Polynomial Selection Using Lattices

Mathias Herrmann Alexander May Maike Ritzenhofen

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr-University Bochum

FACTORING 2009
September 12$^{\text{th}}$

Intro

- Need to find 2 irreducible polynomials $f_1(x), f_2(x) \in \mathbb{Z}[x]$ with common root m modulo N.

Intro

- Need to find 2 irreducible polynomials $f_1(x), f_2(x) \in \mathbb{Z}[x]$ with common root m modulo N.
- $deg(f_1(x)) = 5$ (*algebraic polynomial*)
- $deg(f_2(x)) = 1$ (*linear polynomial*)
- Homogeneous form:

$$
\begin{aligned}
F_1(x,z) &= a_d x^d + a_{d_1} x^{d-1} z + \ldots + a_0 z^d \\
F_2(x,z) &= x - mz
\end{aligned}
$$

Intro

- Need to find 2 irreducible polynomials $f_1(x), f_2(x) \in \mathbb{Z}[x]$ with common root m modulo N.
- $deg(f_1(x)) = 5$ (*algebraic polynomial*)
- $deg(f_2(x)) = 1$ (*linear polynomial*)
- Homogeneous form:

$$\begin{aligned} F_1(x,z) &= a_d x^d + a_{d_1} x^{d-1} z + \ldots + a_0 z^d \\ F_2(x,z) &= x - mz \end{aligned}$$

- Currently, algorithm of Thorsten Kleinjung yields the best polynomial pairs.

"Good" Polynomials

- Want to find many pairs $(a, b) \in \mathbb{Z}^2$ such that $F_1(a, b)$ and $F_2(a, b)$ are smooth.

"Good" Polynomials

- Want to find many pairs $(a, b) \in \mathbb{Z}^2$ such that $F_1(a, b)$ and $F_2(a, b)$ are smooth.
- Need to be able to (quickly) compare polynomial (pairs).

"Good" Polynomials

- Want to find many pairs $(a, b) \in \mathbb{Z}^2$ such that $F_1(a, b)$ and $F_2(a, b)$ are smooth.
- Need to be able to (quickly) compare polynomial (pairs).

Quality Measures

$$Q_2(f_1, f_2) = \int_0^{\pi} \rho\left(\frac{\alpha(F_1) + \log F_1(-A\cos\theta, B\sin\theta)}{\log L_1}\right) \cdot \rho\left(\frac{\alpha(F_2) + \log F_2(-A\cos\theta, B\sin\theta)}{\log L_2}\right) d\theta$$

"Good" Polynomials

- Want to find many pairs $(a, b) \in \mathbb{Z}^2$ such that $F_1(a, b)$ and $F_2(a, b)$ are smooth.
- Need to be able to (quickly) compare polynomial (pairs).

Quality Measures

$$Q_2(f_1, f_2) = \int_0^\pi \rho\left(\frac{\alpha(F_1) + \log F_1(-A\cos\theta, B\sin\theta)}{\log L_1}\right) \cdot \rho\left(\frac{\alpha(F_2) + \log F_2(-A\cos\theta, B\sin\theta)}{\log L_2}\right) d\theta$$

$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2}\log\left(\int_{\substack{|a| \le A \\ 0 < b \le B}} F_1(a, b)^2 \, da\, db\right)$$

"Good" Polynomials

- Want to find many pairs $(a, b) \in \mathbb{Z}^2$ such that $F_1(a, b)$ and $F_2(a, b)$ are smooth.
- Need to be able to (quickly) compare polynomial (pairs).

Quality Measures

$$Q_2(f_1, f_2) = \int_0^{\pi} \rho\left(\frac{\alpha(F_1) + \log F_1(-A\cos\theta, B\sin\theta)}{\log L_1}\right) \cdot \rho\left(\frac{\alpha(F_2) + \log F_2(-A\cos\theta, B\sin\theta)}{\log L_2}\right) d\theta$$

$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2}\log\left(\int_{\substack{|a| \leq A \\ 0 < b \leq B}} F_1(a, b)^2 \, da \, db\right)$$

$$Q_4(f_1) = max_{0 \leq i \leq d} |a_i| s^{d - \frac{i}{2}}$$

Basics in Lattices

Definition

Let $b_1, \ldots, b_n \in \mathbb{Q}^n$ be linearly independent vectors. The set

$$L := \left\{ x \in \mathbb{Q}^n \mid x = \sum_{i=1}^{n} a_i b_i, \quad a_i \in \mathbb{Z} \right\}$$

is a lattice.

Basics in Lattices

Definition

Let $b_1, \ldots, b_n \in \mathbb{Q}^n$ be linearly independent vectors. The set

$$L := \left\{ x \in \mathbb{Q}^n \mid x = \sum_{i=1}^{n} a_i b_i, \quad a_i \in \mathbb{Z} \right\}$$

is a lattice.

Described by basis matrix

$$B(L) = \begin{pmatrix} ---b_1--- \\ \vdots \\ ---b_n--- \end{pmatrix}$$

Basic polynomial selection using lattices

Question

How can we use lattices to perform a polynomial selection?

Basic polynomial selection using lattices

Question

How can we use lattices to perform a polynomial selection?

- Fix a root m modulo N.
- Linear polynomial is just $f_2(x) = x - m$.

Basic polynomial selection using lattices

Question

How can we use lattices to perform a polynomial selection?

- Fix a root m modulo N.
- Linear polynomial is just $f_2(x) = x - m$.
- For algebraic polynomial use the following basis matrix.

$$\begin{array}{cccccc} 1 & x & x^2 & \dots & x^d & f_1(m) \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & m \\ 0 & 0 & 1 & \dots & 0 & m^2 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & m^d \\ 0 & 0 & 0 & \dots & 0 & N \end{pmatrix}$$

Basic polynomial selection using lattices

Question

How can we use lattices to perform a polynomial selection?

- Fix a root m modulo N.
- Linear polynomial is just $f_2(x) = x - m$.
- For algebraic polynomial use the following basis matrix.

$$
\begin{array}{cccccc}
1 & x & x^2 & \dots & x^d & f_1(m)
\end{array}
$$

$$
\begin{pmatrix}
1 & 0 & 0 & \dots & 0 & 1 \\
0 & 1 & 0 & \dots & 0 & m \\
0 & 0 & 1 & \dots & 0 & m^2 \\
\vdots & & & \ddots & & \vdots \\
0 & 0 & 0 & \dots & 1 & m^d \\
0 & 0 & 0 & \dots & 0 & N
\end{pmatrix}
$$

Lattice reduction yields short lattice vector, i.e. polynomial with small coefficients.

Skewness

More sieve reports, if sieving region and polynomial are *skewed*.

$$-46023405120x^5 - 10480176714921624x^4 + 2932832430995490310360 3x^3$$

$$+8308379750900490013986116 63x^2 + 4445551794113058682649421551 8773x$$

$$+1303524908152518880895019863 45593$$

Skewness

More sieve reports, if sieving region and polynomial are *skewed*.

$$-46023405120x^5 - 10480176714921624x^4 + 2932832430995490310360 3x^3$$

$$+8308379750900490013986116 63x^2 + 44455517941130586826494215518773x$$

$$+1303524908152518880895019863 45593$$

Skewness in lattice basis

Multiply basis matrix with a weight matrix that forces the polynomial to be skewed.

$$W = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & s & 0 & \dots & 0 & 0 \\ 0 & 0 & s^2 & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & s^d & 0 \\ 0 & 0 & 0 & \dots & 0 & S \end{pmatrix}.$$

After LLL reduction apply the inverse scaling to obtain desired polynomial.

Result

- Obtain (skewed) polynomial with small coefficients.
- $\Rightarrow$ *good* polynomial with respect to

$$Q_4(f_1) = max_{0 \leq i \leq d} |a_i| s^{d - \frac{i}{2}}$$

.

Result

- Obtain (skewed) polynomial with small coefficients.
- $\Rightarrow$ *good* polynomial with respect to

$$Q_4(f_1) = max_{0 \leq i \leq d} |a_i| s^{d - \frac{i}{2}}$$

.

- BUT: We want to use a better approximation of number of sieve reports.

Result

- Obtain (skewed) polynomial with small coefficients.
- $\Rightarrow$ *good* polynomial with respect to

$$Q_4(f_1) = max_{0 \leq i \leq d} |a_i| s^{d - \frac{i}{2}}$$

.

- BUT: We want to use a better approximation of number of sieve reports.
- $\Rightarrow$ Use different norm for LLL to capture quality with respect to Q_3.

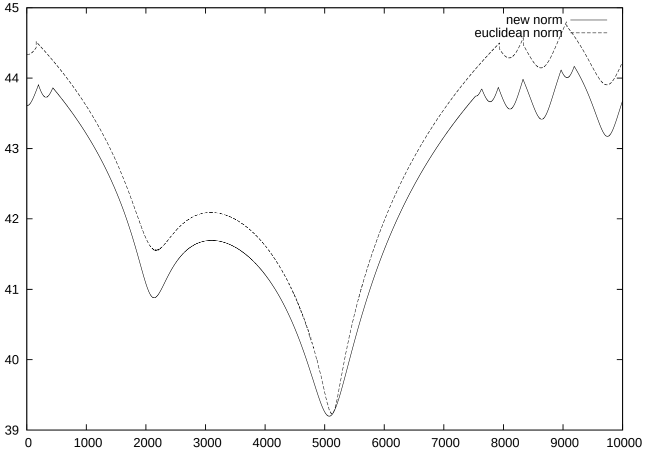
$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2} \log \left(\int_{\substack{|a| \leq A \\ 0 < b \leq B}} F_1(a,b)^2 \, da \, db \right)$$

Modified Norm

Alter norm used by LLL. Define $||v|| := v^T M v$ with

$$M := \begin{pmatrix} \frac{2}{11}s^{-5} & 0 & \frac{2}{27}s^{-3} & 0 & \frac{2}{35}s^{-1} & 0 & 0 \\ 0 & \frac{2}{27}s^{-3} & 0 & \frac{2}{35}s^{-1} & 0 & \frac{2}{35}s & 0 \\ \frac{2}{27}s^{-3} & 0 & \frac{2}{35}s^{-1} & 0 & \frac{2}{35}s & 0 & 0 \\ 0 & \frac{2}{35}s^{-1} & 0 & \frac{2}{35}s & 0 & \frac{1}{27}s^{3} & 0 \\ \frac{2}{35}s^{-1} & 0 & \frac{2}{35}s & 0 & \frac{1}{27}s^{3} & 0 & 0 \\ 0 & \frac{2}{35}s & 0 & \frac{1}{27}s^{3} & 0 & \frac{2}{11}s^{5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S \end{pmatrix}.$$

Comparison: Quality with standard norm vs. new norm



Root Property

- Recall quality measure

$$Q_3(f_1) = \underbrace{\alpha(F_1)}_{\text{Root property}} + \underbrace{\frac{1}{2}\log\left(\int_{\substack{|a| \leq A \\ 0 < b \leq B}} F_1(a,b)^2 \, da \, db\right)}_{\text{Size property}}$$

Root Property

- Recall quality measure

$$Q_3(f_1) = \underbrace{\alpha(F_1)}_{\text{Root property}} + \underbrace{\frac{1}{2}\log\left(\int_{\substack{|a| \leq A \\ 0 < b \leq B}} F_1(a,b)^2 \, da \, db\right)}_{\text{Size property}}$$

- Size property optimed by LLL, but
- Root property $\alpha(F_1)$ has major influence on quality.
- Need to model in lattice.

Lattice basis with improved root property

$$
\begin{array}{ccccccccc}
1 & x & x^2 & \dots & x^d & f_1(m) & f_1(\alpha_1) & \dots & f_1(\alpha_k)
\end{array}
$$

$$
\begin{pmatrix}
1 & 0 & 0 & \dots & 0 & 1 & \alpha_1^0 & \dots & \alpha_k^0 \\
0 & 1 & 0 & \dots & 0 & m & \alpha_1^1 & \dots & \alpha_k^1 \\
0 & 0 & 1 & \dots & 0 & m^2 & \alpha_1^2 & \dots & \alpha_k^2 \\
\vdots & & & \ddots & & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 1 & m^d & \alpha_1^d & \dots & \alpha_k^d \\
0 & 0 & 0 & \dots & 0 & N & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & p_1 & \dots & 0 \\
0 & 0 & 0 & \ddots & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & p_k
\end{pmatrix}
$$

Remarks

- Using some roots modulo small primes of Kleinjung's polynomial, we are able to reconstruct it with the lattice approach.

Remarks

- Using some roots modulo small primes of Kleinjung's polynomial, we are able to reconstruct it with the lattice approach.
- However, we are not able to allow LLL to choose a set of roots modulo small primes.

Remarks

- Using some roots modulo small primes of Kleinjung's polynomial, we are able to reconstruct it with the lattice approach.
- However, we are not able to allow LLL to choose a set of roots modulo small primes.
- Trying all possibilities to complex,
- No iterative method obvious.

Remarks

- Using some roots modulo small primes of Kleinjung's polynomial, we are able to reconstruct it with the lattice approach.
- However, we are not able to allow LLL to choose a set of roots modulo small primes.
- Trying all possibilities to complex,
- No iterative method obvious.

- Need a different approach to obtain a good root property.

Special Galois groups

- A theorem by Odoni states that asymptotically a polynomial, where the Galois group is the Frobenius group, yields smaller values.

Special Galois groups

- A theorem by Odoni states that asymptotically a polynomial, where the Galois group is the Frobenius group, yields smaller values.
- Our goal: Use LLL to find good polynomials with the special Galois group.

Generic polynomials with Galois group F_{20}

$$
\begin{aligned}
f_{gen}(x; a, b) \quad := \quad & x^5 + \left(b^2(a^2+4) - 2a - \frac{17}{4}\right) x^4 + \left(3b(a^2+4) + (a^2+4) + \frac{13a}{2} + 1\right) x^3 \\
& - \left(b(a^2+4) + \frac{11a}{2} - 8\right) x^2 + (a-6)\, x + 1 \\
f_{gen}(x; p, q) \quad := \quad & x^5 + 10px^3 + 20p^2 x + q
\end{aligned}
$$

Special Galois groups

- A theorem by Odoni states that asymptotically a polynomial, where the Galois group is the Frobenius group, yields smaller values.
- Our goal: Use LLL to find good polynomials with the special Galois group.

Generic polynomials with Galois group F_{20}

$$\begin{aligned} f_{gen}(x;a,b) &:= x^5 + \left(b^2(a^2+4) - 2a - \frac{17}{4}\right)x^4 + \left(3b(a^2+4) + (a^2+4) + \frac{13a}{2} + 1\right)x^3 \\ &\quad - \left(b(a^2+4) + \frac{11a}{2} - 8\right)x^2 + (a-6)x + 1 \\ f_{gen}(x;p,q) &:= x^5 + 10px^3 + 20p^2x + q \end{aligned}$$

- But: unable to find a root m modulo N.

Enforce special Galois group

- Generic polynomial with Frobenius group as Galois group.
- Compute roots modulo small primes and use these as input to lattice reduction, randomly chosen m.

Enforce special Galois group

- Generic polynomial with Frobenius group as Galois group.
- Compute roots modulo small primes and use these as input to lattice reduction, randomly chosen m.
- Hope that roots enforce the resulting polynomial to have the desired Galois group.

Enforce special Galois group

- Generic polynomial with Frobenius group as Galois group.
- Compute roots modulo small primes and use these as input to lattice reduction, randomly chosen m.
- Hope that roots enforce the resulting polynomial to have the desired Galois group.

Only happened if a lot of roots modulo small primes were enforce, but then the size of coefficients was very bad.

Optimize parameters a, b

- Try to find parameters of the generic polynomials.

Optimize parameters a, b

- Try to find parameters of the generic polynomials.
- Fix m and use Coppersmith's Algorithm to find a, b s.th. $f_{gen}(m; a, b) = 0 \bmod N$.
- Two problems arise:
  1. Upper bounds on a, b are very small. Experiments never found suitable a, b.
  2. Even if we found good algebraic polynomial, the *polynomial pair* may still be bad.

Optimize parameters a, b

- Try to find parameters of the generic polynomials.
- Fix m and use Coppersmith's Algorithm to find a, b s.th. $f_{gen}(m; a, b) = 0 \bmod N$.
- Two problems arise:
  1. Upper bounds on a, b are very small. Experiments never found suitable a, b.
  2. Even if we found good algebraic polynomial, the *polynomial pair* may still be bad.
- Add m as a further variable in Coppersmith's algortihm.
- But then the bounds get even worse and we cannot expect a solution.

Finding a root m

- All previous approaches used a given root m.
- Now: Try to find a good m with lattice methods.

Finding a root m

- All previous approaches used a given root m.
- Now: Try to find a good m with lattice methods.

Translation

Given a polynomial $f(x)$ with root m modulo N, the polynomial $f'(x) := f(x - \alpha)$ has root $m' = m + \alpha$.

$$(f'(m') = f(m' - \alpha) = f(m + \alpha - \alpha) = 0 \bmod N)$$

Idea

- Start with arbitrary polynomial (known root m modulo N).
- Compute coefficients of translated polynomials.

Idea

- Start with arbitrary polynomial (known root m modulo N).
- Compute coefficients of translated polynomials.

Translated polynomials

$$\begin{aligned} f_1(x) &= px - m \\ f_2(x) &= a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \end{aligned}$$

$$\begin{aligned} f_1'(x) &= f_1(x-\alpha) \\ &= p(x-\alpha) - m = px - (p\alpha + m) \\ f_2'(x) &= f_2(x-\alpha) \\ &= a_5x^5 + (a_4 - 5a_5\alpha)x^4 + (a_3 - 4a_4\alpha + 10a_5\alpha^2)x^3 \\ &\quad +(a_2 - 3a_3\alpha + 6a_4\alpha^2 - 10a_5\alpha^3)x^2 \\ &\quad +(a_1 - 2a_2\alpha + 3a_3\alpha^2 - 4a_4\alpha^3 + 5a_5\alpha^4)x \\ &\quad +(a_0 - a_1\alpha + a_2\alpha^2 - a_3\alpha^3 + a_4\alpha^4 - a_5\alpha^5). \end{aligned}$$

Problem Description

- Find α such that coefficients of translated polynomial are small.

Problem Description

- Find α such that coefficients of translated polynomial are small.

System of modular equations

$$
\begin{aligned}
g_1(\alpha) = p\alpha - m &= \epsilon_1 \bmod N \\
g_2(\alpha) = a_4 - 5a_5\alpha &= \epsilon_2 \bmod N \\
g_3(\alpha) = a_3 - 4a_4\alpha + 10a_5\alpha^2 &= \epsilon_3 \bmod N \\
g_4(\alpha) = a_2 - 3a_3\alpha + 6a_4\alpha^2 - 10a_5\alpha^3 &= \epsilon_4 \bmod N \\
g_5(\alpha) = a_1 - 2a_2\alpha + 3a_3\alpha^2 - 4a_4\alpha^3 + 5a_5\alpha^4 &= \epsilon_5 \bmod N \\
g_6(\alpha) = a_0 - a_1\alpha + a_2\alpha^2 - a_3\alpha^3 + a_4\alpha^4 - a_5\alpha^5 &= \epsilon_6 \bmod N.
\end{aligned}
$$

First Approach: SVP

- Solve SVP!

Lattice L_1

$$
\begin{array}{c|cccccccccccc|}
 & & & & & & & g_1(\alpha) & g_2(\alpha) & g_3(\alpha) & g_4(\alpha) & g_5(\alpha) & g_6(\alpha) \\
1 & 1 & & & & & & -m & a_4 & a_3 & a_2 & a_1 & a_0 \\
\alpha & & 1 & & & & & p & -5a_5 & -4a_4 & -3a_3 & -2a_2 & -a_1 \\
\alpha^2 & & & 1 & & & & & & 10a_5 & 6a_4 & 3a_3 & a_2 \\
\alpha^3 & & & & 1 & & & & & & -10a_5 & -4a_4 & -a_3 \\
\alpha^4 & & & & & 1 & & & & & & 5a_5 & a_4 \\
\alpha^5 & & & & & & 1 & & & & & & -a_5 \\
 & & & & & & & N & & & & & \\
 & & & & & & & & N & & & & \\
 & & & & & & & & & N & & & \\
 & & & & & & & & & & N & & \\
 & & & & & & & & & & & N & \\
 & & & & & & & & & & & & N
\end{array}
$$

First Approach: SVP

- Solve SVP!

Lattice L_1

$$
\begin{array}{c}
\\ 1 \\ \alpha \\ \alpha^2 \\ \alpha^3 \\ \alpha^4 \\ \alpha^5 \\ \\ \\ \\ \\ \\ \\
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
 & & & & & & g_1(\alpha) & g_2(\alpha) & g_3(\alpha) & g_4(\alpha) & g_5(\alpha) & g_6(\alpha)
\end{array} \\
\begin{pmatrix}
1 & & & & & & -m & a_4 & a_3 & a_2 & a_1 & a_0 \\
 & 1 & & & & & p & -5a_5 & -4a_4 & -3a_3 & -2a_2 & -a_1 \\
 & & 1 & & & & & & 10a_5 & 6a_4 & 3a_3 & a_2 \\
 & & & 1 & & & & & & -10a_5 & -4a_4 & -a_3 \\
 & & & & 1 & & & & & & 5a_5 & a_4 \\
 & & & & & 1 & & & & & & -a_5 \\
 & & & & & & N & & & & & \\
 & & & & & & & N & & & & \\
 & & & & & & & & N & & & \\
 & & & & & & & & & N & & \\
 & & & & & & & & & & N & \\
 & & & & & & & & & & & N
\end{pmatrix}
\end{array}
$$

Target vector $t = (1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6)$.

Results

- Not possible to enforce geometric progression $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5)$ of the first components.

Results

- Not possible to enforce geometric progression $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5)$ of the first components.
- Target vector is not among the short vectors in this lattice.

Results

- Not possible to enforce geometric progression $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5)$ of the first components.
- Target vector is not among the short vectors in this lattice.
- Need a different approach.

Second Approach:

LLL-Property

Let $B = b_1, \ldots, b_n$ be LLL-reduced. Then the Gram-Schmidt-orthogonalized vectors b_i^* fulfill

$$|b_i^*| \geq 2^{-\frac{i-1}{4}} \left(\frac{det(L)}{b_{max}} \right)^{\frac{1}{i}}.$$

Second Approach:

LLL-Property

Let $B = b_1, \ldots, b_n$ be LLL-reduced. Then the Gram-Schmidt-orthogonalized vectors b_i^* fulfill

$$|b_i^*| \geq 2^{-\frac{i-1}{4}} \left(\frac{det(L)}{b_{max}} \right)^{\frac{1}{i}}.$$

- If target vector is larger than orthogonalized vector, then $< b_i^*, t >= 0$ gives polynomial equation.

Second Approach:

LLL-Property

Let $B = b_1, \ldots, b_n$ be LLL-reduced. Then the Gram-Schmidt-orthogonalized vectors b_i^* fulfill

$$|b_i^*| \geq 2^{-\frac{i-1}{4}} \left(\frac{det(L)}{b_{max}} \right)^{\frac{1}{i}} .$$

- If target vector is larger than orthogonalized vector, then $< b_i^*, t >= 0$ gives polynomial equation.
- System of modular equations has 7 unknowns.
- If we find at least 7 orthogonal vectors that are larger than target vector, then we may be able to compute a solution.

Remarks

- The lattice L_1 only yields 6 polynomials.

Remarks

- The lattice L_1 only yields 6 polynomials.

Extending the lattice basis

Use additionally powers and products of the g_i.

$$
\begin{array}{c}
 \\ 1 \\ \alpha \\ \alpha^2 \\ \alpha^3 \\ \alpha^4 \\ \alpha^5 \\ \\ \\ \\ \\ \\ \\
\end{array}
\begin{array}{cccccccccccc}
 & & & & & & g_1(\alpha) & g_2(\alpha) & g_3(\alpha) & \dots & g_1^2(\alpha) & g_1 g_2(\alpha) & \dots \\
\end{array}
$$

$$
\begin{array}{c|ccccccccccccc|}
 & & & & & & & g_1(\alpha) & g_2(\alpha) & g_3(\alpha) & \dots & g_1^2(\alpha) & g_1 g_2(\alpha) & \dots \\
1 & 1 & & & & & & -m & a_4 & a_3 & & m^2 & a_4 m & \\
\alpha & & 1 & & & & & p & -5a_5 & -4a_4 & & -2pm & a_4 p + 5a_5 m & \\
\alpha^2 & & & 1 & & & & & & 10a_5 & & p^2 & -5a_5 p & \\
\alpha^3 & & & & 1 & & & & & & & & & \\
\alpha^4 & & & & & 1 & & & & & & & & \\
\alpha^5 & & & & & & 1 & & & & & & & \\
 & & & & & & & N & & & & & & \\
 & & & & & & & & N & & & & & \\
 & & & & & & & & & N & & & & \\
 & & & & & & & & & & & & & \\
 & & & & & & & & & & & N & & \\
 & & & & & & & & & & & & N & \\
\end{array}
$$

- We can explicitly compute upper bounds on variables, s.th. we target vector will be shorter than at least 7 orthogonalized vectors.

Experimental results

- Start with translated version of Kleinjung's polynomial pair $f_i'(x) = f_{KJ_i}(x - \alpha)$
- Goal: Recover the inverse transformation $\alpha' = -\alpha$.

Experimental results

- Start with translated version of Kleinjung's polynomial pair $f_i'(x) = f_{KJ_i}(x - \alpha)$
- Goal: Recover the inverse transformation $\alpha' = -\alpha$.
- Explicit computation of upper bound on α':

$$|\alpha'| \leq N^{0.6}$$

Experimental results

- Start with translated version of Kleinjung's polynomial pair $f'_i(x) = f_{KJ_i}(x - \alpha)$
- Goal: Recover the inverse transformation $\alpha' = -\alpha$.
- Explicit computation of upper bound on α':

$$|\alpha'| \leq N^{0.6}$$

- (Note: Search space of exponential size in polynomial time!)

Experimental results

- Start with translated version of Kleinjung's polynomial pair $f'_i(x) = f_{KJ_i}(x - \alpha)$
- Goal: Recover the inverse transformation $\alpha' = -\alpha$.
- Explicit computation of upper bound on α':

$$|\alpha'| \leq N^{0.6}$$

- (Note: Search space of exponential size in polynomial time!)
- We get enough polynomials, but ...

Experimental results

- Start with translated version of Kleinjung's polynomial pair $f_i'(x) = f_{KJ_i}(x - \alpha)$
- Goal: Recover the inverse transformation $\alpha' = -\alpha$.
- Explicit computation of upper bound on α':

$$|\alpha'| \leq N^{0.6}$$

- (Note: Search space of exponential size in polynomial time!)
- We get enough polynomials, but ...
- Problem: Obtained system of equations is still not 0-dimensional.
- $\Rightarrow$ Does not allow to efficiently recover the root.

Summary

Improving polynomial selection for the GNFS using lattice methods.

Summary

Improving polynomial selection for the GNFS using lattice methods.

- We are able to construct polynomials with good size property.

Summary

Improving polynomial selection for the GNFS using lattice methods.

- We are able to construct polynomials with good size property.
- New definition of norm better resembles polynomial quality.

Summary

Improving polynomial selection for the GNFS using lattice methods.

- We are able to construct polynomials with good size property.
- New definition of norm better resembles polynomial quality.
- It is possible to provide (fixed) roots modulo (fixed) small primes, but no selection by LLL.

Summary

Improving polynomial selection for the GNFS using lattice methods.

- We are able to construct polynomials with good size property.
- New definition of norm better resembles polynomial quality.
- It is possible to provide (fixed) roots modulo (fixed) small primes, but no selection by LLL.
- Enforcing a special Galois group, s.th. the polynomial has a good root property dramatically worsens the size property.

Summary

Improving polynomial selection for the GNFS using lattice methods.

- We are able to construct polynomials with good size property.
- New definition of norm better resembles polynomial quality.
- It is possible to provide (fixed) roots modulo (fixed) small primes, but no selection by LLL.
- Enforcing a special Galois group, s.th. the polynomial has a good root property dramatically worsens the size property.
- Searching for a good root m by means of LLL did not work (yet).