Flexible Partial Enlargement to Accelerate Gröbner Basis Computation over $\mathbb{F}_2$

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

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

Abstract. Recent developments in multivariate polynomial solving algorithms have made algebraic cryptanalysis a plausible threat to many cryptosystems. However, theoretical complexity estimates have shown this kind of attack unfeasible for most realistic applications. In this paper we present a strategy for computing Gröbner basis that challenges those complexity estimates. It uses a flexible partial enlargement technique together with reduced row echelon forms to generate lower degree elements–mutants. This new strategy surpasses old boundaries and obligates us to think of new paradigms for estimating complexity of Gröbner basis computation. The new proposed algorithm computed a Gröbner basis of a degree 2 random system with 32 variables and 32 equations using 30 GB which was never done before by any known Gröbner bases solver.

Keywords: Algebraic cryptanalysis, Gröbner basis, Complexity, HFE.

1 Introduction

The standard way to represent the polynomial ideal is to compute a Gröbner basis of it. Solving a system of multivariate polynomial equations is one of the most important application of Gröbner bases. The complexity of algorithms for computing a Gröbner basis of an ideal depends on how large the subset of elements of the ideal used during the computation is. Minimizing the size of this subset is an extremely important target for research in this area.

Suppose P is a finite set of degree 2 polynomials. For $d \geq 2$ consider the set

$$H_d := \{t \cdot p \mid t \text{ is a term}, p \in P, \deg(tp) \leq d\} \tag{1}$$

It is well known that for d large enough, a row echelon form of H_d is a Gröbner basis for $\langle P \rangle$ (see Section 1.1 for a definition of row echelon form and other

⋆ Corresponding author.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 69–81, 2010.

notations used in this introduction). However, systems that appear in algebraic cryptanalysis and their corresponding degrees needed to solve them are so large, that constructing H_d is unfeasible. Usually memory is the main constraint. A key strategy we have been and are following is to explore and do computation in H_d for large enough d, but without storing the whole set H_d at any given time.

A first breakthrough came with the concept of mutant polynomials [4,5]. Informally, a polynomial p is called mutant, if $p \in \text{span}(H_d) \setminus \text{span}(H_{d-1})$ but its degree is strictly less than d. If x is a variable, then the polynomial xp belongs to $\text{span}(H_{d+1})$, however, since its degree is less than $d+1$ we can reduce it using only elements from H_d.

A second breakthrough came with the partial enlargement technique [9]. By partitioning the set $X \times H_d$ according to leading variables, it is possible to explore H_{d+1} one step at a time without being forced to store the whole set at once. In [8] the MXL_3 algorithm was introduced which uses an optimized mutant strategy, the partial enlargement technique and a mutant based termination criterion.

In this paper we introduce an efficient algorithm, called Mutant-based Gröbner Basis algorithm (MGB), that uses a more flexible partial enlargement to omit some parts of H_d and still satisfies MXL_3's criterion. Similar heuristics were attempted by Gotaishi and Tsujii in [7]. The MGB algorithm is fully implemented in C++. We give experimental results that compare the performance of MGB with both MXL_3 and the Magma's implementation of F_4 algorithm [6]. Our experiments are based on randomly generated MQ systems and HFE cryptosystems. We show that MGB computed a Gröbner basis of a degree 2 random system with 32 variables and equations in 2.3 days using only 30 Gigabytes.

This paper is organized as follows. In Section 2 we discuss the theoretical foundation of the new proposed algorithm. In Section 3 we present the MGB algorithm and exemplify its operation in Section 4. A complexity analysis of the algorithm is discussed in Section 5 and experimental results are presented in Section 6. Finally, we conclude the paper and give the future work in Section 7. Before continuing let us introduce the necessary notation.

1.1 Notation

Let $X := \{x_1, \ldots, x_n\}$ be a set of variables, upon which we impose the following order: $x_1 > x_2 > \ldots > x_n$. (Note the counterintuitive $i < j$ imply $x_i > x_j$.) 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 terms of R ordered by the graded lexicographical order $<_{glex}$. We represent an element of R by its minimal representative polynomial over $\mathbb{F}_2$ where degree of each term w.r.t any variable is 0 or 1. We denote by $T_d(x_{j_1}, \ldots, x_{j_s})$ the set of terms of degree d in the variables $x_{j_1}, \ldots, x_{j_s}$, and by T_d all the terms of degree d.

Let $P = \{p_1, \ldots, p_m\}$ be set of polynomials in R. A row echelon form is simply a basis for $\text{span}(P)$ with pairwise distinct head terms, (see [8] for definition). We will denote by $P_{(op)d}$ the subset of all the polynomials of degree $(op)d$ in P, where

(op) is any of $\{=, <, >, \leq, \geq\}$. We define the leading variable of $p \in P$, denoted by $\mathrm{LV}(p)$, as the largest variable in $\mathrm{HT}(p)$ according to the order defined on the variables set.

Suppose that all polynomials in P are of degree d. We define P_{x_j} as the set of all polynomials in P with leading variable x_j, so that P is partitioned in n sets $P_{x_1}, \ldots, P_{x_n}$. We refer to these sets as *variable-partitions*. Given a pair (x, p) in the Cartesian product $X \times P$, we define its leading variable to be $\mathrm{LV}(x, p) := \max(x, \mathrm{LV}(p))$ (note that in general $\mathrm{LV}(x, p) \neq \mathrm{LV}(xp)$). Consider also the partition of the Cartesian product $X \times P$ in n sets, $L_1(P), \ldots, L_n(P)$ defined by

$$L_j(P) := \{(x, p) \in X \times P \mid \mathrm{LV}(x, p) = x_j\} , \tag{2}$$

and note that that for $(x, p) \in L_j(P)$, $\mathrm{LV}(xp) \leq x_j$ or $\deg(xp) < d + 1$.

2 A More Flexible Partial Enlargement

The partial enlargement technique introduced first in [9] is effective in reducing the number of polynomials needed at the highest degree D where mutants start to appear. However, it offers no advantage over generating the whole set H_d for $d < D$ (as defined in equation (1)). We propose a method for systematically omitting subsets of H_d whenever a given condition is met. In this section we provide the theoretical foundation for this method.

The partial enlargement technique introduced in [9] can be described as follows. Let P_2 be a finite set of degree 2 polynomials in R and assume P_2 is in row echelon form. Initialize the set G with P_2. Then, enlarge P_2 one variable at a time, i.e., for $j = n, n-1, \ldots, 1$ (in that order), construct the set $L_j(P_2)$, then append the set $\{xp \mid (x, p) \in L_j(P_2)\}$ to G and compute a row echelon form of G. If no mutants are found in G, make P_3 the set of new polynomials in G and repeat the process for P_3 only.

While no mutants are produced, we produce a sequence of sets $P_2, P_3, \ldots$ such that for $p \in P_d$, $\deg(p) = d$ and $\mathrm{span}(P_d) = \mathrm{span}(H_d)$. We have observed, that often, the largest variable-partitions of each P_d are *full*, meaning that they have as many linearly independent polynomials as they can possibly have. So, there exist $J_d < n$ such that for all term $t \in T_d$ such that $\mathrm{LV}(t) \geq x_{J_d}$, there exist $p \in P_d$ such that $t = \mathrm{HT}(p)$. Note that if the j partition of P_d is full, necessarily the j partition for P_{d+1} is also full. Hence, the sequence $J_2, J_3, \ldots$ is non-decreasing.

If the j partition of P_d is full, we propose to omit the j partition from any subsequent P_D $(D > d)$, under the certainty that it will be full. The problem is that when mutants appear, it is possible that the head term of a mutant happens to be precisely one of the omitted head terms, thus failing to be completely reduced, in the sense that its head term is already in the ideal generated by the head terms of the elements in G.

To remedy this situation, we compute row reduced echelon forms instead of just any row echelon form, by means of which, we are able to perform reductions

in advance, and ensure polynomials are fully reduced with respect to the omitted partitions. The following proposition states this result precisely.

Proposition 1. *Let P_d be a finite subset of degree d polynomials from $K[\underline{x}]$ in row-echelon form. Suppose there exist $J < n$ such that for all term $t \in T_d$ such that $\mathrm{LV}(t) \geq x_J$, there exist $p \in P_d$ such that $t = \mathrm{HT}(p)$. Let*

$$G := P_d \cup \bigcup_{j=J+1}^{n} \{xp \mid (x,p) \in L_j(P_d)\} ,$$

$\widetilde{G}$ the row reduced echelon form of G and

$$P_{d+1} := \{p \in \widetilde{G} \mid \deg(p) = d+1\} .$$

Then for $j > J$, $(x,p) \in L_j(P_{d+1})$ and any term $s \in T_{d+1}$ such that $\mathrm{LV}(s) \geq x_J$, the term s is not a term in xp.

Proof. Let $j > J$, $(x,p) \in L_j(P_{d+1})$ and $s \in T_{d+1}$ such that $\mathrm{LV}(s) \geq x_J$. From the definitions of G and P_{d+1}, all terms of degree $d+1$ in p belong to $T_{d+1}(x_{J+1}, \ldots, x_n)$ thus s is not a term in p.

Since P_{d+1} is a subset of $\widetilde{G}$, the row reduced echelon form of G, and all terms of degree d with leading variables $\geq x_J$ appear as head term of an element of $P_d \subset G$, then such terms do not appear in p, i.e. all terms of degree d in p are elements of $T_d(x_{J+1}, \ldots, x_n)$. Moreover, $x < x_J$ since $j > J$, hence, there is no term t in p with degree d that satisfies $xt = s$. Therefore, s is not a term in xp.

The importance of this proposition is that the polynomials of degree $d+1$ with leading variables $x_1, \ldots, x_J$ are not needed to reduce polynomials coming from $L_j(P_{d+1})$ for $j > J$. Hence, when enlarging P we may omit the polynomials coming from $L_1(P), \ldots, L_J(P)$. This does not imply that this polynomials are not needed ever again, but just that their computation can be postponed yet obtaining sets of polynomials row reduced with respect to this missing polynomials. This observation leads to the heuristic strategy described in the following section. Section 4 illustrates the operation of the algorithm.

3 The MGB Algorithm

In this section we introduce the MGB algorithm to compute Gröbner bases over the function ring R under the graded lexicographic order. The MGB is based on the MXL3 algorithm [8], and differs in a heuristic method to omit some variable-partitions based on the flexible partial enlargement explained in Section 2. The heuristic consists in enlarging up to the first full partition. By this way many partitions are omitted, yet enough polynomials are enlarged.

Algorithms 1, 2, 3 and 4 provide a detailed description of the MGB algorithm. The most important difference with MXL_3 is the extra condition

$$|P_{=D-1}(x_1, \ldots, x)| = |T_{=D-1}(x_1, \ldots, x)|$$

in line 20 of the *Enlarge* subroutine (Algorithm 4). It means that all the terms of degree $D-1$ with leading variable $\in x_1, \ldots, x$ appear as leading terms in P. When the condition is met, the flag *newExtend* is set to true, which forces D to be increased in the next call to *Enlarge*.

Throughout the operation of the algorithm a degree bound D is used. This degree bound denotes the maximum degree of the polynomials contained in P. An elimination degree bound ED is used as a bound for the degrees of polynomials in P that are being eliminated. RREF(P, ED) means the reduced row echelon form of $P_{\leq ED}$. We mean by the new elements the set of all polynomials produced during the previous enlargement. The set M stores the mutants obtained during the *Echelonization* process. We define the array S to keep the the largest leading variable at each degree level. Note that the content of P is changed throughout the operation of the algorithm.

Algorithm 1. Algorithm1(P)

Require: P is a finite sequence from R
1: $D = \max\{\deg(p) \mid p \in P\}$
2: $ED = D$
3: $M = \emptyset$
4: $S = \{s_i = x_1 : 1 \leq i \leq D\}$
5: $x = x_1$
6: $newExtend = true$
7: **loop**
8: *Echelonize*(P, M, ED)
9: $G =$*Gröbner*(P, M, D, ED)
10: **if** $G \neq \emptyset$ **then**
11: **return** G
12: **end if**
13: *Enlarge*$(P, M, S, x, D, ED, newExtend)$
14: **end loop**

Algorithm 2. Echelonize(P, M, ED)

1: Consider each term in P as a new variable
2: $P =$ RREF(P, ED)
3: $M = \{p \in P : P$ is a new element and $\deg(p) < ED\ \}$

On an actual implementation we do not compute the full row reduced echelon form of the constructed matrix in each step. For the columns corresponding to highest degree terms we only clear entries under the pivot. For the rest of the columns we clear above and below the pivot. The idea of using the full reduction in this case is creating the vertical hole of the unextended partitions as explained in the previous section and illustrated in the next section.

Algorithm 3. Gröbner(P, M, S, D, ED)

1: $G = \emptyset$
2: **if** $M_{<ED} = \emptyset$ **and** $(ED < D$ **or** $newExtend = true)$ **then**
3: $s = S[ED - 1]$
4: **if** $|P_{=ED-1}| = |T_{=ED-1}(s, \ldots, x_n)|$ **then**
5: **if** $s < x_1$ **then**
6: Recover(P, ED, S)
7: **end if**
8: $G = P_{<ED}$
9: **end if**
10: **end if**
11: **return** G

Algorithm 4. Enlarge$(P, M, S, x, D, ED, newExtend)$

1: **if** $M \neq \emptyset$ **then**
2: $k = \min\{\deg(p)\colon p \in M\}$
3: Select a necessary number of mutants NM from $M_{=k}$
4: $y = \max\{\mathrm{LV}(p) : p \in NM\}$
5: Multiply selected mutants by all variables $\leq y$
6: Remove the selected mutants from M
7: Add the new polynomials to P
8: $ED = k + 1$
9: **else**
10: **if** $newExtend$ **then**
11: $D = D + 1$
12: $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}\}$
13: $newExtend = false$
14: **else**
15: $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}$ and $\mathrm{LV}(p) > x\}$
16: **end if**
17: $S[D] = x$
18: Multiply all $p \in P_{D-1}$ that has leading variable x by all the variables $\leq x$ without redundancy
19: Add the newly obtained polynomials to P
20: **if** $|P_{=D-1}(x_1, \ldots, x)| = |T_{=D-1}(x_1, \ldots, x)|$ **or** $(x = x_1)$ **then**
21: $newExtend = true$
22: **end if**
23: Set $ED = D$
24: **end if**

The *Recover* function in line 6 of Algorithm 3 enlarges all partitions of $P_{\leq Ed}$ that were omitting during the enlargement process and echelonizes P. Also, it multiplies any mutants found.

The correctness of the MGB algorithm is guaranteed by the following proposition, first stated and proved in [8].

Proposition 2. *Let G be a finite subset of R with D being the highest degree of its elements. G is a Gröbner basis if it satisfies the following conditions:*

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

The MGB algorithm terminates only when it returns the set $G = P_{<ED}$ that is computed by the *Gröbner* subroutine. We are going now to prove that G is a Gröbner basis of the ideal generated by P.

The first if statement of Algorithm 3, line 2, guarantees the second condition of Proposition 2 since all the polynomials up to degree $ED - 1$ are extended one degree more without producing any mutants ($M_{<ED} = \emptyset$). The second if statement of Algorithm 3, line 4, represents a second difference from MXL_3. It means that all the terms of leading variable $\in s, \ldots, x_n$ appear as leading terms in $P_{=Ed-1}$. This will guarantee the first condition of Proposition 2 as follows.

In case of $s = x_1$, $P_{<ED}$ contains all terms of degree $ED-1$ as leading terms. In case of $s < x_1$, MGB needs to recover $P_{\leq ED}$ as explained above. This leads to satisfying the two conditions of Proposition 2 since these unextended partitions have full rank.

Therefore MGB returns the set $G = P_{<ED}$ that satisfies the two conditions of Proposition 2. Then it is a Gröbner basis. The worst case of MGB is to reproduce the MXL_3 algorithm. So MGB terminates since MXL_3 terminates, theorem 1 in [8]. As an important note, for the experiments run so far, the recovering process was never necessary.

4 The Algorithm in Action

We describe the behavior of the algorithm in a concrete example, a sequence of 24 degree 2 polynomials in 24 variables. We refer to Figure 1 for a schematic representation of the process.

After *Echelonization*, the leading variables of the original polynomials range from x_1 to x_2 as depicted at the bottom of Figure 1. Then, the x_1 and x_2 partitions are enlarged and echelonized to obtain degree 3 polynomials with leading variables ranging from x_1 to x_3. Here we encounter that the variable partitions for x_1 and x_2 are full and we represent it with the darker shading in Figure 1.

So in the next step, only the x_2 and x_3 partitions are enlarged to degree 4. After *Echelonization* we obtain polynomials of degree 4 with leading variables ranging from x_2 to x_5. Note that since the two partitions x_1 and x_2 of degree 3 polynomials are full, no term with leading variable x_1 or x_2 of degree 3 appears in the degree 4 polynomials. This is depicted in Figure 1 with vertical stripes.

At degree 4, the variable partitions corresponding to x_2, x_3 and x_4 are full, so only the x_4 and x_5 partitions are enlarged to degree 5. After *Echelonization*, we obtain polynomials of degree 5 with leading variables ranging from x_4 to x_{12}. Note that by Proposition 1, no term with leading variable x_1 of degree 4 appears in the degree 5 polynomials, and *Echelonization* clears x_2 through x_4.

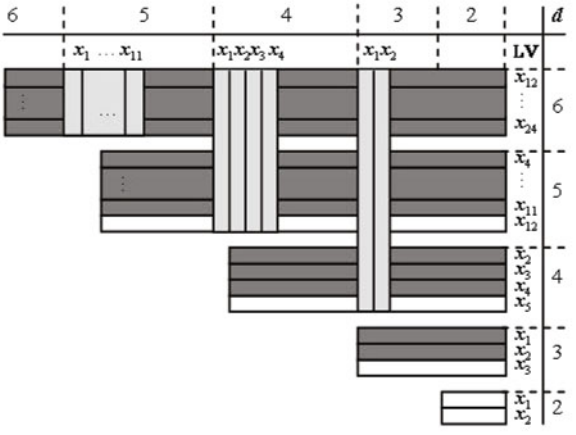

Fig. 1. Behavior of the algorithm for a sequence of 24 degree 2 equations in 24 variables. Horizontal stripes represent variable-partitions, darker ones are full. Vertical stripes represent terms that do not appear in the given polynomials.

At degree 5, the variable partitions from x_4 to x_{11} are full, so only the x_{11} and x_{12} partitions are enlarged to degree 6. In fact, once the x_{12} partition is enlarged and echelonized, mutants are produced and after a few steps of enlarging mutants and *echelonization*, we arrive at a situation in which the MXL_3 criterion is satisfied and the algorithm terminates.

5 Complexity Analysis

Studies of the complexity of Gröbner basis computation have mostly focused on the maximum degree of the polynomials that occur during computation. For example, [2] provides an exact formula for computing the degree of regularity (D_{reg}) of a homogeneous semi-regular sequence. In the case of a somehow generic sequence, this degree coincides with the maximum degree of the polynomials that occur during computation using the F_5 algorithm. If the system is homogeneous, they argue that linear algebra over a square matrix of size the number of terms of degree D_{reg} would solve the system and use this as an upper bound. This line of argument is sound in the case of a homogeneous semi-regular sequence.

For non-homogeneous semi-regular sequences we should add the lower degree terms to obtain

$$\sum_{d=0}^{D_{reg}} \binom{n}{d}$$

The algorithm proposed in this paper puts in question the sharpness of this bound. For example, in Section 4 we described the behavior of the algorithm for a random system of 24 degree 2 polynomials in 24 variables, in which the size of the largest matrix produced by the new algorithm was $26\,409 \times 33\,245$. This contrasts with

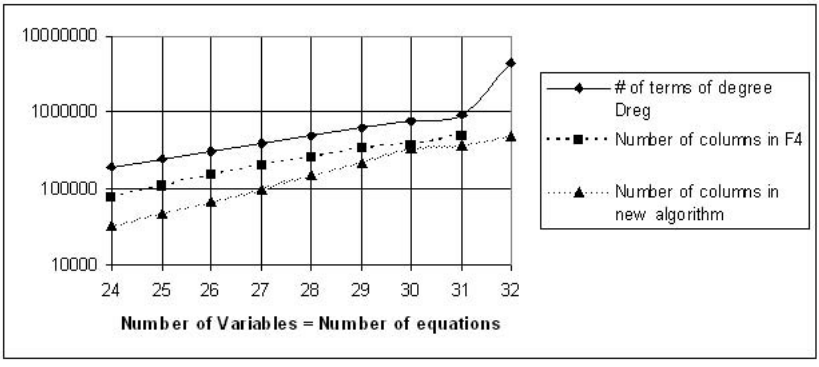

Fig. 2. Experimental results compared with number of terms at the degree of regularity

the number of terms up to degree 6 in 24 variables which is 190 051. Such small size was achieved because some partitions were omitted at different degrees and as a consequence some terms never appeared in the computation.

The complexity of the algorithm presented in this paper depends not only on the highest degree D of the system and the number of variables n, but also on the sequence $n_1, n_2, \ldots, n_D$ of variables omitted at each degree d. The number of columns of the largest matrix is given by

$$\sum_{d=1}^{D} \binom{n - n_d}{d} + 1$$

Figure 2 compares matrix size estimated solely based on degree of regularity with experimental results. It shows a significant gap between the number of terms up to degree D_{reg} and the number of columns of the new algorithm and even that of the F_4 algorithm (for a more complete report on the results see Section 6).

Although we don't have a way to predict or even estimate the sequence $n_1, n_2, \ldots, n_D$, It is very clear in this case that omitting substantial number of partitions could drastically change the complexity to solve the corresponding system, however it remains very speculative how this will really work and we will explore this case in a subsequent paper.

6 Experimental Results

We present our experiments to compare the efficiency of MGB with both MXL_3 and F_4 algorithms. We tested them with random systems generated by Courtois [3] and HFE systems generated by the code of John Baena. We run all the experiments in a Sun X4440 server, with four "Quad-Core AMD Opteron^{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 main experiments of dense random systems with many solutions and the experiments of HFE systems of univariate degree 288,

Table 1. Experiments for dense random systems

	F_4		MXL_3		MGB	
n	D	max. matrix	D	max. matrix	D	max. matrix
24	6	207150×78637	6	50367×57171	6	26409×33245
25	6	248495×108746	6	66631×76414	6	37880×47594
26	6	298592×148804	6	88513×102246	6	55063×67815
27	6	354189×197902	6	123938×140344	6	92296×99518
28	6	420773×261160	6	201636×197051	6	132918×148976
29	6	499222×340254	6	279288×281192	6	173300×224941
30	6	1283869×374081	6	332615×351537	6	265298×339236
31	6	868614×489702	6	415654×436598	6	349778×381382
32	ran out of memory		ran out of memory		7	437172×507294

Table 2. Experiments for HFE(288,n) systems

	F_4		MXL_3		MGB	
n	D	max. matrix	D	max. matrix	D	max. matrix
30	5	149532×136004	5	86795×130211	5	68468×109007
35	5	200302×321883	5	155914×296872	5	116737×254928
36	5	219438×382252	5	173439×344968	5	125133×297503
37	5	247387×444867	5	192805×399151	5	142460×345635
38	5	274985×512311	5	212271×459985	5	153181×399855
39	5	305528×588400	5	234111×528068	5	171985×460727
40	ran out of memory		5	258029×604033	5	192506×528849
49	ran out of memory		5	561972×1765465	5	371368×1584984
50	ran out of memory		ran out of memory		5	382392×1766691
51	ran out of memory		ran out of memory		5	410169×1964756

respectively. We denote the number of variables and equations by n and the highest degree of the iteration steps by D. We also show the maximum matrix size. It is evident from Table 1 and 2 how the new strategy improves MXL_3.

Figure 3 displays a comparison between MGB, MXL_3 and F_4 in terms of space and time. It is clear from Figure 3(a) that the MGB algorithm uses less memory than both MXL_3 and Magma's F_4 since it constructs smallest matrices. However, it is not much faster than MXL_3 in terms of the size of the system becomes bigger as shown in Figure 3(b). The reason is that the new algorithm uses a row-reduced echelon form while MXL_3 uses only the row echelon form. Also, the gap between MGB and MXL_3 becomes a little smaller as the size of the system increased.

Table 3 shows the detailed result of computing a Gröbner basis of a dense random system with 32 variables by MGB. For each step we give the degree (D), the matrix size, the rank of the matrix (Rank), a set of leading variable of the level partitions, the number of variables in the degree D terms (n_D), the number of lowest degree mutants found (NM), the number of used mutants (UM), and finally the lowest degree of mutants found (MD).

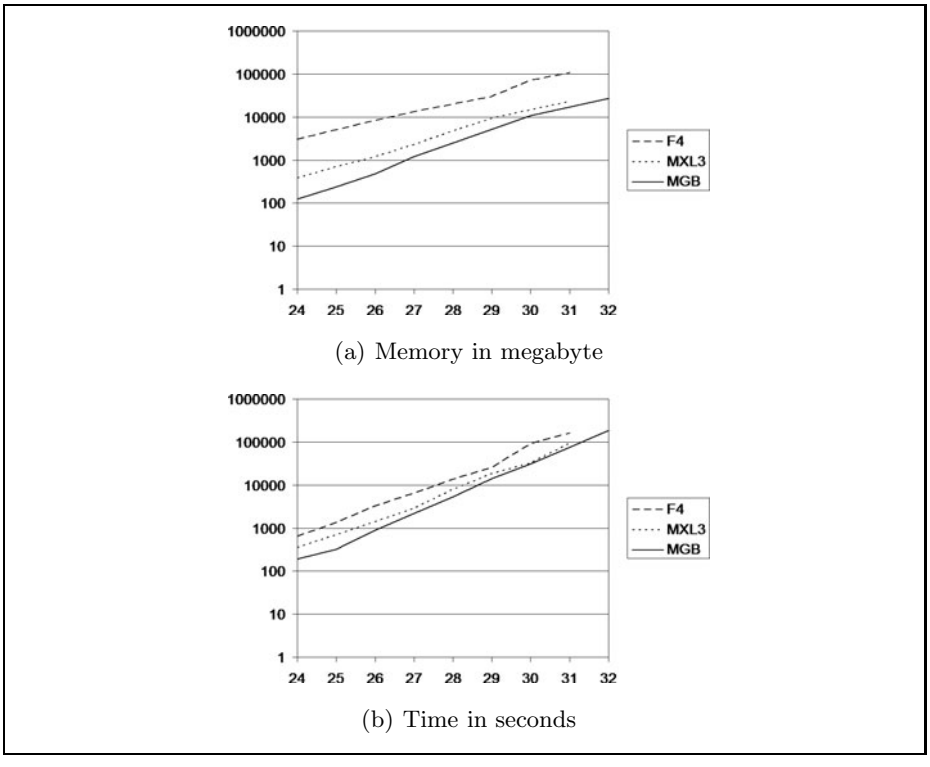

Fig. 3. Comparison between MGB, MXL_3, and F_4 for dense random systems

Table 3. Results for the system Random-32 with the MGB algorithm

Step	D	Matrix Size	Rank	partitions	n_D	NM	UM	MD
1	2	32×529	32	$\{x_1, x_2\}$	32	0	0	-
2	3	1056×5489	1056	$\{x_1, x_2, x_3\}$	32	0	0	-
3	4	11798×36954	11776	$\{x_2, x_3, x_4\}$	31	0	0	-
4	5	93534×179460	91378	$\{x_3, \ldots, x_7\}$	30	0	0	-
5	6	389286×475470	372679	$\{x_6, \ldots, x_{16}\}$	27	0	0	0
6	7	**437172×507294**	437172	$\{x_{15}, \ldots, x_{26}\}$	18	21445	2158	5
7	6	305685×314056	305685	$\{x_9, \ldots, x_{27}\}$	24	18589	199	4
8	5	175490×179460	175490	$\{x_3, \ldots, x_{28}\}$	30	3910	16	3
9	4	36875×36954	36875	$\{x_2, \ldots, x_{29}\}$	31	6	1	1
10	2	535×529	528	$\{x_1, \ldots, x_{31}\}$	32	25	0	1

Table 3 explains how the MGB algorithm works. As the degree D is going up the number of variables in degree D terms goes down. Starting from step 3, the number of variables of degree 4 terms starts to decrease. At step 6 the degree of the system reaches 7 by extending only two partitions of degree 6 polynomials (x_{15}, x_{16}), while the number of variables in degree 7 terms is only 18. The system starts to generate mutants. It generates 21445 mutants of degree 5. Only 2158 of

them are used. All of these mutants have leading variable x_9. So by multiplying them with variables $\geq x_9$, we have new polynomials of degree 6 with leading variables at most x_9. We do not multiply mutants by x_6, x_7, and x_8 since their partitions are not needed in the Gaussian elimination of step 7. This leads to decreasing the dimension of the matrix of the step. The system continuously generates low degree mutants until 6 linear mutants are produced at step 9. Another 25 linear ones are generated at step 10. By multiplying all mutants of degree ≤ 2, the system does not produce more mutants which in turn leads to a Gröbner basis of the ideal generated by the initial 32 polynomials.

For F_4, we used Magma version V2.13-10 implementation of Faugère's F_4 algorithm which is considered the best available implementation of F_4. When we use the new version of Magma (V2.16), we found no big difference between them. the new version is worse in terms of memory, while it is a little bit faster. For both, the MGB algorithm and the MXL_3 algorithm, we used our C++ implementation. For the *Echelonization* step, we used an adapted version of M4RI [1], the dense matrix linear algebra over $\mathbb{F}_2$ library. Our adaptation was in changing the strategy of selecting the pivot during Gaussian elimination to keep the old elements in the system intact. We use the M4RI method that has complexity $O(n^3/logn)$ [1].

7 Conclusions and Future Work

This paper presents a new strategy to improve algorithms to compute efficiently Gröbner bases. This new strategy is to use a more flexible partial enlargement technique that avoids computing polynomials at different degrees. As the first application of this strategy, we produced a new algorithm that has the ability of computing Gröbner bases more efficiently than the MXL_3 algorithm, which already performed better than the F_4 in Magma. Our experiments confirm that the new proposed algorithm is substantially better than MXL_3 and F_4 algorithms in both randomly generated instances of MQ and HFE systems and the experiment data also suggests that the complexity of the new algorithm challenges known theoretical estimates. Our preliminary complexity analysis of this new algorithm suggested that this new strategy may change substantially our thinking on the hardness of computing Gröbner bases and this new strategy of flexible partial enlargement may leads to new paradigms in Gröbner bases computation.

We plan to study the connection between the complexity of the algorithm presented in this paper and the complexity of other Gröbner bases algorithms. We are working on a priory complexity estimates for the algorithm and security levels of various cryptosystems based on this new algorithm. We are also experimenting with other heuristics that exploit the new strategy.

Acknowledgments

J. Ding and D. Cabarcas are especially grateful for the insightful and critical discussions with Professor Shigeo Tsujii and Mr. Masahito Gotaishi during a

2009 visit to Japan, which was supported by the "Strategic information and COmmunications R & D Promotion programmE" (SCOPE) from the Ministry of Internal Affairs and Communications of Japan. We also acknowledge useful comments from Stanislav Bulygin on a preliminary draft and valuable suggestions by anonymous referees. J. Ding would also like to thank partial support from NSF, NSF China and Taft Foundation for this project. D. Cabarcas is also partially supported by the Taft Foundation.

References

1. Albrecht, M., Bard, G.: M4RI – linear algebra over GF(2) (2008), `http://m4ri.sagemath.org/index.html`
2. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In: MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27-June 1 (2005)
3. Courtois, N.T.: Experimental algebraic cryptanalysis of block ciphers (2007), `http://www.cryptosystem.net/aes/toyciphers.html`
4. 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)
5. 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)
6. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F4). Pure and Applied Algebra 139(1-3), 61–88 (1999)
7. Gotaishi, M., Tsujii, S.: Hxl -a variant of xl algorithm computing gröbner bases. In: Presented in Special Track on Symbolic Computation and Cryptology of the 4th International Conference on Information Security and Cryptology (Inscrypt 2008) (December 2008)
8. Mohamed, M.S.E., Cabarcas, D., Ding, J., Buchmann, J., Bulygin, S.: MXL3: An efficient algorithm for computing gröbner bases of zero-dimensional ideals. In: ICISC 2009. LNCS. Springer, Heidelberg (2009) (accepted for publication)
9. 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)