COS 533: Advanced Cryptography — Princeton University
Lecture 20 (November 29, 2017)
Lecturer: Mark Zhandry — Scribe: Fermi Ma

Notes for Lecture 20

1 Using LLL to Find Roots Mod N

Last time we looked at the LLL algorithm. Given an arbitrary basis for the lattice, it finds a somewhat short basis for the lattice, and in particular it finds a somewhat short vector in the lattice. Today we'll see how to use the LLL algorithm to find solutions to problems that, on the surface, seem to have nothing to do with lattices.

1.1 Motivation

Consider the following problem. Let f be a degree d polynomial, and let N be an integer of unknown factorization (think of this as a product of two large primes). Find solutions to $f(x) \equiv 0 \mod N$.

In general, this is as hard as factoring N. You can take $f(x) = x^2 - a$, and solutions are square roots of a mod N. It's known that the ability to take square roots gives you the ability to factor N. So this is believed to be a hard problem.

But in some cases, you can find small roots in polynomial time.

Example is the RSA cryptosystem. The RSA problem is: given $x^e \mod N$ along with e, N, find x (where e is chosen so that this x^e is one to one). Often we'll have $e = 3$ in practice.

So given $f(x) = x^e \mod N$, finding x is assumed to be hard. But assume we have a guess $\tilde{x}$ for x, where $|\tilde{x} - x|$ is small. Then consider the function $f(z) = (\tilde{x} + z)^e - y$ where $y = x^e$. Note that a valid solution is $z = x - \tilde{x}$, since then you get $f(z) = 0$. This means that if we are given a good guess $\tilde{x}$ for x, we can find x exactly as long as we can find a small solutions for z. This is precisely what the following theorem gives us.

1.2 Main Theorem

Theorem. Let N be an integer with unknown divisor $N' \geq N^\beta$ where $0 \leq \beta \leq 1$. Let $f(x)$ be a monic degree d polynomial. Let $c \geq 1$. Then we can find all solutions x_0 to $f(x_0) \equiv 0 \mod N'$ such that $|x_0| \leq cN^{\beta^2/d}$ in time $\mathsf{poly}(c, d, \log N)$.

As a special case, let $\beta = 1$. This implies $N' = N$. Then the theorem says we can find all solutions to $f(x_0) \equiv 0 \mod N$ such that $|x_0| \leq N^{\frac{1}{d}}$

In our RSA example, this theorem says we can find a root z as long as $|\tilde{x} - x| \leq N^{\frac{1}{e}}$.

Another application is that we can factor $N = pq$ given an approximation $\tilde{p}$ for p where $|\tilde{p} - p|$ is small. Also assume that $p \approx q \approx \sqrt{N}$. We can prove this with the main theorem, but not the special case.

We let $f(x) = \tilde{p} + x$. The goal is to find a short solution to $f(x) \equiv 0 \mod p$. If we set $\beta = 1/2$ and $N' = p$. We can find $x_0 = p - \tilde{p}$ as long as $|x_0| \leq N^{\frac{1}{4}}$.

This gives an algorithm for factoring N in time $N^{\frac{1}{4}}\mathsf{poly}(\log N)$ by just trying different guesses for $\tilde{p}$.

This theorem can sort of be extended to multivariate polynomials, although the multivariate version of this algorithm is just a heuristic.

A further application of this theorem is to rule out RSA with a small decryption exponent d. Normally in RSA we have $pk = e$ and $sk = d$ and $ed \mod \phi(N) \equiv 1$. Decryption computes $c^d \mod N$. Note that the running time of decryption grows with $\log d$ (via repeated squaring). This theorem tells us that we can factor N if d is too small. Alternatively if $N = p^r q$, we can factor N if r is too big.

1.3 Main Theorem Proof

Now we prove the main theorem.

Proof Ideas. What if the coefficients of f are all small? Then $f(x_0) \equiv 0 \mod N$ for a small root x_0 actually implies $f(x_0) = 0$ over $\mathbb{Z}$ (as long as $|f(x_0)| < N$). If it holds over the integers, then we can find the roots using standard numerical methods. But for all the examples we're looking at, f does not have small coefficients.

So the goal is to find g such that $g(x_0) \equiv 0 \mod N$ for all roots x_0 of f, and g has small coefficients (g may have more roots, but that doesn't matter since we can just test which ones are roots of f).

Let $n \geq d$ be some integer that's at least the degree of f. The set of polynomials with integer coefficients of degree at most n form a lattice. This lattice isn't that interesting, but now consider L_{x_0}, the lattice of polynomials with root $x_0 \mod N'$. This is a lattice since adding two such polynomials still gives another polynomial with this property. Unfortunately, we don't know how to compute this lattice since we don't know what x_0 is. Instead, we'll construct a lattice $L \subseteq L_{x_0}$ that is contained in L_{x_0}. Then we'll use LLL to find a short vector in L, which will be our g.

Proof. How do I construct this lattice L? L will be the span of some polynomials in L_{x_0}. One such polynomial is f, but we can also look at $xf(x), x^2(f)x, f^2(x), f^3(x), N, Nx, Nx^2, etc.$

We choose some integer m and some integer t. We'll come back to figuring out what this m is later. We'll consider the polynomials

$$g_{ij}(x) = N^i x^j f^{m-i}(x),$$

where $i \in [1, m]$ and $j \in [0, d-1]$, and the polynomials

$$h_i(x) = x^i f^m(x),$$

where $i \in [0, t-1]$.

We're not going to apply LLL directly to these polynomials. Instead we'll apply LLL to $g_{i,j}(xB), h_i(xB)$ for some bound B on $|x_0|$. So we'll think of these polynomials as functions of xB. This scales up the higher order coefficients by powers of B. Also we'll show that the dimension of the space these polynomials span is $n = md + t$ (this is immediate once we show these polynomials are linearly independent).

First we use (but won't prove) a lemma that states LLL produces a vector of length at most $2^{\frac{n-1}{4}} det(L)^{\frac{1}{n}}$. To compute this determinant we can take the determinant of any basis of the lattice.

Computing the determinant of this lattice isn't too hard, if we observe that the polynomials can be arranged into a lower triangular matrix (where the determinant is just the product of the all the diagonal entries). To see this, look at the degree of $g_{i,j}$. f has degree d, so it has degree $j + d(m-i)$. For each choice of i and j, this gives me something distinct (think of looking at the degree of g as a two digit number in base j, where the least significant bit tells me j and the most significant bit is enough information to uniquely figure out i). The degree of h_i is $dm + i$. For all the allowed choices of i and j, these degrees hit every integer from 1 to $n = md + t$ and don't overlap. So we have linearly independence, but more importantly we can just get the determinant by multiplying the leading terms of these polynomials.

This is a straightforward computation, since f is a monic polynomial. Note that that we're looking at the inputs scaled up by B. Collecting all the terms gives

$$det(L) = \prod_{i,j} N^i B^{d(m-i)+j} B^{dm+i} = N^{\frac{1}{2}dm(m+1)} B^{\frac{1}{2}m(n-1)}$$

The point of multiplying by B is that the bound we calculate actually bounds what you would get if you plugged in an x that is at most B.

So LLL outputs g such that $|g(xB)| \leq 2^{\frac{n-1}{4}} N^{\frac{md(m+1)}{2n}} B^{\frac{(n-1)}{2}}$.

Note that elements in this lattice are not just 0 mod N', but actually 0 mod $(N')^m$.

Now we state one more lemma (this time with proof) that allows us to claim that the LLL algorithm outputs a polynomial g that is small enough that we can solve it over the integers.

Lemma. Let $g(x)$ be a polynomial of degree n. Let $m \geq 0, B \geq 0$. Suppose that $x_0 \leq B$ and that the following two conditions hold:

$$\begin{aligned} &1) \quad g(x_0) \equiv 0 \mod (N')^m \\ &2) \quad |g(xB)| < \frac{(N')^m}{\sqrt{n}} \end{aligned}$$

Then we have that $g(x_0) = 0$ over $\mathbb{Z}$.

The proof of this lemma is straightforward. We write $|g(x_0)| = |\sum_i c_i x_0^i| \leq \sum_i |c_i x_0^i|$ where we've applied the triangle inequality to bring the absolute value into the sum.

Then using our bound on x_0, we conclude $|g(x_0)| \leq \sum_i |c_i| B^i$. Relating an ℓ_1 norm to an ℓ_2 norm incurs a $\sqrt{n}$ factor, we have $\sum_i |c_i| B^i \leq \sqrt{n}|g(xB)| < (N')^m$ (last inequality is from condition 2). So we conclude that $g(x_0) = 0$ over the integers (end of lemma proof).

If we want to apply this lemma, we match up what we know with what the lemma requires for its second condition. So we need that

$$2^{\frac{n-1}{4}} N^{\frac{md(m+1)}{2n}} B^{\frac{(n-1)}{2}} \leq \frac{N^{\beta m}}{\sqrt{n}}$$

What you get is $B \leq N^{\frac{\beta^2}{d} - \epsilon}$ for any $\epsilon \in (0, \beta/6]$ in time $poly(\frac{1}{\epsilon}, d, \log N)$. So far this shows that we can get arbitrarily close to our theorem statement, but we don't get the exact theorem statement. What we can do is set $\epsilon = \frac{1}{\log N}$, which makes the running time $N^{\frac{-1}{\log N}} N^{\frac{\beta^2}{d}} = N^{\beta^2/d}$. Basically we repeat this algorithm c times (for c different guesses) and we get the running time $cN^{\beta^2/d}$.

So in summary, the algorithm first constructs a lattice that's a sublattice of the lattice L_{x_0} we care about. We apply LLL to get a short vector in the lattice, and then use a lemma to conclude this short vector actually corresponds to a polynomial we can solve over the integers.