Cryptanalysis of LWE

Notes by Henry Corrigan-Gibbs

MIT - 6.5610
Lecture 20 (April 24, 2024)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Linearization (Arora-Ge): An algebraic attack
- BKW: A combinatorial attack
- LLL: A geometric attack

I highly recommend Vinod's lecture notes on lattices. The discussion here is heavily influenced by his notes.

Introduction

Today I will give you a quick and dirty summary of the known *classical* attacks on the learning-with-errors problem. We will focus on algorithms for the "decision" version of the problem, which is the one that we have discussed up to this point in the class.

Some algorithms we will see actually solve the "search" version of the problem, in which the attacker must find the LWE secret. The search and decision versions of the problem are equally hard, up to polynomial factors.

The goal of this lecture is just to show you the breadth of ideas that we use in cryptanalysis—each drawing on a different mathematical idea. The strange thing about cryptanalysis is that the same ideas come up all over the place. Each of the three ideas I will present today are also useful for breaking symmetric-key cryptosystems, which have no apparent connection to LWE. Knowing these handful of techniques is already enough to do lots of damage.

There is no one "best" attack on LWE: which attack works best is a function of the particular choice of LWE parameters you are using. I will try to give some intuition as we go for why that is.

The take-home message for you should be that: for a given choice of the security parameter n (i.e., the dimension of the secret), what determines the hardness of LWE is the "noise-to-modulus" ratio: how large the LWE error is relative to the modulus.

Often people call this the "modulus-to-noise" ratio, which is just the inverse.

To refresh your memory, the LWE assumption of Regev, loosely speaking, asserts that it is hard to solve noisy linear equations modulo integers of a certain type.

The LWE assumption is a family of assumptions associated with integral parameters $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$ and an error distribution $\chi = \chi(\lambda)$ over $\mathbb{Z}_q$.

Definition 1. The $\mathrm{LWE}_{n,m,q,\chi}$ assumption asserts that

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \xleftarrow{R} \mathbf{Z}_q^m$.

We will think of the distribution χ as being B-bounded—that is, its support is over $\{-B, \ldots, B\}$, with our standard interpretation of negative numbers in $\mathbb{Z}_q$. By default, we will take χ to be the uniform distribution over $\{-B, \ldots, B\}$.

Furthermore, we will take $q = \mathrm{poly}(n)$. Similar ideas in the more general case of a super-polynomially large modulus q with more work.

An algebraic attack on LWE

This attack when the LWE error is sampled from a very narrow distribution—that is, when $B = O(1)$ or similar. In this case, it can recover the entire LWE secret in polynomial time, provided that the number of samples that the attacker has access to is $m = n^{\Omega}(B)$.

The attack uses "linearization," which you saw in one of the problem sets. This linearization trick is a good one to know, since it comes up in a few places in cryptanalysis. The basic idea is that you can (sometimes) solve a system of non-linear equations if you have enough equations. The idea is to replace each non-linear monomial, such as x^2y, with a single new variable z_1. Then you can replace the second non-linear monomial with a second variable z_2, and so on, until you have a linear system in $z_1, z_2, \ldots$. If you have more independent linear equations then variables, you can solve the system.

The trick is showing that (1) you have enough independent linear equations and (2) that the solution that you get to the linear system is actually a valid solution to the original system.

As an example of where things can go wrong: If you have

$$x^2y = 12$$
$$xy = 6,$$

you can linearize the system and you will have two equations and two unknowns. But the resulting trivial equations will not help you solve the system.

Attacking LWE using linearization. This attack is due to Arora and Ge [1].

Say that $B = 1$, so that the LWE errors are in $\{-1, 0, 1\}$. If we have a noisy LWE sample $(\mathbf{a}, b) = (\mathbf{a}, \mathbf{a}^\intercal \mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, we know that either:

When I say an "LWE sample," I am referring to a row of the $\mathbf{A}$ matrix and it's corresponding entry in the $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ vector.

- $b = \mathbf{a}^\intercal \mathbf{s} - 1 \quad \Rightarrow \quad 0 = \mathbf{a}^\intercal \mathbf{s} - b - 1$,
- $b = \mathbf{a}^\intercal \mathbf{s} - 0 \quad \Rightarrow \quad 0 = \mathbf{a}^\intercal \mathbf{s} - b$, or
- $b = \mathbf{a}^\intercal \mathbf{s} + 1 \quad \Rightarrow \quad 0 = \mathbf{a}^\intercal \mathbf{s} - b + 1$,

where everything is in $\mathbb{Z}_q$.

Multiplying these equations together, we get an equation of total degree three in the elements of the LWE secret $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$:

The *total degree* is just the maximum, over all monomials, of the sum of the exponents of the monomial.

$$0 = (\mathbf{a}^\intercal \mathbf{s} - b - 1)(\mathbf{a}^\intercal \mathbf{s} - b)(\mathbf{a}^\intercal \mathbf{s} - b + 1).$$

Since $\mathbf{a}$ and b are just constants (i.e., known to the attacker), the only variables here are the s_is. This equation has monomials of the form s_i, $s_i s_j$, and $s_i s_j s_k$.

The linearization attack replaces each of these monomials with a variable z_i, and then solve the resulting linear system using Gaussian elimination. Doing so requires at least $\approx n^3$ equations. It takes quite a bit of care to argue that the resulting system of equations is linearly independent (with high probability, over the choice of the $\mathbf{A}$ matrix) and that the solution that Gaussian elimination gives back is a solution to the original system.

So, if the error is small and the number of samples is large, this attack is devastating—a polynomial-time secret-recovery attack.

The attack only works if the error is very small. As soon as $B = \omega(1)$, the running time of the attack becomes $n^{\omega(1)}$, which is super-polynomial in n.

This attack may still be better concretely than the other attacks we will discuss. It just no longer runs in polynomial time.

Moreover, if the attacker only gets to see a small number of samples, the attack does not work either. For example, if the attacker gets only $m = 10n$ samples and the error is in $\{-1, 0, 1\}$, it is not clear how to make this attack work.

Another surprise is that the same trick does not seem to work if the LWE *secret* is small. Even if the LWE secret is very small—0/1, for example—this small-secret flavor of LWE is roughly as hard as the standard version. So the hardness of the problem is not symmetric in the secret and error distribution.

BKW attack

This attack is due to Blum, Kalai, and Wasserman [2]. The BKW attack requires a large amount of time and samples: roughly $2^{O(n)}$.

At the same time, this subexponential running time is much better than the $2^{O(n \log n)} = n^{O(n)}$ brute-force guess-the-secret attack.

Write the LWE problem instance as a pair $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$. We are looking for the secret $\mathbf{s}$.

Warm up: If we get lucky...

Now, say that we (the attacker) got really lucky and one of the rows of $\mathbf{A}$ had the form $(1\ 0\ 0\ 0\ \cdots 0) \in \mathbb{Z}_q^n$. Then I claim that we can get a noisy version of the first component of the secret.

In particular, we would have an LWE sample of the form

$$(\mathbf{a}, \mathbf{a}^\mathsf{T}\mathbf{s} + e) = (1\ 0\ 0\ 0\ \cdots 0)^\mathsf{T}\mathbf{s} + e,$$

which gives us $s_1 + e \in \mathbb{Z}_q$ for some error $e \xleftarrow{R} \{-B, \ldots, B\}$, where $s_1 \in \mathbb{Z}_q$ is the first component of the secret.

Having a noisy version of s_1 is not good enough: we need s_1 itself. But to do that, we can just hope to get many samples of this form and then average them.

If we are able to repeat this entire process T times and sum the results, we will expect the error terms to add up to something like $B\sqrt{T}$ over the integers. The average of these errors values is then $B/\sqrt{T}$. To make this error $< 1/2$, we just need $T = \Omega(B^2)$. Once the error is $< 1/2$, rounding the average of the samples will give us back s_1. We repeat for all n components of the secret to recover it all in time roughly $nB^2 = \text{poly}(n)$, ignoring $\log n$ factors.

I'm being very sloppy here...
We need $B\sqrt{T} \ll q$ or else correctness fails.

What's the catch? Well, the probability that we are so lucky as to get an LWE sample with the special form $(1\ 0\ 0\ 0 \cdots 0)$ is vanishingly small: roughly $1/q^n$. So this attack really requires something like q^n samples, which is too many.

Rescuing the idea. At the highest level, the basic idea of the BKW algorithm is to find ℓ LWE samples that add up to the standard-basis vector $(1\ 0\ 0\ 0\ \cdots 0)$. The total error of the summed up samples will be at most ℓB.

For correctness to hold, we need to make sure that the error ℓB does not wrap around the modulus q, which requires that $\ell B \ll q$, or $\ell \ll q/B$. Taking the average of $\text{poly}(\ell, B)$ of these samples gives us back the first element of the secret. Then we repeat this entire process n times to recover the entire secret.

Here is one place where the modulus-to-noise ratio shows up.

The trick then is how you find a small subset of the rows of the $\mathbf{A}$ matrix that sum up to $(1\ 0\ \cdots\ 0)$.

I won't try to describe the full idea, but I will try to give you a taste of what is going on.

Finding weight-two combinations.

Let's say that we have a large collection of LWE samples and we are looking for two that sum to the "target" vector $(1\ 0\ 0 \cdots 0)$. How hard is this?

This is actually just the birthday problem in disguise: Take two lists of m LWE samples. Sum up sample i from the left list and sample j from the right list. What is the probability that they sum to the target vector?

If you sum up two random vectors in $\mathbb{Z}_q^n$, the probability that they sum to zero is $1/q^n$. There are m^2 pairs of vectors in the two lists, so heuristically (and we can make this formal), the probability of finding two that sum to our target vector is $\sqrt{q^n} = q^{n/2}$. Using hashing, we can find this pair in time $O(m) = q^{n/2}$. This is not a great running time, but still beats the q^n brute-force search.

For whatever reason, I find it easier to think of the BKW algorithm in the language of Wagner's generalized birthday problem [3]. A neat property of the BKW/Wagner algorithm is that it has many applications to symmetric-key cryptanalysis as well. Take a look at Wagner's paper for details.

Finding weight-four combinations

Let's look at how you can do a little bit better than birthday. In particular, we will show how to find size-four subsets of the LWE samples that add up to the target vector. We will do this in time $q^{n/3}$, which is again a big improvement over the standard birthday attack's $q^{n/2}$.

The BKW algorithm takes this trick "all the way" to get the full algorithm.

The birthday algorithm works by finding colliding vectors. Here we will find partial collisions and then collisions with collisions.

Step 1: Partial collisions In particular, we will start out by looking for many pairs of vectors whose first $n/3$ components sum to our target vector:

$$(\underbrace{1\ 0\ \cdots\ 0}_{n/3} \,\|\, \underbrace{\text{random garbage}}_{2n/3}).$$

How hard is it to find such partial collisions? A pair of random vectors sums to the target vector with probability $q^{-n/3}$, so if we have m vectors, the number of partial-collision pairs we will have is roughly $m^2 q^{-n/3}$.

If we take the list size/sample complexity $m = q^{n/3}$, then we will expect to end up with $q^{n/3}$ partial collisions of this type.

Step 2: Full collisions Given $q^{n/3}$ partial collisions, we now want to find complete collisions. What is the probability that two of the partial-collision vectors sum up to a full collision?

The probability of a success with any two is $q^{-2n/3}$. But since we have $m = q^{n/3}$ partial collisions, the probability of finding a good pair is $m^2 q^{-2n/3}$, which is some constant.

Moreover, each of the full collisions we have found is just the sum of four original samples.

The full BKW algorithm finds sums of roughly q/B samples that match the target vector.

When ℓ elements that sum to the target vector, the BKW algorithm runs in time roughly $q^{n/\log \ell}$, which gives $q^{n/\log(q/B)}$ for LWE.

When $q/B = \text{poly}(n)$ the running time is $q^{n/\log n}$. In some applications, we want $q/B = 2^{\sqrt{n}}$ or something. In this case, the running time of BKW is much better: $q^{\sqrt{n}}$ or so.

As the modulus-to-noise ratio size grows then, the security parameter n has to grow as well to keep the same level of LWE hardness.

Lattice-basis reduction

The last class of attacks we will discuss uses the LLL algorithm of Lenstra, Lenstra, and Lovász. I am going to be even more hand-wavy here than I was with the other two attacks.

I'll describe this as an algorithm for the decision problem.

The LLL algorithm is a polynomial-time algorithm with the following API:

- **Input:** A matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$.
- **Output:** A non-zero vector $\mathbf{x} \in \mathbb{Z}^n$ such $\|\mathbf{Ax}\|_2 \in \mathbb{Z}^m$ is "small."

Notice that this matrix is over the integers, not modulo q.

The promise of the algorithm is that it returns a vector $\mathbf{x}$ such that $\|\mathbf{Ax}\|_2$ is within a factor of 2^m (I am being VERY imprecise here) of $\min_{\mathbf{x} \in \mathbb{Z}^n} \|\mathbf{Ax}\|_2$.

If we look at the matrix

$$\mathbf{A}' = \left(\mathbf{A} \| \mathbf{b} \| q\mathbf{I}\right),$$

we know that there is an $\mathbf{x}$ such that $\|\mathbf{A}'\mathbf{x}\|_2$. That vector is $\mathbf{x}^* = (\mathbf{s}\| - 1 \| *)$, for which we have $\|\mathbf{A}'\mathbf{x}^*\|_2 \approx Bm$.

In contrast, if we are looking at a random non-LWE instance $\mathbf{A}' = (\mathbf{A}\|\mathbf{b}\|q\mathbf{I})$, we will have $\min_{\mathbf{x}} \|\mathbf{A}'\mathbf{x}\|_2 \approx q^{n/m}$.

If $\mathbf{A}'$ were not structured (i.e., not an LWE instance), we would expect the shortest vector to have size $q^{(m-n)/m}$, via an argument that we won't look at. So if LLL returns a very short vector, we will know we have an LWE instance. If not, then not.

We will expect to see a "gap" whenever:

$$2^m B < q^{(m-n)/m}.$$

We can set $m \approx \sqrt{n} \log q$. This tells us that the attack works whenever $q^{\sqrt{n}} \ll q/B$. So if q is exponential in n and $B = \text{poly}(n)$, this attack may work.

References

[1] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

[2] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[3] David Wagner. A generalized birthday problem. In *Annual International Cryptology Conference*, pages 288–304. Springer, 2002.