Algebraic solvers for certain lattice-related problems

Jintai Ding
Southern China University of technology
University of Cincinnati
Email: Jintai.Ding@gmail.com

***Abstract*—In this paper, we present a new algorithm to solve algebraically the following lattice-related problems:**

1) the small integer solution (SIS) problem under the condition: if the solution is bounded by an integer β in l_∞ norm, which we call a bounded SIS (BSIS) problem, and if the difference between the row dimension n and the column dimension m of the corresponding basis matrix is relatively small with respect the row dimension m;

2) the learning with errors (LWE) problems under the condition: if the errors are bounded – the errors do not span the whole prime finite field F_q but a fixed known subset of size D ($D < q$), which we call a learning with bounded errors (LWBE) problem.

We will show that we can solve these problems with polynomial complexity.

I. Introduction

Recently, lattice-based cryptosystems have attracted a lot of attentions. One well known problem in the area is the Small Integer Solution (SIS) problem, which is closely related to the provable security property of certain class of problems with average-case to worst-case reduction. There are arguments that the SIS problem is very hard due to its connection with the worst-case lattice problems such as SIVP (the shortest independent vectors problem).

SIS problem can be described as follows.

Let q be a prime number and 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 is to find a vector $\vec{v} \in \Lambda_q^\perp(A)$ with $\|\vec{v}\|_p \leq \beta$.

Here, we have key parameters n, m q, p and β.

In this paper, we will first present a new algorithm to solve a subclass of the SIS problems, which, we call, a bounded SIS (BSIS) problems, namely the case where

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

and each component of the solution vector $\vec{v}$ are all bounded by the integer β (l_∞ norm), and if

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

where

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

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

Here $Q(y)$ is the number of monomial (including 1) in the polynomial ring $F_q[x_1, ..., x_y]$ when D is less than q. Therefore the number of monomials (excluding 1) is exactly $Q(y) - 1$.

Note here that a vector is l_∞ bounded does not at all mean that it is l_p bounded for a fixed p and $p \neq \infty$.

We show that we can solve this problem with polynomial complexity in $O(m^3)$.

We would remark here that our algorithm also works in the case where the solution component are not small integers but rather integers in a fixed small subset of size β. Clearly, here we exclude the case q = 2.

Another problem we will study is the Learning with Errors (LWE) problem, introduced by Regev in 2005 [4]. It is a problem closely related to the SIS problem. It is also used in cryptographic constructions with some good provable secure properties and the main claim is that it is as hard as certain worst-case lattice problems.

LWE problem can be described as follows.

First, we have a parameter n, a prime modulus q , and an "error" probability distribution κ on the finite field F_q 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).

In the case q = 2, this problem corresponds to the learning parity with noise (LPN) problem.

There are several ways to solve this family of problems. One naive way to solve LWE is through a maximum likelihood algorithm by directly solving about O(n) equations. This leads to an algorithm that uses only O(n) samples, and runs in time $2^{O(nlogn)}$. There are other similar algorithms with similar complexity. A more sophisticated algorithm is developed by Blum, Kalai, and Wasserman [2] , and it requires $2^{O(n)}$ samples and time. This algorithm is based on the method to find a special small set of equations among $2^{O(n)}$ equations to solve the problem. The Blum et al. algorithm is the best known algorithm for the LWE problem in general, which is related to the fact that the best known algorithms for lattice problems require $2^{O(n)}$ time.

On the theory side, there are again arguments that the LWE problem is very hard due to the complexity of current

978-1-4577-0437-6/11/$26.00 ©2011 IEEE

algorithms and due to the connection of LWE problems with various known hard problems such as the LPN problem, and the worst-case lattice problems including GAPSVP (the decision version of the shortest vector problem) and SIVP (the shortest independent vectors problem). It is even considered to be a hard problem for quantum computers.

We will present a similar new algorithm to solve 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 ($D < q$). We show that we can solved this problem with computation complexity $O(n^{3D})$ and $O(n^D)$ queries.

This paper is organized as follows. In the next section, we will present the algorithm for the BSIS problem and its complexity analysis. In the third section, we will present the algorithm for the BLWE problem and its complexity analysis. We will present the conclusion in the end.

II. The new algorithm for the BSIS problem

Let us first define BSIS problem precisely.

A. The BSIS

BSIS problem is given as follows.

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

The BSIS problem is to find a vector $\vec{v} \in \Lambda_q^{\perp}(A)$ with $\|\vec{v}\|_\infty \leq \beta$, where A $\in \mathbb{Z}_q^{n\times m}$, is chosen from a uniform distribution over $\mathbb{Z}_q^{n\times m}$, and

$$\Lambda_q^{\perp}(A) = \{\vec{x} \in \mathbb{Z}^m : A\vec{x} \equiv \vec{0} \in \mathbb{Z}^n \pmod q\},$$

which can be viewed as an (m-n)-dimensional lattice.

Here, let us add our first additional assumption that the matrix A is row independent, if not, we can always use a Gaussian to make the rows linearly independent.

From now on, let us assume that $n < m$.

Also for the simplicity, the problem we will look into is a subclass of the BSIS problems, namely we require that $n-m$ is relatively small which is defined as:

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

In the case $D = 2$, this means that

$$m - n \approx 2\sqrt{m};$$

and in the case of $D = 3$,

$$m - n \approx 6\sqrt[3]{m}.$$

B. The new algorithm

Clearly the vector $\vec{v}$ is a short solution to the equation:

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

where

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

Since we know that $\vec{v}$ is bounded in the l_∞ norm β. Then we immediately have that

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

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

Then we also have a set of linear equations

$$AX = 0,$$

which is also a set of linearly independent equations.

For this set of linear equations, we can perform Gaussian elimination, and then we substitute these equations into the set of m degree D equations.

Then we have a set of m degree D equations with only now $m - n$ variables.

1) If $m > Q(m-n, D)$, then we can perform a standard linearization, namely we treat each monomial as an independent variable and by solving a set of linear equations, we should be able to find the solution we are looking for, and the complexity is roughly $O(m^3)$. If it does not work , we will do as in the case below, namely increase the degree by 1 using partial enlargement.
2) If $m \approx Q(m-n, D)$, then we can perform a partial enlargement as in [3], to produce a larger set of more that $Q(m-n, D+1)$ degree less or equal to $D+1$ equations by multiplying each of equations by monomials of degree 1, then we can solve it by linearization, and the complexity is roughly $O(m^{3+3/D})$.

By linearization, we mean that we assign each monomial of $x_1, .., x_{m-n}$ (not including 1) a new variable y_i and the number of monomial is exactly $Q-1$ and we assign x_i to be $y_{Q-n+i-1}$.

Then this linearization will produce a new set of linear equations in the form of

$$L \times Y = B,$$

where

$$Y = (y_1, y_2, \cdot, \cdot, \cdot, y_{Q-1})^t$$

and L is a $Q' \times (Q-1)$ matrix and B a constant vector, where Q' is the number of equations we have.

We then solve the linear equation

$$LY = B$$

and the output $(y_{Q-(m-n)-1}, ..., y_{Q-1})$ gives the solution $\vec{v}$ and we end the algorithm.

C. Analysis of the Algorithm and Complexity

One can see easily that the success of the algorithm depends on if we can solve the linear equation:

$$LY = B,$$

which comes from the linearization of the set of m polynomial equations of degree D with $m-n$ variables.

Since the entries of the matrices follows certain almost uniformly distribution, it is not unreasonable to assume that the matrix from those m nonlinear equations is quite generic and its coefficients are somewhat randomly and uniformly distributed. In this case, it is not at all difficult to deduce that we have a very good probability to succeed at degree D or for sure we will succeed in degree $D+1$ in the linearization solving step.

In all the extensive experiments, (with relative large q and $D=3$), we have never failed. Therefore the conclusion is that the new algorithm works nearly 100 percent correctly.

Now let us look at the complexity. It is clear that the matrix size of the linearization step is either m, if we solve at degree D or the size of roughly $m\times \sqrt[D]{m}$, if we solve at degree $D+1$. Then the complexity should be either $O(m^3)$ or $O(m^{3+3/D})$.

Therefore the complexity is for sure polynomial in m.

On the other hand, surely the biggest memory requirement is to store the matrix associate with linearization, which is of the size roughly $O(m^2)$.

Then one may ask about the case when $m-n$ is large, for example, in the case of NTRU, where the associated SIS problem is the case $m=2n$. In such a case, we can use some of the polynomial solving algorithms such as Groebner basis or MXL algorithms [3], the complexity is surely expected to be much higher and we do not expect to solve such a problem easily using directly our algorithm. The details will be discussed further a subsequent paper.

Another remark we have is that our algorithm is designed to solve a subclass of the SIS problem, but we can easily transform it into an algorithm of a generic short vector problem, where the short vector is bounded by a l_∞ norm of size β, and the conclusion, we can draw here, is that the solver does not really depend on the distribution of component of the short vector. Similarly we can conclude here that if the dimension of the lattice is relative small compared with the total dimension of the space and it is l_∞ bounded, then this is an easy problem just like the problem we solve in this paper, since they are equivalent.

III. The new algorithm for the BLWE problem

Let us first define LWBE problem.

A. The LWBE

LWBE problem is given as follows.

There are a parameter n, a prime modulus q, and a bounded "error" probability distribution κ on the finite field F_q with q elements such that there are only D (and $D<q$) elements whose distribution probability from κ is not zero while the rest are all zero.

Let $ES=\{e_1,..,e_D\}$ be the set of elements whose probability in the distribution κ is not zero and we call this set the error set. This set could include the zero element in F_q and not necessarily.

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 additions are performed in F_q.

An algorithm that solves LWBE 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).

Surely we first conclude this problem excludes the case that $q=2$, since the error can only be 1.

The main motivation to consider this problem comes from the consideration that often in the lattice related problems, the short vector (or the 'error') are often select mainly from the small set $\{1,-1\}$ as in the NTRU case.

B. The new algorithm

Let

$$S=(x_1,x_2,\cdot,\cdot,\cdot,x_n)^t.$$

Let

$$Q=\binom{D}{n+D}=\frac{(n+D)!}{D!(n-1)!},$$

which is the number of monomial (including 1) in the polynomial ring $F_q[x_1,...,x_n]$ when D is less than q. Therefore the number of monomials (excluding 1) is exactly $Q-1$.

For a fixed D, clearly Q is of the class $O(n^D)$.

1) **Step 1. Queries**
We will make $Q'=Q+O(n)$ queries.
For the i-th query, we shall derive a linear equation that

$$\sum a_{i,j}x_j=b_i,$$

where b_i carries the errors. Therefore it is only probabilistically true.
For each such linear equation, we will produce the degree D equation:

$$\prod_{k=1}^{D}(\sum a_{i,j}x_j-b_i+e_k)=0. \tag{2}$$

We collect those degree D equations to form a new set we call C.

Note here that D needs to be less than q, otherwise the equation above will be totally trivial, namely the so-called field equations:

$$x_i^q=x_i.$$

2) **Step 2. Linearization**
We linearize the set of equations C such that we assign each monomial of $x_1,..,x_n$ (not including 1) a new variable y_i and the number of monomial is exactly $Q-1$ and we assign x_i to be $y_{Q-n+i-1}$.

Then this linearization will produce a new set of linear equations in the form of

$$L \times Y = B,$$

where

$$Y = (y_1, y_2, \cdot, \cdot, \cdot, y_{Q-1})^t$$

and L is a $Q' \times (Q-1)$ matrix and B a constant vector

3) **Step 3. Solving** the linear equation

$$LY = B$$

and the output $(y_{Q-n-1}, ..., y_{Q-1})$ gives the solution S and we end the algorithm.

Note here that we have more rows than columns in L.

If we can not find the solution (there are too many depend equations), we make another R (of size $O(n)$) queries to derive a new set of linear equations

$$\sum a'_{i,j} x_j = b'_i,$$

and produce another R degree D equation:

$$\prod_{k=1}^{D} (\sum a'_{i,j} x_j - b_i - e_k) = 0. \quad (3)$$

Then we amend these equation to C and then go to Step 2 again.

The reason that Step 2 works is very obvious since we know that one of the linear factors of the polynomial $\prod_{k=1}^{D}(a_{i,j}x_j - b_i + e_k)$ *must be zero since it covers all possible errors.*

The key point is that the degree D equations in C are precise equations and therefore 100 percent correct unlike the linear equations. This fundamental idea behind this method is the same as that is used in [1], namely to use interpolation formula to even out the noise to derive a set of precise equations

IV. A TOY EXAMPLE

We will do a example over $GF(5)$.

Let n=2.

Let us assume that our error set is $ES = (0, 1)$ and in this case $D = 2$. This means the error e can only be 1 (or 0 – no error).

We also have

$$Q = C(\binom{D+n}{D}) = C(\binom{2}{2+2}) = 6.$$

In this case, let us assume that we make 6 queries and the query vectors are randomly selected as

$$\begin{pmatrix} (1,1) \\ (3,2) \\ (-1,3) \\ (1,-1) \\ (2-1) \\ (3,-1) \end{pmatrix}$$

Then query results are given as

$$W = (1, 2, 2, 0, 1, 3)^t.$$

The corresponding probabilistic linear equations can be written as the set:

$$\begin{pmatrix} (x_1 + x_2 - 1) = 0 \\ (3x_1 + 2x_2 - 2) = 0 \\ (-x_1 + 3x_2 - 2) = 0 \\ (x_1 - x_2) = 0 \\ (2x_1 - x_2 - 1) = 0 \\ (3x_1 - x_2 - 3) = 0, \end{pmatrix}$$

which are the probabilistically true.

From this, because the error set is $\{0, 1\}$, we can derive the corresponding quadratic (d=2) equations as:

$$\begin{pmatrix} (x_1 + x_2 - 1)(x_1 + x_2) = 0 \\ (3x_1 + 2x_2 - 2)(3x_1 + 2x_2 - 1) = 0 \\ (-x_1 + 3x_2 - 2)(-x_1 + 2x_2 - 1) = 0 \\ (x_1 - x_2)(x_1 - x_2 + 1) = 0 \\ (2x_1 - x_2 - 1)(2x_1 - x_2) = 0 \\ (3x_1 - x_2 - 3)(3x_1 - x_2 - 2) = 0, \end{pmatrix}$$

which are 100 percent true.

Now we assign the linearization variables as:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_5 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_1 * x_2 \\ x_2^2 \\ x_1 \\ x_2 \end{pmatrix}$$

Then we derive the linear equation:

$$L \times Y = B$$

$$\begin{pmatrix} 1 & 2 & 1 & 4 & 4 \\ 4 & 2 & 4 & 1 & 4 \\ 1 & 0 & 1 & 3 & 3 \\ 1 & 3 & 1 & 1 & 4 \\ 4 & 1 & 1 & 3 & 1 \\ 4 & 4 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

This gives us the solution that:

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_5 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 4 \\ 2 \\ 3 \end{pmatrix}.$$

From this we derive that

$$\begin{pmatrix} y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Therefore

$$S = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Therefore the correct query result should be

$$\bar{W} = (0, 2, 2, -1, 1, 3)^t.$$

The error vector then is given as

$$\bar{E} = (1, 0, 0, 1, 0, 0)^t,$$

namely only the first and the third queries carry errors and the rest are correct.

A. Analysis of the Algorithm and Complexity

One can see easily that the success of the algorithm depends on if we can solve the linear equation:

$$L \times Y = B.$$

Since the query vectors A_i are randomly and uniformly chosen, it is not unreasonable to assume that coefficients of the matrix L are somewhat randomly and uniformly distributed. In this case, it is not at all difficult to deduce that the random matrix L has a very high probability (roughly $1 - 1/q$)) to be of rank $Q - 1$ and therefore we can derive a solution. In all the extensive experiments (thousands and $q > 3$), we have never failed in the first round of our algorithm. Therefore the conclusion is that algorithm works nearly 100 percent.

In the case of toy example in section above, if we select from the 6 queries we have 5 queries, which is minimum we need, we can see that among all 6 choices, all but one also are sufficient to solve the problem. The only one that does not work is the case we choose the queries 1,2,4,5,6. This confirms our argument above.

Now let us look at the complexity. It is clear that the matrix size of L if roughly

$$(n^D/D!)^2$$

and therefore the complexity of solving $LY = B$ is roughly

$$n^{3D}/(6 \times D!).$$

Therefore, we conclude that for any fixed D, we have polynomial time solver in terms of n.

On the other hand, surely the biggest memory requirement is to store the matrix L, which is of the size

$$(n^D/D!)^2,$$

and could be a serious problem if D and n is large. However in this case, we can use some of the polynomial solving algorithms such as MXL algorithms [3] to make fewer queries but using more time to solve it, which we are now working on.

Another remark we have is that our algorithm does not really depend on the distribution of the errors on the error set.

V. Conclusion

In this paper, we first present a new algorithm to solve a subclass of the small integer solution (SIS) problem, if the solution is bounded by an integer β in l_∞ norm, which we call a bounded SIS (BSIS) problem, and and if the difference between the row dimension and the column dimension of the corresponding basis matrix is relatively small with respect the row dimension, the complexity is polynomial in m, the dimension of the solution vector. In addition, we also present a new algorithm to solve the learning with bounded errors (LWBE) problems, whose errors are bounded – the errors do not span the whose finite field but a fixed know subset of size D, with complexity $O(n^D)$.

This algorithm, we believe, present a new direction to look at the security of the cryptographic algorithms that are related to the SIS problem, in particular the NTRU cryptosystems and the SWIFT family of algorithms and the security of the cryptographic algorithms that are related to the LWE problem. We hope that we can develop new attack tools along this line to really enhance exiting attacks. One particular interesting direction is to use more sophisticated algebraic solvers like in [3] to enhance our algorithms.

Acknowledgment

The algebraic solver idea of this paper, in particular, the use of polynomial equations (1) and (2), was first discovered by the author in attacking certain family of lattice-based cryptosystems in 2006-2007 when the author was a Humboldt fellow in TU Darmstdat. We would like to thank R. Linderner R. Weimann, A. May for stimulating discussions. We would like to thank the Humboldt foundation for the support in Germany. We would also like to thank D. Cabarcas for performing experiments.

This work is partially supported by NSF China grant 60973131 and the Taft foundation at the University of Cincinnati.

References

[1] Sanjeev Arora, Rong Ge, Learning Parities with Structured Noise, TR10-066, April 2010

[2] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM, 50(4):506519, 2003.

[3] Johannes Buchmann, Daniel Cabarcas, Jintai Ding, Mohamed Saied Emam Mohamed: Flexible Partial Enlargement to Accelerate Grbner Basis Computation over F2. AFRICACRYPT 2010: 69-81

[4] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6):34, 2009. Preliminary version in STOC05.

[5] Oded Regev, The Learning with Errors Problem (Invited Survey), CCC, pp.191-204, 2010 25th Annual IEEE Conference on Computational Complexity, 2010