

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

Johannes Buchmann*, Daniel Cabarcas†, Jantai Ding†,

Mohamed Saied Emam Mohamed*

*Technische Universität Darmstadt

†University of Cincinnati

Africacrypt 2010

Stellenbosch, South Africa, May 2010

- Motivation
- Gröbner Basis
- Algorithms for Computing GB
- Enlargement
- Flexible Partial Enlargement
- MGB Algorithm
- Experimental Results
- Future Work

Multivariate Cryptography

- Factoring and discrete logarithm: insecure under the assumption that quantum computer with enough Qbits exist
- Multivariate based cryptosystems: potential to resist the quantum computer attacks

Multivariate Cryptography

- Factoring and discrete logarithm: insecure under the assumption that quantum computer with enough Qbits exist
- Multivariate based cryptosystems: potential to resist the quantum computer attacks

Algebraic Cryptanalysis

Breaking a good cipher should require “as much work as solving a system of simultaneous equations in a large number of unknowns” (Shannon 1949)

The MQ Problem

- Given finite set of quadratic polynomials P in $X = \{x_1, \dots, x_n\}$ over finite field F
- Find $v \in F^n$, $p(v) = 0 \forall p \in P$, for example:
 - $x_1x_2 + x_1x_3 + x_2x_3 = 0$
 - $x_1x_3 + x_2x_3 + x_1 + 1 = 0$
 - $x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2 + 1 = 0$
- General MQ is NP-hard even if P is over \mathbb{F}_2

Algebraic Attacks

- Cryptosystem \rightarrow MQ Polynomial equations system
- Solving MQ polynomial equations system \rightarrow Recovering the secret

Algebraic Attacks

- Cryptosystem \rightarrow MQ Polynomial equations system
- Solving MQ polynomial equations system \rightarrow Recovering the secret

Attacking MPKCs

- PK is a set of MQ Polynomial equations
- Encryption = Evaluation
- Decryption = Inversion
- Multivariate encryption scheme
Multivariate systems \rightarrow Decrypting a ciphertext using PK
- Multivariate signature scheme
Multivariate systems \rightarrow Signing a message using PK

Attacking Block Cipher

- Using the pair of (known) plaintext-ciphertext values, the secret key and a large number of intermediate variables arising in the cipher
- Solving the resulting multivariate system is equivalent to recovering the secret key

Attacking Block Cipher

- Using the pair of (known) plaintext-ciphertext values, the secret key and a large number of intermediate variables arising in the cipher
- Solving the resulting multivariate system is equivalent to recovering the secret key

Attacking Stream Cipher

- Set up system of polynomial equations in unknown K and known keystream bits z_t

$$f_1(K, Z) = 0, \dots, f_N(K, Z) = 0$$

- Solving the multivariate system to get K

The key question

How to solve multivariate polynomial systems
efficiently?

The key question

How to solve multivariate polynomial systems
efficiently?

Experiments

- Dense random systems
- HFE systems of different univariate degrees

- Gröbner basis algorithms are the best known techniques for solving multivariate systems
- Definition: A Gröbner basis is a finite subset G of an ideal \mathcal{I} satisfying:

$$\langle \text{LT}(G) \rangle = \text{LT}(\mathcal{I})$$

- Properties:
 - Computing the variety of \mathcal{I}
 - Membership Problem

Matrix-based algorithms

- F_4 algorithm
- F_5 algorithm
- XL algorithm (single solution)
- MutantXL algorithm (single solution)
- MXL_3 algorithm

Matrix-based algorithms

- F_4 algorithm
- F_5 algorithm
- XL algorithm (single solution)
- MutantXL algorithm (single solution)
- MXL_3 algorithm

Algorithms for Computing GB

- Input: $P(x) = 0$
- Output: G a Gröbner basis of $\langle P(x) \rangle$

repeat

Echelonize(P)

$G = \text{Gröbner}(P)$

if termination criterion satisfied **then**

return G

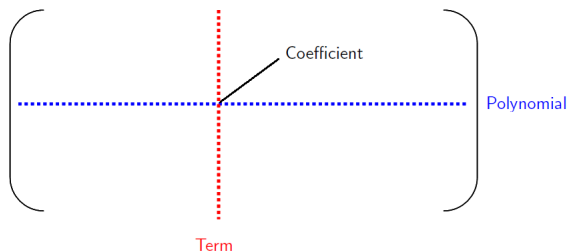
terminate

Enlarge(P)

Algorithms for Computing GB

Echelonize(P):

- Linearize(P)
- Gaussian Elimination
- Problem \rightarrow very large matrix and computation time



Termination Criterion:

- $F_4, F_5 \rightarrow$ No more pairs exist

Termination Criterion:

- $F_4, F_5 \rightarrow$ No more pairs exist
- $XL, \text{MutantXL} \rightarrow$ Computing univariate equations

Termination Criterion:

- $F_4, F_5 \rightarrow$ No more pairs exist
- XL, MutantXL \rightarrow Computing univariate equations
- $MXL_3 \rightarrow$ Saturation Criterion
- Computing G with highest degree d :
 - G contains all terms of degree d as leading terms
 - If $H = G \cup \{t \cdot g : g \in G \text{ deg}(t \cdot g) \leq d + 1\}$
 - No new $t \in \text{HT}(\tilde{H})$ and $\text{deg}(t) \leq d$

Enlarge(P):

- Extends P by multiplying a selected set of polynomials by monomials
- Affects on the size of the constructed matrix and the memory

$$\begin{array}{l} p_1 \\ p_2 \\ \vdots \\ p_m \end{array} \begin{pmatrix} t_1 & t_2 & \dots & 1 \\ \times & \times & \dots & \times \\ \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ \times & \times & \dots & \times \end{pmatrix}$$

Enlargement

Enlarge(P):

- Extends P by multiplying a selected set of polynomials by monomials
- Affects on the size of the constructed matrix and the memory

$$\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_m \end{array} \begin{pmatrix} t_1 & t_2 & \dots & 1 \\ \times & \times & \dots & \times \\ \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ \times & \times & \dots & \times \end{pmatrix} \xrightarrow[m_j \cdot p_i]{Enlarge} \rightarrow$$

Enlargement

Enlarge(P):

- Extends P by multiplying a selected set of polynomials by monomials
- Affects on the size of the constructed matrix and the memory

$$\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_m \end{array} \begin{pmatrix} t_1 & t_2 & \dots & 1 \\ \times & \times & \dots & \times \\ \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ \times & \times & \dots & \times \end{pmatrix} \xrightarrow[m_j \cdot p_i]{\text{Enlarge}} \begin{array}{c} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_k \end{array} \begin{pmatrix} s_1 & s_2 & s_3 & \dots & \dots & 1 \\ \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \times & \dots & \dots & \times \end{pmatrix}$$

Selection Strategy:

- F_4, F_5 algorithms \rightarrow S-Polynomials
- Selecting a subset of Critical pairs based on the degree of LCM of the pair

Selection Strategy:

- F_4, F_5 algorithms \rightarrow S-Polynomials
 - Selecting a subset of Critical pairs based on the degree of LCM of the pair
-
- XL algorithm \rightarrow No selection
 - Enlarge the system by multiplying P by all monomials up to a certain degree D

Selection Strategy:

- F_4, F_5 algorithms \rightarrow S-Polynomials
- Selecting a subset of Critical pairs based on the degree of LCM of the pair
- XL algorithm \rightarrow No selection
- Enlarge the system by multiplying P by all monomials up to a certain degree D
- MutantXL algorithm \rightarrow Mutant Criterion

Mutants

- Echelonize process yields polynomials of smaller degree than expected at a current degree D of the algorithm (mutants)
- Using them to enlarge the system before the degree goes up

Mutants

- Echelonize process yields polynomials of smaller degree than expected at a current degree D of the algorithm (mutants)
- Using them to enlarge the system before the degree goes up

$$p_1 = x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1$$

$$p_2 = x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 + x_2 + x_3 + 1$$

$$p_3 = x_1x_2 + x_1x_3 + x_2x_3 + x_3x_4 + x_1 + x_4 + 1$$

$$p_4 = \quad \quad \quad x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + 1$$

Which Polynomial Does MutantXL Multiply

- No mutants \rightarrow XL enlargement

Which Polynomial Does MutantXL Multiply

- No mutants \rightarrow XL enlargement

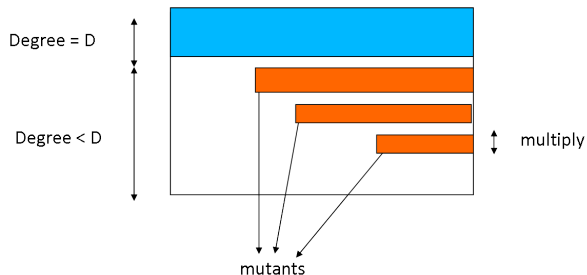


Which Polynomial Does MutantXL Multiply

- There are some mutants \rightarrow enlarge mutants

Which Polynomial Does MutantXL Multiply

- There are some mutants \rightarrow enlarge mutants



Compare performance

$$\begin{array}{ccc} \text{XL} & & \text{MutantXL} & & \text{F}_4 \\ \text{Mat. Size} & \geq & \text{Mat. Size} & \geq & \text{Mat. Size} \\ & | & & | & \\ \text{XL} & & \text{MutantXL} & & \text{F}_4 \\ \text{Time} & \geq & \text{Time} & \geq & \text{Time} \end{array}$$

Compare performance

XL
Mat. Size

\geq

MutantXL
Mat. Size

\geq

F₄
Mat. Size

XL
Time

\geq

MutantXL
Time

\geq

F₄
Time

↓
Mutant
Strategy

↓
Buchberger
Criterion

MXL₃ Enlargement

- XL strategy
- Improved Mutants Criterion
- Partial enlargement technique

MXL₃ Enlargement

- XL strategy
- Improved Mutants Criterion
- Partial enlargement technique

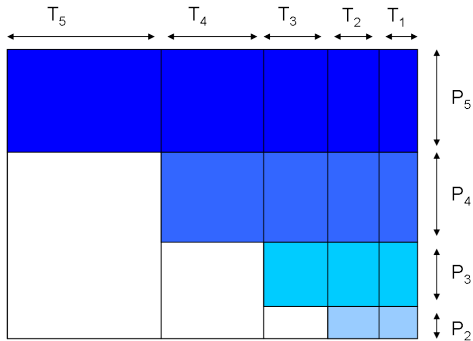
Notation

- $X := \{x_1, \dots, x_n\}$ set of variables, $x_1 > x_2 > \dots > x_n$
- $R \rightarrow$ Boolean ring over X , $P \subset R$
- Terms of R ordered by graded lex order
- Leading variable of $p \rightarrow$ largest variable in leading term of p
$$p = x_1 x_2 x_4 + x_1 x_3 x_5 + \dots$$
- $x_{partition}(P_d) \rightarrow$ Polynomials in P_d with leading variable x
- Saturated $x_{partition}(P_d) \rightarrow$ all degree d terms with leading variable x are appeared as leading terms in P_d

Enlargement

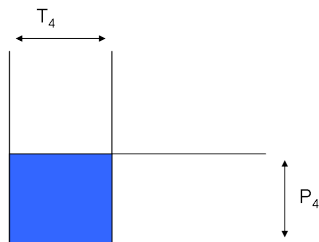
Degree-based enlargement

- F_4 , F_5 , XL, and MutantXL



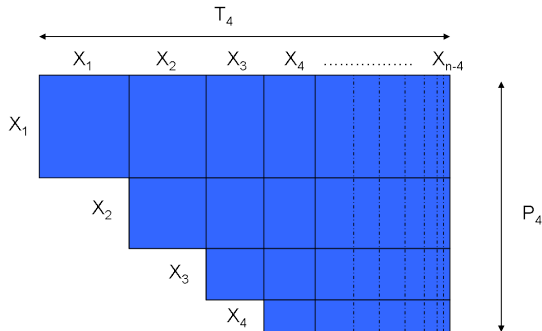
Degree-based enlargement

- F_4 , F_5 , XL, and MutantXL



Variable-partitions based enlargement

- $MXL_3 \rightarrow$ partial enlargement



Compare performance

MutantXL
Mat. Size

\geq

F_4
Mat. Size

\geq

MXL_3
Mat. Size

MutantXL
Time

\geq

F_4
Time

\geq

MXL_3
Time

Enlargement

Compare performance

MutantXL
Mat. Size

\geq

F_4
Mat. Size

\geq

MXL_3
Mat. Size


MutantXL
Time


\geq

F_4
Time

\geq

MXL_3
Time

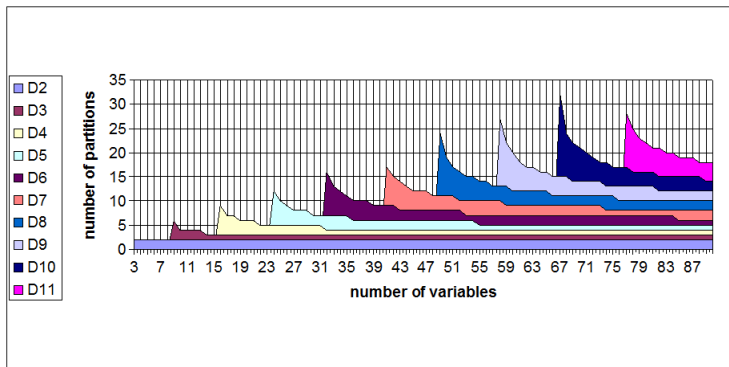

Buchberger
Criterion


Partial
Enlargement

Enlargement

XL Enlargement

- The number of partitions generated for random systems
- Let i partitions at degree $d \rightarrow$ at least $i - 1$ are saturated



Flexible partial enlargement

- Let $(x_1, \dots, x_j)_{Partitions}$ be the non-empty partitions of P_d and i partitions be saturated, $i \leq j$
- Enlarging only $(x_i, \dots, x_j)_{partitions}$ of P_d
- x_1, \dots, x_{j-1} excluded from P_{d+1}
- Experimentally: All $x_{partitions}$, ($x \leq x_j$) of P_{d+1} extended polynomials are created in the same way as MXL 3 does



- XL algorithm methodology
- Flexible partial enlargement technique
- Mutant strategy
- MXL_3 improvements
- MXL_3 termination criterion to detect a Gröbner basis

The MGB algorithm

MGB performance

F_4
Mat. Size

\geq

MXL_3
Mat. Size

\geq

MGB
Mat. Size

F_4
Time

\geq

MXL_3
Time

\geq

MGB
Time

The MGB algorithm

MGB performance

F_4
Mat. Size

\geq

MXL_3
Mat. Size

\geq

MGB
Mat. Size

F_4
Time

\geq

MXL_3
Time

\geq

MGB
Time

↓
Partial
Enlargement

↓
Flexible
Enlargement

The MGB algorithm

Algorithm in action

- Behavior of MGB for 32 variables random system.

Step	D	Matrix Size	Rank	Zeros	start	n_D
1	2	32×529	32	0	x_1	32
2	3	1056×5489	1056	0	x_1	32
3	4	11798×36954	11776	0.2	x_2	31
4	5	93534×179460	91378	2.3	x_3	30
5	6	389286×475470	372679	4.3	x_6	27
6	7	437172×507294	437172	0	x_{15}	18

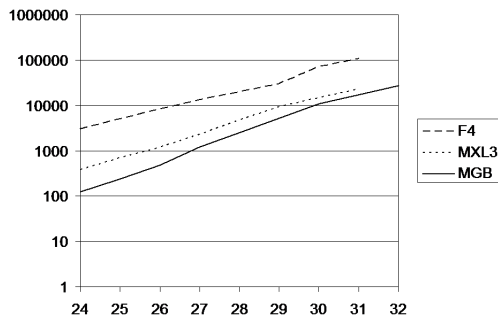
Table

- Experiments for dense random systems

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

Figure

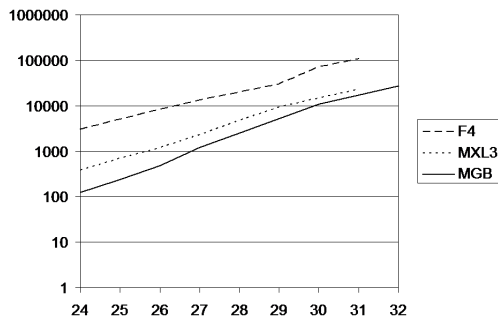
- Memory comparison between MXL_3 and F_4 for dense random systems



Experimental Results

Figure

- Time comparison between MXL_3 and F_4 for dense random systems
- Magma's F_4 Uses super linear algebra



Experimental Results

Table

- Experiments for HFE(288,n) systems

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

- Theoretically analyze the complexity of MGB
- Revisit the security of cryptosystems against MGB
- Improve the selection strategy used for MGB
- Combining MGB and F_4 strategies
- Adapt MGB for sparse systems (stream ciphers)
- Using sparse linear algebra (Wiedemann)

Thanks for your attention!