbner basis algorithms over Finite Fields. Research Report RR-5251, Institut National de Recherche en Informatique et en Automatique, INRIA (2004)
14. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
15. Mohamed, M.S.E., Ding, J., Buchmann, J., Werner, F.: Algebraic Attack on the MQQ Public Key Cryptosystem. In: Proceedings of the 8th International Conference on Cryptology And Network Security (CANS 2009), Kanazawa, Ishikawa, Japan, December 2009. LNCS, Springer, Heidelberg (to appear, 2009)
16. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: MXL2: Solving Polynomial Equations over GF(2) using an Improved Mutant Strategy. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)
17. Sugita, M., Kawazoe, M., Imai, H.: Relation between the XL Algorithm and Gröbner Basis Algorithms. Transactions on Fundamentals of Electronics, Communications and Computer Sciences (IEICE) E89-A(1), 11–18 (2006)

Appendix A

Table 4. Performance of MXL_3 versus MXL_2 for dense random system

	MXL_3	MXL_2
n	max. matrix	max. matrix
15	1422×1577	1946×1758
16	2295×2573	2840×2861
17	3211×3676	3740×4184
18	4477×5335	6508×7043
19	8150×8039	9185×11212
20	8494×10564	14302×12384
21	16128×16115	14365×20945
22	20332×20737	35463×25342
23	23415×26407	39263×36343
24	52215×57171	75825×69708

Appendix B

Table 5. Performance of MXL_3 versus F_4 for HFE(96,n) systems

	MXL_3	F_4
n	max. matrix	max. matrix
20	5236×5227	7053×6196
25	9979×9941	12459×15276
30	22515×31931	20003×31931
35	33705×59536	30081×59536
40	37005×84516	43124×102091
50	67525×251176	79116×251176
60	144030×523686	130755×523686
70	181335×974121	201343×974121
80	268840×1666981	293287×1666981