Matrix F_5

for the working cryptographer

Martin Albrecht (M.R.Albrecht@rhul.ac.uk)

Information Security Group,
Royal Holloway, University of London

November 27, 2008

Table of Contents

Information Security Group

Table of Contents

This Talk

- This talk is **not** about F_5 but
  - about Matrix F_5
  - and the basic ideas behind F_5.
- Matrix F_5 is not published in English, but
  - several French PhD theses and
  - several sets of slides by Jean-Charles Faugère

  exist describing it (in brief).
- The algorithm was explained to us by Ludovic Perret at Sage Days 12.
- John Perry and Christian Eder helped us to refine some points and to understand some relations to F_5.

Table of Contents

Notation

Information Security Group

- $\mathbb{K}$ is a field;
- $P = \mathbb{K}[x_0, \dots, x_{n-1}]$ is a polynomial ring;
- I is an ideal $\subset P$;
- J is the homogenisation of I.

We restrict our attention to homogeneous polynomials in this talk. This make everything much easier.

We note that while F_5 needs homogeneous inputs (or some sugar strategy) XL doesn't require homogeneous inputs.

Lazard's Theorem [Laz83] I

Information Security Group

Let $f_0, \dots f_{m-1}$ be homogeneous polynomials in P. We can construct the Macaulay matrix $\mathcal{M}^{acaulay}_{D,m}$. Write down horizontally all the degree D monomials from smallest to largest. Multiply each f_i by all monomials of degree $D - d_i$ where $d_i = deg(f_i)$.

$$\mathcal{M}^{acaulay}_{D,m} = \begin{array}{c c} & \text{monomials of degree } D \\ \begin{array}{c} (t_0, f_0) \\ (t_1, f_0) \\ \vdots \\ (u_0, f_1) \\ \vdots \\ (v_s, f_{m-1}) \end{array} & \left(\begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \right) \end{array}$$

Lazard's Theorem [Laz83] II

Theorem

For D "sufficiently" large Gaussian elimination on all $\mathcal{M}^{acaulay}_{d,m}$ *for* $1 \leq d \leq D$ *computes a Gröbner basis.*

Lazard's Theorem [Laz83] III

To see why this is true recall the definition of S-polynomials

Definition (S-Polynomial)

Let $f, g \in \mathbb{K}[x_1, \ldots, x_n]$ be polynomials $\neq 0$ and define $x^\gamma = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))$. Then the S-polynomial of f and g is defined as

$$S(f, g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g.$$

and multivariate polynomial division.

Lazard's Theorem [Laz83] IV

...we have got everything in these matrices we need.

- the S-polynomial for $S(f,g)$ with $x^\gamma = LCM(LM(f), LM(g))$ is represented in $\mathcal{M}^{acaulay}_{d,m}$ for $d = deg(x^\gamma)$ as the rows matching $\frac{x^\gamma}{LM(f)} \cdot f$ and $\frac{x^\gamma}{LM(g)} \cdot g$;
- all multiples of f_i of degree d are in $\mathcal{M}^{acaulay}_{d,m}$;
- ...Gaussian elimination takes care of the rest

Rediscovery: *XL* [CKPS00]

```
def gauss_elimination(F):
  A,v = CoefficientMatrix(F)
  E = EchelonForm(A)
  return E*v

def xl(F, D):
  M = "all monomials of degree D"
  Ftilde = []
  for f in F:
    for m in M:
      Ftilde.append(m*f)

  Ftilde = gauss_elimination(Ftilde)
  return Ftilde
```

XL & Gröbner Bases

```
def xlgb(F, D):
  basis = []
  for d in range(D+1):
    basis.extend( xl(F,d) )
  return basis
```

$$J = \langle x_0 + x_1 + x_2 + x_3, \\ x_0x_1 + x_1x_2 + x_0x_3 + x_2x_3, \\ x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3, \\ x_0x_1x_2x_3 - h^4\rangle$$

```
sage: P.<x0,x1,x2,x3> = PolynomialRing(GF(32003))
sage: J = sage.rings.ideal.Cyclic(P).homogenize()
sage: gb1 = xlgb(J, 3).reduced_basis()
sage: gb2 = J.groebner_basis()
sage: gb1 == gb2
True
```

XSL [CP02a] I

"In order to solve these equations, we are going to introduce an improved version of the XL approach from [CKPS00], that takes advantage of their specific structure and sparsity. We call it 'the XSL algorithm' where XSL stands for: 'eXtended Sparse Linearization' or 'multiply(X) by Selected monomials and Linearize'. In the XL algorithm, we would multiply each of the equations by all possible monomials of some degree $D - 2$, see [CKPS00]. Instead we will only **multiply them by carefully selected monomials**. *It seems that the best thing to do, is to use products of monomials, that already appear in other equations." – [CP02a]*

XSL [CP02a] II

"Therefore, no matter how large the parameter P [number of monomials, malb] is, there is no hope that the XSL algorithm (as described in [CP02a]) can solve the initial set of equations." – [CL05]

"Furthermore it should be clear that there seems to be no benefit in running this method [sXL, malb] instead of simply applying XL or XL2 to the simplified AES system of 8000 equations over 1600 variables described in [CP02b]." – [CL05]

. . . so is there a clever way to select the monomials?

XL + Critical Pairs: F_4 [Fau99]

You Heard About This Last Week

Table of Contents

Reconsider XL_{GB}

Information Security Group

```
def xlgb(F, D):
  basis = []
  for d in range(D+1):
   M = "all monomials of degree d"
    Ftilde = []
    for f in F:
      for m in M:
        Ftilde.append(m*f)
    Ftilde = gauss_elimination(Ftilde)
    basis.extend( Ftilde )
  return basis
```

Example

Information Security Group

$$J = \langle x_0 + x_1 + x_2 + x_3,$$
$$x_0x_1 + x_1x_2 + x_0x_3 + x_2x_3,$$
$$x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3,$$
$$x_0x_1x_2x_3 - h^4\rangle$$

over $\mathbb{F}_{32003}[x_0, x_1, x_2, x_3, h]$ with *degrevlex*.

d	$XL_{GB}1$
1	
2	4×11
3	20×34
4	60×69
5	140×125
6	280×209

$\tilde{F}$ vs. F I

Observation

If for the degree d the polynomial $m_j \cdot f_k$ reduces to zero then so will $x_i m_j \cdot f_k$ for degree $d+1$ and all $0 \leq i < n$.
So instead of starting from scratch in step $d+1$ from the f_is reuse the linear dependencies already discovered for degree d.

...this is the first criterion used by F_5: "Rewritten Criterion"

$\tilde{F}$ vs. F II

Information Security Group

```
def xlgb2(F, D):
    basis = F
    for d in range(1,D+1):
        Ftilde = []
        for f in F:
            for x in variables:
                Ftilde.append(x*f)
        Ftilde = gauss_elimination(Ftilde)
        F = [f for f in Ftilde if f != 0]
        basis.extend(F)
    return basis
```

$\tilde{F}$ vs. F III

Information Security Group

d	$XL_{GB}1$	$XL_{GB}2$
1		
2	4×11	
3	20×34	20×32
4	60×69	100×69
5	140×125	270×125
6	280×209	550×209

$\tilde{F}$ vs. F IV

That avoids one problem but introduces another: When the original code multiplied by e.g. xy only we will multiply by xy and yx due to the incremental strategy. We need to keep track by what monomials we multiplied already.

Definition (Signature)

A Signature is a tuple (m, f_i) attached to a row r of $\mathcal{M}_{d,m}^{acaulay}$, encoding that this row is the result of the multiplication $m \cdot f_i$.

$\tilde{F}$ vs. F V

Information Security Group

```
def xlgb3(F, D):
  for f in F:
     set_signature( (1, f), f)
  basis = F
  for d in range(1,D+1):
    Ftilde = []
    for h in F:
      m, fi = get_signature(h)
      for x in variables:
        if x < max( variables(m) ):
          continue
        Ftilde.append( x*h )
        set_signature( (x*m, fi), x*h )
    Ftilde = gauss_elimination(Ftilde)
    F = [h for h in Ftilde if h!=0]
    basis.extend(F)
  return basis
```

$\tilde{F}$ vs. F VI

d	$XL_{GB}1$	$XL_{GB}2$	$XL_{GB}3$
1			
2	4×11		
3	20×34	20×32	20×32
4	60×69	100×69	60×69
5	140×125	270×125	121×118
6	280×209	550×209	201×171

Trivial Syzygys I

Information Security Group

A syzygy for $F = (f_0, \dots, f_{m-1})$ is a vector $G = (g_0, \dots, g_{m-1})$ such that

$$\sum_{i=0}^{m-1} g_i f_i = 0.$$

We have that $g_i = f_j, g_j = -f_i, g_k = 0$ for $k \neq i, j$ is a trivial syzygy for F because

$$f_i f_j - f_j f_i = 0.$$

We want to avoid all reductions to zero caused by these trivial relations.

Trivial Syzygys II

Information Security Group

Consider f_0, f_1, f_2 as an example. A combination of the trivial relations $f_i f_j = f_j f_i$ can always be written as

$$u(f_1 f_2 - f_2 f_1) + v(f_0 f_2 - f_2 f_0) + w(f_1 f_0 - f_0 f_1)$$

where u, v, w are arbitrary polynomials. This can be rewritten

$$(uf_1 + vf_0)f_2 - uf_1 f_2 - vf_0 f_2 + wf_1 f_0 - wf_0 f_1$$

Hence the (trivial) relations for f_2 are in the ideal generated by f_0 and f_1. So it is easy to remove lines if we have compute the Gröbner basis for $\langle f_0, f_1 \rangle$ already.
So, we need to restrict elimination, such that we iteratively compute the Gröbner basis for $\langle f_0 \rangle$, $\langle f_0, f_1 \rangle$ etc.

Trivial Syzygys III

A more general way of putting it:

> *Given signatures of a current basis, when considering whether to generate a new polynomial — when computing $x \cdot h$ —, if the normal way of computing the signature — $x \cdot m, f_i$ — would give a signature that is recognizably larger than it needs to be, then there is a syzygy that allows one to rewrite the polynomial with a smaller signature. Top-cancellations with smaller signatures have already been considered, so the polynomial can be discarded.*

— John Perry

Gaussian Top Elimination I

```
def gauss_elimination2(F):
  A,v = CoefficientMatrix(F)
  for c in range(A.ncols()):
    for r in range(0,A.nrows()):
      if A[r,c] != 0: # is pivot?
        if any(A[r,i] for i in xrange(c)):
          continue # this wouldn't happen normally

       A.rescale_row(r, A[r,c]^(-1))
        for i in range(r+1,A.nrows()):
          if A[i,c] != 0: # clear below?
            if any(A[i,k] for k in range(c)):
              continue # this wouldn't happen normally
           A.add_multiple_of_row(i, r, -A[i,c], c)
        break
  return (A*v)
```

Gaussian Top Elimination II

We perform normal Gaussian elimination, but:

- we don't compute the **reduced** row echelon form
- we dont' allow row swaps
- we don't allow lower rows to affect higher rows ever

The F_5 Criterion [Fau02]

Information Security Group

To detect redudant rows we can apply the following theorem due to Jean-Charles Faugère.

Theorem (F_5 Criterion)

For all $j < m$, if we have a row labeled (t, f_j) in the matrix $\mathcal{M}^{acaulay}_{D-d_m,m-1}$ that has leading term t' then the row (t', f_m) in $\mathcal{M}^{acaulay}_{D,m}$ is redundant.

If $\exists g \in \mathcal{M}^{acaulay}_{D-\mathbf{d_m},m-1}$ with $LM(g) = \mathbf{t'} \longrightarrow h \notin \mathcal{M}^{acaulay}_{D,m}$ with $signature(h) = (t', f_m)$.

Matrix F_5 I

Information Security Group

```
def matrixf5(F, D):
  for d in range(1,D+1):
    for fi in F:
      if deg(fi) == d:
        add_signature((1, fi), fi)
       M[d].append(fi); continue
      for f in M[d-1] with get_signature(f) == (*,fi):
        m, fi = get_signature(f)
        for x in variables:
          if x < max(mult.variables()):
               continue
          if t in M[d-deg(fi)] with LM(t) == x*m:
               m2,fj = get_signature(t)
               if j < i:
                 continue
          add_signature((x*m, fi), x*f)
         M[d].append( x*f )

    M[d] = [f for f in gauss_elimination2(M[d]) if f!=0]
  return [f for d in range(D+1) for f in M[d]]
```

Matrix F_5 II

Information Security Group

d	$XL_{GB}1$	$XL_{GB}2$	$XL_{GB}3$	Matrix F_5	F_4
1					
2	4×11			4×11	4×11
3	20×34	20×32	20×32	20×34	15×28
4	60×69	100×69	60×69	54×69	37×44
5	140×125	270×125	121×118	110×125	31×36
6	280×209	550×209	201×171	194×209	

Table of Contents

Information Security Group

F_5 Critera & Buchberger's Criteria

Information Security Group

The F_5 criteria are **not** generalisations of Buchberger's criteria

For example consider

$$\begin{aligned}
(1, f_0): \quad f_0 &= xy + \dots \\
(1, f_1): \quad f_1 &= z^2 + \dots \\
(1, f_2): \quad f_2 &= yz^2 + \dots
\end{aligned}$$

Buchberger's first criterion tells us that $S(f_0, f_1)$ reduces to zero, since $GCD(z^2, xy) = 1$. However, in F_5 we restrict elimination such that this reduction (to zero) might not be performed.

XL and Matrix F_5

Information Security Group

- Matrix F_5 removes **only** rows from $\mathcal{M}^{acaulay}_{d,m}$ if we know that they are redundant.
- *XL* thus cannot be more efficient than Matrix F_5 because it strictly does more useless work.
- One can do to Matrix F_5 matrices whatever one can do to *XL* matrices, **as long as ordering is preserved**
  - MutantMatrixF_5 ?
  - MatrixF_5-Wiedemann [FJ03]
  - GeometryMatrixF_5?

F_4, F_5 and Matrix F_5

Information Security Group

- Even if we don't reduce to zero, we still compute a lot of useless information in Matrix F_5: linear algebra & F_5 criteria;
- F_4 is more efficient in many examples because it only considers critical pairs: linear algebra & critical pairs;
- F_5 also only considers critical pairs, thus is much more efficient than Matrix F_5 for sparse examples: F_5 criteria & critical pairs;
- $\rightarrow$ F_4-style F_5: linear algebra & F_5 criteria & critical pairs

Questions?

Thank You!

References I

Information Security Group

Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir.
Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations.
In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer Verlag, 2000.

Carlos Cid and Gaëtan Leurent.
An Analysis of the XSL Algorithm.
In *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer Verlag, 2005.

Nicolas T. Courtois and Josef Pieprzyk.
Cryptanalysis of Block Ciphers with Overdefined Systems of Equations.
In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer Verlag, 2002.

Nicolas T. Courtois and Josef Pieprzyk.
Cryptanalysis of Block Ciphers with Overdefined Systems of Equations.
Cryptology ePrint Archive, Report 2002/044, 2002.
Available at `http://eprint.iacr.org/2002/044`.

Jean-Charles Faugère.
A New Efficient algorithm for Computing Gröbner Basis (F4), 1999.
Available at `http://modular.ucsd.edu/129-05/refs/faugere_f4.pdf`.

Jean-Charles Faugère.
A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5).
In *Proceedings of ISSAC*, pages 75–83. ACM Press, 2002.

References II

Jean-Charles Faugère and Antoine Joux.
Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases.
In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

D. Lazard.
Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations.
In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *Lectures Notes in Computer Science*. Springer-Verlag, 1983.