Algebraic (numerical) solvers for certain lattice-related problems

Jintai Ding

University of Cincinnati

Feb. 23, 2012

Outline

1. Introduction
2. Algorithm for BSIS
3. Algorithm for BLWE
4. The REAL motivation and new results

Outline

1 Introduction

2 Algorithm for BSIS

3 Algorithm for BLWE

4 The REAL motivation and new results

- This worked was most done in 2006 while visiting TU Darmsatdt as a Humboldt fellow.
- I would like to thank the people with whom I had useful discussions, in particular,
  Richard Lindner, Alexander May, Ralf-Philipp Weinmann.
- Part of results were presented in IEEE Information Theory Workshop 2011 in Paraty, Brazil
- Part is a joint work with Dieter Schmidt.

Certain shortest vector problem (SVN) in a lattice

The SIS problem.

- Let q be a prime number.
- $A \in \mathbb{Z}_q^{n \times m}$, where A is chosen from a uniform distribution over $\mathbb{Z}_q^{n \times m}$.
- $\Lambda_q^{\perp}(A) = \{\vec{x} \in \mathbb{Z}^m : A\vec{x} \equiv \vec{0} \in \mathbb{Z}^n \pmod{q}\}$ is an m-dimensional lattice.
  The SIS problem is to find a vector $\vec{v} \in \Lambda_q^{\perp}(A)$ with $\|\vec{v}\|_p \leq \beta$.

The lattice

Let $V_1, ..., V_d$ be a basis of $\Lambda_q^{\perp}(A)$ over $\mathbb{F} = F_q$.
The lattice over integer ring is spanned by

$$\begin{pmatrix} V_1^t \\ . \\ . \\ . \\ V_d^t \\ qE_1 \\ . \\ . \\ qE_m \end{pmatrix},$$

where

$$E_i = (0, .., 1, 0, .., 0).$$

d= m-n.

The NTRU lattice

The NTRU lattice is spanned by the rows of matrix in the following form:

$$\begin{pmatrix} I_n & H \\ 0_n & qI_n \end{pmatrix},$$

where I_n is the nxn identity matrix, 0_n is the nxn identity matrix and q is not necessarily a prime number.

The NTRU lattice

The key property here is that there is a nonesero short vector v in the space

$$v = (v_1, ..., v_n),$$

where $|v_i| \leq 1$, namely v_i can only be 1,0,-1, and there should $1/3$ of them to be 1,0,-1 equally.
Namely we have that

$$\|\vec{v}\|_\infty \leq 1 = \beta.$$

The NTRU lattice

To break NTRU can be reformulated as certain type of SIS problem due to Coppersmith and Shamir.

This is the origin motivation of our work in Darmstadt.

The BSIS problem

The BSIS problem.

- $$p = \infty, \quad 2\beta + 1 < q$$

- $$m > \text{or} \approx Q(m - n, D),$$

  where

  $$D = 2\beta + 1 < q,$$

  $$Q(y) = \binom{D}{y + D} = \frac{(y + D)!}{D!(y - 1)!}.$$

The LWE problem

LWE problem can be described as follows.

- a parameter n, a prime modulus q , and an "error" probability distribution κ on the finite field $\mathbb{F}$ with q elements.
- Let $\Pi_{S,\kappa}$ on F_q be the probability distribution obtained by selecting an element A in F_q^n randomly and uniformly, choosing $e \in F_q$ according to κ, and outputting $(A, < A, S > +e)$, where $+$ is the addition that is performed in F_q.
- An algorithm solves LWE with modulus q and error distribution κ, if, for any S in F_q^n , with an arbitrary number of independent samples from $\Pi_{S,\kappa}$, it outputs S (with high probability).

The lattice

The lattice is spanned the rows of

$$A = (\begin{pmatrix} (A_1, -b1) \\ . \\ . \\ . \\ (A_N, -b_n) \end{pmatrix})^t.$$

$$(S, 1) \times A = -E,$$

where

$$E = (r_1, , .., r_N).$$

The BLWE problem

BLWE problem can be described as follows.

- A subclass of the LWE problems, which, we call, the learning with bounded errors (LWBE) problems, namely the errors from the queries do not span the whole finite field but a fixed known subset of size D ($D < q$).

Outline

1 Introduction

2 Algorithm for BSIS

3 Algorithm for BLWE

4 The REAL motivation and new results

How to solve BSIS

3 parameter n, m, a prime modulus q, and a fixed positive integer β and $2\beta + 1 < q$.

- The vector $\vec{X}$ is a short solution to the equation:

$$A\vec{(x)} = 0.$$

where

$$\vec{x} = (x_1, x_2, \cdot, \cdot, \cdot, x_m)^t.$$

- $\vec{v}$ is bounded in the l_∞ norm β implies that

$$\prod_{j=-\beta}^{j=\beta} (x_i - j) = 0. \tag{1}$$

This is a set of m degree D equations with $m - n$ variables.

How to solve BSIS

- By linear substitution, we have a set of
  1) m degree D equations;
  2) m-n variables.
- In general, we can also try to solve this set of equations by Groebner basis solvers, and the complexity can be vastly different.
- If the short vector entries can only be $0, 1$, we have a set of quadratic equations, which therefore relies on the same security assumption as the multivariate public key crytosystems.

How to solve BSIS

- By linear substitution, we have a set of
  1) m degree D equations;
  2) m-n variables.
- If m is sufficiently large in comparison with $m - n$, GB solver becomes solving by linearization.

Outline

1 Introduction

2 Algorithm for BSIS

3 Algorithm for BLWE

4 The REAL motivation and new results

How to solve BLWE

- Let
$$S = (x_1, x_2, \cdot, \cdot, \cdot, x_n).$$
- $$(S, 1) \times (A_i, -b_i)^t = r_i,$$
implies
$$\prod_{k=1}^{D} (\sum a_{i,j} x_j - b_i + e_k) = 0, \tag{2}$$
where $A_i = (a_{i,j})$, $\{e_i\}$ is the set of all possible errors.

How to solve BLWE

- we have
  1) n variables;
  2) N degree D equations.
- In general, we can also try to solve this set of equations by Groebner basis solvers, and the complexity can be vastly different.

How to solve BLWE

- we have
  1) n variables;
  2) N degree D equations.
- N is much large than n, GB solver becomes solving by linearization.

Outline

1 Introduction

2 Algorithm for BSIS

3 Algorithm for BLWE

4 The REAL motivation and new results

The NTRU with message in tenary

- parameters: n, p and q, where p and q have to be relatively prime. q is either a prime number or $q = 2^n$, but here we will concentrate on the case where $q = 2^n$, which is the case suggested for practical applications.
- $p = 3$.

The NTRU with message in tenary

- One works over the ring

$$R = Z_q[x]/(x^n - 1)$$

  with the coefficients defined over the ring (or field) $Z/qZ = Z_q$.

- The polynomials are expressed the form

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

  with the coefficients represented in the symmetric form, that is

$$\lfloor \frac{q-1}{2} \rfloor \leq a_j \leq \lfloor \frac{q}{2} \rfloor \qquad \text{for } j = 0, \ldots, n-1.$$

The NTRU with message in tenary

- We select two "small" polynomials f and g such that a third of the coefficients are chosen to be -1, another third 1, and rest are set to 0.
- f has to be invertible mod q and also mod p, that is, polynomials f_q and f_p have to be found so that

$$f_q \times f = 1 \bmod q$$

and

$$f_p \times f = 1 \bmod p.$$

The NTRU with message in tenary

- To simplify, one usually select instead a small polynomial F and to set then $f = 1 + pF$ so that automatically $f_p = f$. One then computes $h = pf_q \times g \mod q$.
- **The public key** is h.
  The secret key is F and g.

The NTRU with message in binary

- In this case, one selects:

$$p = 2 + x.$$

- F and g can be selected with coefficients either 0 or 1.

The NTRU attack in terms of lattice

- H a nxn circular matrix.
- Find a short vector S such that
  S is short;
  $S \times H(\mathrm{mod}(q))$ is also short.

The NTRU lattice

- Let
$$A = \begin{pmatrix} I_n & H \\ 0_n & qI_n \end{pmatrix}$$
- Find
$$X = (x_1, .., x_{2n})$$
such that $Y = X \times A$ is short.

A new attack

We will look only at the binary case.

- To find the secret key is to factor

$$h(x) = pg(x) \times f_q(x)$$

- Let

$$F(x) = F_0 + F_1 x + \ldots + F_{n-1} x^{n-1},$$

  and

$$g(x) = g_0 + g_1 x + \ldots + g_{n-1} x^{n-1}.$$

  Then we have

$$h(x) \times (1 + pF(x)) = pg(x).$$

- We will derive a set of n linear equations over Z_q in the variables F_i and g_i.

A new attack

- Because the choice of the coefficients is only (1,0) for F_i and g_i, we also have

$$F_i(F_i - 1) = (F_i - 0)(F_i - 1) = 0,$$
$$g_i(g_i - 1) = (g_i - 0)(g_i - 1) = 0.$$

- This means we can solve a set of n linear equations with $2n$ quadratic equations to find the secret key.
- We can solve by GB type of solvers over a ring.
- This is in general very difficult since it is over Z_q.
- This work was done in 2006 in Darmstadt for the tenary case, where we tried to solve degree 3 equations over a finite field.

A new attack

- Let us look again at the set the equation:

$$h(x) \times f(x) = pg(x),$$

over R.

- Let

$$\bar{R} = Z(x)/(x^n - 1).$$

Then the equations above become a set of equations over $\bar{R}$ in the form

$$h(x) + ph(x) \times F(x) = pg(x) + qG(x), \tag{3}$$

where the part $qG(x)$ comes from the modular operation in the original equations, and

$$G(x) = G_0 + G_1 x + \cdots + G_{n-1} x^{n-1}.$$

A new attack

- If we know the statistical range of $G(x)$ then we can build a set of new equations in the form of

$$\prod_{i=1}^{d}(G_j - a_i) = 0,$$

so that the coefficients G_j are now restricted to the set of integers $a_1, \ldots, a_d$.

- When we represent elements in Z_q, we represent them in the form

$$-q/2+1, \ldots, 0, \ldots, q/2.$$

Therefore the range of $G(x)$ should not be large statistically.

A new attack

Numerical experiment results.

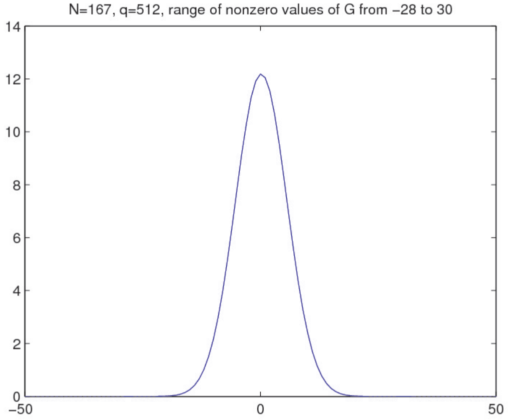

Figure: Frequency of nonzero values for coefficients G_i

The REAL attack

- We can find $F(x)$ and $g(x)$ by solving a set of equations consisting of
  n linear equations;
  $2n$ quadratic equations;
  n degree d over Z.
- **We can use Newton type of numerical method to solve this set of equations.**
- We could first do substitutions from the linear equations. If we substitute all the variables F_i, we would have a set of equations consisting of $2n$ quadratic equations and n degree d over Z.

The "REAL" attack

- We will now work with the assumption that $|G_i| \leq d$ for $i = 0, \ldots, n-1$.
- With the given public key

$$h(x) = h_0 + h_1 x + \cdots + h_{n-1} x^{n-1}$$

and knowing that the function $f(x)$ has the form $f(x) = 1 + (2 + x)F(x)$, we have

$$p(x) \times h(x) \times F(x) - p(x) \times g(x) - qG(x) + h(x) = 0 \quad (4)$$

- This gives a system of linear equations in the form $Ay + c = 0$ with the unknown vector of coefficients given by

$$y = (F_0, F_1, \ldots, F_{n-1}, g_0, g_1, \ldots, g_{n-1}, G_0, G_1, \ldots, G_{n-1})^t$$

and

$$c = (h_0, h_1, \ldots, h_{n-1})^t.$$

The REAL attack

- The system of equations to be solved is therefore

$$\begin{aligned} Ay + c &= 0 \\ F_i(F_i - 1) &= 0 \\ g_i(g_i - 1) &= 0 \\ G_i \prod_{j=1}^{d} (G_i - j)(G_i + j) &= 0. \end{aligned}$$

The REAL attack

- We can use numerical routines to solve the system of nonlinear equations.
- MATLAB provides such a routine and it is called `fsolve`. It requires two parameters, a reference to the function and an initial guess.
- The optional parameter allows for the setting of various options, for example the tolerance in the function evaluations and/or the tolerance in the x values, before MATLAB decides that a solution has been found.
- Another parameter is the number of function evaluations and it can be set to a large value, since in our case the given functions are easy to evaluate.

The "REAL" attack

- If the Hessian of the system of equations can be computed explicitly, which is true in our case, this can also be given as a parameter instead of asking MATLAB to find the Hessian numerically.
- Options can also be set on how much MATLAB should report on the progress of finding a root.
- The underlying method uses the sum of the squares of the equations and then tries to minimize this function.
- The Levenberg–Marquardt algorithm is used but the faster but less robust Gauss–Newton algorithm can also be selected.

The "REAL" attack

- As expected the choice of the initial guess places a significant role in finding the solution, since the function to be minimized has lots of local minima.
- Even for this small example finding the correct solution is not guaranteed unless one starts within a reasonable distance from the actual solution.
- Starting with the zero vector as initial guess, `fsolve` usually returns with an answer which is closer to zero than the actual solution.
- Similar things will happen when we start with a vector of all elements set to one, except that after rounding to the nearest integers not all elements will be one.
- It is clear that we have to find a better initial condition. The LLL?

A simplified "REAL" attack

- We have investigated what happens when some of the parameters of variables are known in advance.
- For example assume that all of the G_i's are known in advance.
- In this case the values for the F_i's are usually close to the actual solution, so that after rounding them to the nearest integer (0 or 1) we obtain the correct values for the F_i's.

A simplified "REAL" attack

Examples

- In the first example, we use $n = 167$ and $q = 512$ and start with an initial vector for y with the elements set to zero or one at random.
  The running time was 0.7 seconds.
- When looking at the coefficients found for F most of them are within 10^{-6} of their actual values so that rounding them gives the correct answer.

A simplified "REAL" attack

Examples

- For the second example, we use $n = 809$ and $q = 2048$.
- We have chosen the zero vector as the starting point, which usually speeds up the convergence to a solution.
- MATLAB completed in 44.9 seconds.
- Despite the claim of MATLAB that it did not converge to a root, the coefficients of F in the solution vector are again close enough to their correct integer values. After rounding them we can find the values for the g_i's.

A simplified "REAL" attack

- With this approach we are able to recover the coefficients F_i and g_i even for large systems starting with the zero vector as the initial guess.
- The numerical routine `fsolve` is surprisingly fast and it might even work for larger systems.
- The only problem seems to be that it uses single precision in order to preserve memory, and thus has its own limitations.

Equations over real

- Using Newton method directly is not sufficient.
- The modular part very important.

What we can do and what next

- Better approximation on G?
- Better solver?
- New direction – LLL with Newton

Acknowledgment

Thank you and any question?