IAS/Park City Mathematics Series
Volume 00, Pages 000–000
S 1079-5634(XX)0000-0

Small solutions to polynomial equations using lattices and applications in cryptography

Nadia Heninger

Abstract. These lectures give an overview of Coppersmith and Howgrave-Graham's methods for finding small solutions to polynomial equations modulo integers by finding a short vector in a lattice. We present several variations on the form of the problem, with applications to cryptography.

These notes are a very rough draft, and are very informal in many places.

1. Introduction

Lattices have numerous applications to problems of interest in cryptography, both destructive (cryptanalysis using lattice algorithms) and constructive (developing new cryptographic algorithms that we believe are secure if our understanding of lattice algorithms is correct).

- **Cryptanalysis:** Can use approximation algorithms for SVP in lattices to cryptanalyze a wide variety of classical cryptography:
  - Attacks on low public exponent RSA
  - Factoring with partial knowledge
  - (EC)DSA with partial information about nonces
  - Knapsack-based cryptosystems

  The first three of these applications will be the focus of these lecures.

- **Cryptographic constructions:**
  - Lattice problems appear to be hard to solve for quantum computers, so lattice-based cryptosystems among most promising candidates for post-quantum cryptography. (Several just chosen by NIST!)
  - Algebraic structure of lattices leads to many interesting cryptographic constructions that may someday be practical, like fully homomorphic encryption, identity-based encryption, etc.

2010 *Mathematics Subject Classification.* Primary 14Dxx; Secondary 14Dxx.
Key words and phrases. Park City Mathematics Institute.

2. Lattice background

Definition 2.0.1. A *lattice* is a subset of $\mathbb{R}^n$ generated by integer linear combinations of some linearly independent basis vectors $\{b_1, \dots, b_m\}$, $b_i \in \mathbb{R}^n$.

We can represent elements in our lattice explicitly as vectors. For example, the origin $v_o = (0, 0, \dots, 0)$ is always in any nonempty lattice. In general, a lattice vector $v = \sum_{i=0}^{m} a_i b_i = (z_1, \dots, z_n)$ with $a_i \in \mathbb{Z}$ and b_i basis vectors, so $z_i \in \mathbb{R}$. Since we focus on computational applications, any real number will have to be approximated by a rational. Given a lattice of rationals, we can clear denominators and work over the integers $\mathbb{Z}$.

A lattice over $\mathbb{R}^n$ has useful computational properties. Algebraically, it is a group under the operation addition. Geometrically, it lives in $\mathbb{R}^n$ so it inherits the dot product and distance metrics from $\mathbb{R}^n$. We mostly care about the ℓ_1 and ℓ_2 metrics. We'll write $|\cdot|$ for the ℓ_2 metric when not otherwise specified.

It is possible to show that a lattice is *discrete*, that is, that $\exists \delta > 0$ s.t. $|v_i - v_j| > \delta$ $\forall\, v_i \neq v_j \in L$.

We can also use these properties to give a less constructive definition of a lattice that is equivalent to Definition 2.0.1.

Definition 2.0.2. A *lattice* is a discrete additive subgroup of $\mathbb{R}^n$.

Theorem 2.0.3. *In n dimensions a lattice has a basis of size at most n.*

We will generally represent a basis as a matrix B whose rows are the basis vectors $b_1, \dots, b_m$. A lattice may not be full rank, but for our applications it typically is, so we can take $m = n$. We will write $L(B)$ to represent the lattice generated by basis B.

A basis for a lattice is not unique.

Theorem 2.0.4. *Deciding if $L(B) = L(B')$ for $B \neq B'$ is efficient.*

Proof. The Hermite Normal Form of a matrix over $\mathbb{Z}$ is unique and efficient to compute. Given B and B' we can check if $\mathrm{HNF}(B) = \mathrm{HNF}(B')$. □

Definition 2.0.5. The **determinant** of a lattice with basis B is $|\det B|$.

Theorem 2.0.6. *The determinant is invariant for a given lattice.*

Geometrically, we can interpret the determinant of the lattice as giving the volume of the fundamental parallelepiped inscribed by the basis vectors: $\sum_{i=0}^{n} a_i b_i$ with $0 \leqslant a_i < 1$.

Definition 2.0.7 (Dual lattice)**.** For a full-rank lattice L, we can define the *dual lattice* $L^* = \{u \in \mathbb{R}^n \mid \forall\, v \in L, \langle v, u \rangle \in \mathbb{Z}\}$.

For a full-rank lattice, computing a basis for the dual lattice is easy: for a basis B of $L(B)$, $B^* = (B^{-1})^T$ is a basis for the dual $L^*(B)$.

Let $\lambda_1 > 0$ be the length of the shortest vector in the lattice. This is well defined because the lattice is discrete.

Definition 2.0.8. The i*th successive minimum* λ_i is the smallest radius of a ball containing i linearly independent lattice vectors.

You can think of λ_i as the length of the shortest vector linearly independent to the vectors achieving the $i-1$ successive minima.

Theorem 2.0.9 (Minkowski)**.** $\lambda_1(L) < \sqrt{n} \det L^{1/n}$

The *Gaussian Heuristic* gives an expectation for the length of the shortest vector in a lattice. It says that

$$gh(L) \approx \sqrt{\frac{n}{2\pi e}} (\det L)^n$$

Definition 2.0.10 (Shortest Vector Problem (SVP))**.** Given an arbitrary basis B for a lattice, find a shortest vector in $L(B)$ the lattice generated by B.

Theorem 2.0.11.
SVP is NP-hard.

Definition 2.0.12 (Closest Vector Problem (CVP))**.** Given an arbitrary basis B, and a point x find a vector in $L(B)$ closest (with shortest distance to) to x.

- CVP is NP-hard.

Definition 2.0.13 (Bounded-Distance Decoding (BDD))**.** Find all lattice points within a specified radius of an arbitrary point.

It may be NP-hard to solve the shortest vector problem exactly, but we can compute an approximation to the shortest vector (and in fact to all of the successive minima) in polynomial time, albeit with an exponential bound on the approximation factor.

Theorem 2.0.14 (Lenstra Lenstra Lovasz (LLL)[12])**.** *Given a basis* B *for a lattice, we can in polynomial time find a* reduced *basis* $\{b_i\}$ *s.t.*

$$|b_i| \leqslant 2^{(n-1)/2} \lambda_i$$

Note that the approximation factor is exponential in the lattice dimension. This does not seem like a very good approximation, but it turns out that it is good enough for many applications. In addition, many applications might only require lattices of fixed (small) dimension, in which case this approximation factor is a constant.

Theorem 2.0.15 (LLL, Informal)**.** *We can find a vector of length*

$$|v| < 2^{\dim L}(\det L)^{1/\dim L}$$

Empirically, the LLL algorithm seems to perform better than the worst-case bounds on random lattices. For many cryptanalysis-type problems, the behavior of LLL matches this prediction.

Conjecture 2.0.16 (Nguyen Stehlé [15])**.** *For a random lattice, LLL finds a vector* $|v| < 1.02^{\dim L}(\det L)^{1/\dim L}$

It is an open problem to explain or fully characterize this behavior.

If we need a better approximation factor, we can exchange exponential running time to achieve it by using blockwise SVP solvers for larger block sizes k. [1,8,9,13]

Theorem 2.0.17 (Informal)**.** *Given a lattice basis, can in time* $2^{O(k)}$ *find a reduced basis s.t.* $|b_i| \leqslant k^{O(n/k)}\lambda_i$.

There is a long historical connection between lattice reduction and problems concerning polynomials. Consider the following problem from the original Lenstra Lenstra Lovasz lattice reduction paper [12]. Let α be an algebraic number. If we have a guess for its degree, we can try to find its minimal polynomial by finding a short vector in the lattice generated by the rows of the following basis:

$$B = \begin{bmatrix} 1 & 0 & \dots & 0 & \alpha^d \\ 0 & 1 & \dots & 0 & \alpha^{d-1} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & \alpha^0 \end{bmatrix}$$

If $f(x) = f_d x^d + \cdots + f_0$ is the minimal polynomial of α so $f(\alpha) = 0$, we would hope that there is a vector $v_f = (f_d, \dots, f_0, 0)$ in this lattice. There are a few complications to making this work in practice. First, we need to represent α to some precision, so the last column will not be exactly 0. Second, we want this vector to be particularly short in order to find it among the short vectors of the lattice, which means that we should multiply a scaling factor to the last column of the basis. It requires some work to find the right balance of these two choices.

3. Univariate polynomials modulo integers

This section will introduce Coppersmith's method for using lattice basis reduction to find small roots of polynomials modulo integers. Coppersmith's original presentation of this method is in [6]. Howgrave-Graham gave a dual formulation of this algorithm that is more intuitive to reason about, which is the presentation we give here. [11] Coppersmith later wrote up a survey of improvements and optimizations to this method. [7]

3.1. Motivation Let's begin with some context for why this algorithm is interesting in the first place, since the question of how difficult it is to find roots of polynomials varies by the setting.

- **Polynomial time:** Solving $f(x) \equiv 0$ over $\mathbb{Q}$. In fact, the LLL paper gives such an algorithm.
- **Polynomial time:** Solving $f(x) \equiv 0 \bmod p$, p prime. The Cantor-Zassenhaus algorithm and Berlekamp's algorithm are polynomial-time algorithms for factoring polynomials over finite fields.
- **Polynomial time:** Solving $f(x) \equiv 0 \bmod N$, factorization of N known, N has few factors. For a composite integer, factor into primes, solve mod each prime, and use Chinese remainder theorem and Hensel lifting to lift solutions mod N. This is efficient if N is easy to factor or its factorization is already known, and N does not have too many prime factors.
- **Not polynomial time:** Finding all solutions to $f(x) \equiv 0 \bmod N$, factorization of N known, N has many factors.

  f can have exponentially many roots in the number of factors of N. Consider $f(x) = x^d$ and $N = p^d$ for a prime p. Any of the p^{d-1} multiples of p modulo N can be a root.
- **Hopefully not polynomial time:** Computing roots of $f(x)$ modulo N, factorization of N unknown.

  We hope that this problem is hard because a general method to solve polynomials mod N would break RSA. Recall that an RSA public key is an integer $N = pq$ whose factorization is not made public, along with a public exponent e; say $e = 3$ is not known to be insecure in general. With textbook RSA, the ciphertext c encrypting a message m is computed as $c = m^e \bmod N$.

  Now imagine that the adversary has observed the ciphertext c and the public key N and e. Then the polynomial

  $$x^e - c \equiv 0 \bmod N$$

  has a root $x = m$ for m our original message. If an adversary can efficiently find roots of polynomials of this form, then RSA is insecure.

  Right now, the only methods known to solve this problem in general require learning the factorization of N and computing m as above. However, intriguingly, it is not known that finding roots of the RSA polynomial above is equivalent to factoring the modulus.

3.2. Coppersmith's theorem Coppersmith's method gives us an efficient method to find some roots of low degree polynomials modulo integers without having to factor the modulus. However, the tradeoff is that we need to accept a bound on the size of root that can be efficiently found.

Theorem 3.2.1 (Coppersmith)**.** *Given a monic polynomial $f \in \mathbb{Z}[x]$ of degree d and modulus $N \in \mathbb{Z}$, there is an algorithm to find all $r_i \in \mathbb{Z}$ satisfying*

$$f(r_i) \equiv 0 \bmod N$$

when $|r_i| < N^{1/d}$. The algorithm runs in time polynomial in $\log N$ and d.

3.3. Coppersmith's algorithm outline We will start with a high-level overview of the method, and then dive into each of the steps in return.

Input: $f = x^d + f_{d-1}x^{d-1} + \cdots + f_0 \in \mathbb{Z}[x]$, modulus $N \in \mathbb{Z}$

Desired output: All $r \in \mathbb{Z}, |r| < N^{1/d}$ with $f(r) \equiv 0 \bmod N$.

(1) Consider the lattice generated by coefficient vectors of polynomials h_i of bounded degree that satisfy $h_i(r) \equiv 0 \bmod N$. Any polynomial h in this lattice is an integer combination of the h_i and thus satisfies $h(r) \in \mathbb{Z}$ for r satisfying $f(r) \equiv 0 \bmod N$.

(2) Pick a concrete basis for a sublattice and run the LLL algorithm to find a short element in this sublattice.

(3) Construct an auxiliary polynomial Q from the shortest vector output and prove that $Q(r) = 0$ over $\mathbb{Z}$ not just modulo N.

(4) Find the roots r_i of Q over $\mathbb{Q}$ and output those satisfying $f(r_i) \equiv 0 \bmod N$.

3.4. Choosing a suitable set of basis polynomials Let $f(x) = x^3 + f_2x^2 + f_1x + f_0$ and N be the inputs to our problem. Any polynomial multiple $h_i(x)$ of $f(x)$ and/or N satisfies $h_i(r) \equiv 0 \bmod N$ by construction.

Example 3.4.1. If we only care about polynomials Q of degree 3, then we might try to construct our target polynomial Q from

$$Q(x) = c_3 f(x) + c_2 N x^2 + c_1 N x + c_0 N$$

with $c_3, c_2, c_1, c_0 \in \mathbb{Z}$.

$$\begin{array}{ccccccccc}
 & c_3 & (x^3 & + & f_2x^2 & + & f_1x & + & f_0) \\
+ & c_2 & & & Nx^2 & & & & \\
+ & c_1 & & & & & Nx & & \\
+ & c_0 & & & & & & & N \\
\hline
 & & Q_3x^3 & + & Q_2x^2 & + & Q_1x & + & Q_0
\end{array}$$

3.4.1. Manipulating polynomials as coefficient vectors We can represent elements of $\mathbb{Z}[x]$ as coefficient vectors:

$$g_d x^d + g_{d-1}x^{d-1} + \cdots + g_0 \qquad \leftrightarrow \qquad (g_d, g_{d-1}, \ldots, g_0)$$

If we construct the matrix

$$\begin{bmatrix} 1 & f_2 & f_1 & f_0 \\ & N & & \\ & & N & \\ & & & N \end{bmatrix}$$

Then the coefficient vector representing our polynomial

$$Q(x) = c_3 f(x) + c_2 N x^2 + c_1 N x + c_0 N$$

is an integer combination of the rows of this matrix.

Observe that integer combinations of the coefficient vectors of polynomials over $\mathbb{Z}[x]$ of fixed maximum degree form a lattice. (It is a group under addition, and satisfies the discreteness property because the vectors are integers.

3.5. Ensuring that our auxiliary polynomial Q vanishes over $\mathbb{Z}$. The inputs to our algorithm were $f(x) \in \mathbb{Z}[x]$, $N \in \mathbb{Z}$.

As an intermediate output of the algorithm, we wanted to find $Q(x)$ such that $Q(r) = 0$ over $\mathbb{Z}$.

(1) $Q(x) \in \langle f(x), N\rangle$ so $Q(r) \equiv 0 \bmod N$ by construction.

(2) If $|r| < R$, then we can bound

$$\begin{aligned} |Q(r)| &= |Q_d r^d + Q_{d-1} r^{d-1} + \cdots + Q_1 r + Q_0| \\ &\leqslant |Q_d| R^d + \cdots + |Q_2| R^2 + |Q_1| R + |Q_0| \end{aligned}$$

(3) If $|Q(r)| < N$ and $Q(r) \equiv 0 \bmod N$ then $Q(r) = 0$.

We want a Q in our lattice with short coefficient vector!

However, the lattice in the example above doesn't quite capture the same notion of shortness that we need. Observe that if we scale each column of the lattice corresponding to the coefficient of the monomial x^i in our basis polynomial by R^i, then the ℓ_1 norm of the vector corresponding to $h(x)$ is an upper bound on the evaluation of $|h(r)|$ for $|r| < R$. The bounds given by the LLL algorithm are in terms of the ℓ_2 norm, but we can multiply by $\sqrt{\dim L}$ to get a bound.

We can establish the following series of inequalities.

$$|Q(r)| \leqslant |v_Q|_1$$

We showed this above, by construction.

$$|v_Q|_1 \leqslant \sqrt{\dim L}|v_Q|_2$$

$$\sqrt{\dim L}|v_Q|_2 \leqslant \sqrt{\dim L}2^{\dim L} \det L^{1/\dim L}$$

This is the loose bound for the LLL algorithm. If you wish, you can use the more precise or empirical bounds.

Thus in order to achieve the bound $|Q(r)| < N$, it suffices to show for a particular lattice construction

$$\sqrt{\dim L}2^{\dim L} \det L^{1/\dim L} < N$$

This is easiest demonstrated with a small example.

Example 3.5.1. Input: $f(x) = (x+a)^3 - c$, N
Output: $r < R$ such that $f(r) \equiv 0 \bmod N$.

(1) Construct lattice basis

$$\begin{bmatrix} R^3 & 3aR^2 & 3a^2R & a^3 - c \\ & NR^2 & & \\ & & NR & \\ & & & N \end{bmatrix}$$

We have

$$\dim L = 4$$
$$\det L = R^6 N^3$$

So to establish the theorem, we should show

$$\sqrt{\dim L}2^{\dim L} \det L^{1/\dim L} < N.$$

Since $\dim L$ is a small constant, let's drop the approximation factors below. Then we expect the algorithm to work when

$$\det L^{1/\dim L} < N$$
$$(R^6 N^3)^{1/4} < N$$
$$R < N^{1/6}$$

That is, we can find roots of absolute value up to $N^{1/6}$ just by reducing a 4-dimensional lattice.

3.6. Achieving the Coppersmith bound $r < N^{1/d}$ What is shown above isn't quite enough to get all the way to Coppersmith's bound $|r| < N^{1/d}$. In order to achieve this bound, the polynomials in the input basis need two additional properties:

(1) Higher multiplicities: Generate lattice from subset of $\langle f(x), N\rangle^k$. That means that $Q(r) \equiv 0 \bmod N^k$.
(2) Higher degree polynomials: Allow higher degree polynomials (higher than the polynomials of kd generated by the above).

The multiplicity shouldn't be the same as total degree, and asymptotically you don't want it to be.

In order to achieve the bounds, bound $|Q(r)|$ by computing the dimension and determinant of the lattice in terms of multiplicity and degree, and then optimize the multiplicity and total degree. This gets one almost to the bound $N^{1/d}$. Coppersmith himself resorts to brute forcing a few final bits to get all the way to the bound.

This optimization is a bit annoying, so we will leave it as an exercise for the dedicated reader. Or you can check the computations worked out in general in [5]. In order to obtain a polynomial-time algorithm, we need polynomial bounds on the total degree (which determines the dimension of the lattice to be reduced), the multiplicity (which determines the size of the entries in the lattice), and to show that a polynomial-time algorithm like LLL can find a sufficiently short vector.

It is interesting that this result does not require a better than exponential approximation factor for the lattice reduction algorithm used. Coppersmith notes that the approximation factor of LLL becomes a constant factor of 2 on the bound of the size of the root that can be obtained, which can be eliminated by brute forcing one bit of the root.

3.6.1. Dual lattices and the solution vectors Sometimes it is easier to work with the dual of the lattice as defined above. Observe that for our lattice L as constructed above whose vectors are all of the form $v_h = (h_m R^m, \dots, h_d R^d, \dots, h_0$ corresponding to the coefficient vectors of some polynomial $h(x) = h_m x^m + \cdots + h_0$ satisfying $h(r) \equiv 0 \bmod N^k$, for any of our desired integer roots r, the vector $u_r = (r^m/(R^m N^k), \dots, r_d/(R^d N^k), \dots, r_0/N^k$ is an element of the dual L^*, because $\langle v_h, u_r \rangle = h(r)/N^k \in \mathbb{Z}$ by construction. We can also observe that for $|r| < R$, a vector u_r will be small.

3.6.2. Is Coppersmith's bound $|r| < N^{1/d}$ optimal? Yes, in general. Consider $f(x) = x^d$, $N = p^d$. Attempting to improve bound to $|r| < N^{1/d+\epsilon}$ yields $2N^\epsilon$ multiples of p that are roots. Exponentially many solutions cannot be enumerated in polynomial time.

This counterexample does not rule out most applications of cryptographic interest, where e.g. $N = pq$ or solution is known to be unique by construction.

Using more advanced methods, it is possible to rule out the existence of an "obvious" improvement over the above that might improve on Coppersmith's bound. That is, it is not possible to solve for $r > N^{1/d}$ with any method that constructs an auxiliary polynomial $Q(x)$ using powers, polynomial multipliers, or any other method that preserves all of the p-adic and rational roots of the input polynomial.

Theorem 3.6.1 (Chinburg, Hemenway, Heninger, Scherr [4]). *Let* $f \in \mathbb{Z}[x]$ *be monic, degree* d*, modulus* $N \in \mathbb{Z}$*,* $\epsilon > 0$*. There is no auxiliary polynomial of the form*

$$h(x) = \sum_{i,j} a_{i,j} x^i (f/N)^j$$

so that $|h(r)| < 1$ *for all* r *satisfying* $|r| \leqslant N^{1/d+\epsilon}$.

3.7. Some open problems

(1) Is there an alternative to Coppersmith's method that circumvents these limitations?

(2) Lattices throw away the multiplicative structure of our ideals. Is there a tool that doesn't?

(3) Is computing eth roots mod N equivalent to factoring N?

3.8. Exercises Your exercise for this lecture is to implement a simplified version of Coppersmith's method to break an RSA-encrypted ciphertext with a partially predictable message.

4. Finding roots of polynomial equations modulo divisors of integers

In this section, we will study the problem of finding solutions to polynomials (simplified to a linear equation to start with) modulo divisors of integers. Like the previous section, we will address a few variants of this problem in order to understand what makes it potentially interesting.

- **Polynomial time:** Solving $x - a \equiv 0$ over $\mathbb{Q}$. This one is obvious as stated.
- **Polynomial time:** Solving $x - a \equiv 0 \bmod B$, B prime.
- **Polynomial time:** Solving $x - a \equiv 0 \bmod B$, such that $B \mid N$, factorization of N known, N has few factors. As before, we can solve modulo each prime, and then use the Chinese remainder theorem and Hensel lifting to enumerate solutions modulo each divisor of N.
- **Not polynomial time:** Finding all solutions to $x - a \equiv 0 \bmod B$ such that $B \mid N$, factorization of N known, N has many factors. There are exponentially many B, so enumerating all of them is exponential time.
- **Hopefully not polynomial time:** Solving $x - a \bmod B$, $B \mid N$, factorization of N unknown. Finding such a solution would reveal a factor of N.

We will sketch the method to prove the following theorem in this section.

Theorem 4.0.1 (Howgrave-Graham). *Given* $f(x) = a + x$*, integer* N*,* $0 < \beta \leqslant 1$*, we can find* r *satisfying*

$$f(r) \equiv 0 \bmod B$$

for $B \mid N$, $|B| > N^{\beta}$*, when* $|r| < N^{\beta^2}$.

This immediately gives an alternative proof of a result that Coppersmith proved using a slightly different lattice method, which is stated in a specialized form to the RSA application below.

Theorem 4.0.2 (Coppersmith 1996)**.** *Let* $N = pq$ *with* $p, q \approx \sqrt{N}$*. Given half the bits (most or least significant) of* p*, we can factor* N *in polynomial time.*

To show this using Theorem 4.0.1, let a be a value such that $p = a + r$ with $|r| < p^{1/2}$.Then r is a small solution to $f(x) = a + x$ modulo p, a divisor of N. Since $p \approx \sqrt{N}$, we set $\beta = 1/2$, and then Theorem 4.0.1 tells us we should be able to find $r < N^{1/4} \approx p^{1/2}$.

If we wish, we can also combine Theorem 3.2.1 with Theorem 4.0.1. This version was proven by May.

Theorem 4.0.3 (May)**.** *Given degree* d *polynomial* f*, integer* N*, we can find roots* r *modulo divisors* B *of* N *satisfying*

$$f(r) \equiv 0 \bmod B$$

for $|B| > N^{\beta}$*, when* $|r| < N^{\beta^2/d}$ *in time polynomial in* $\log N$ *and* d*.*

4.1. Howgrave-Graham's algorithm outline The algorithm to solve this variant is very similar to the previous section; we just need to change the bound on the size of the vector we are looking for in the lattice.

Input: $f = x + a \in \mathbb{Z}[x],\ N \in \mathbb{Z},\ 0 < \beta < 1$

Desired output: All $r \in \mathbb{Z}, |r| < N^{\beta^2}$ s.t. $f(r) \equiv 0 \bmod B,\ B \mid N$.

(1) Generate lattice of polynomial multiples of f, N:
$$N^k, xN^k, \dots, fN^{k-1}, \dots, f^k, xf^k, \dots$$
Any polynomial h in this lattice satisfies $h(r) \equiv 0 \bmod B^k$ for $r \in \mathbb{Z}$ satisfying $f(r) \equiv 0 \bmod B$.

(2) Pick a concrete basis for a sublattice and run the LLL algorithm to find a short element in this sublattice.

(3) Construct a polynomial Q from the shortest vector output.

(4) Find the roots r_i of Q. $|Q(r)| < B^k$ and $h(r) \equiv 0 \bmod B^k \implies h(r) = 0$, so any desired r must be a root of Q over $\mathbb{Q}$.

4.2. Partial RSA key recovery example Let a encode some most significant bits of p, a divisor of N, so that $p - a$ is small, that is, there is a small r such that $a + r = p$. We will give a small example to show how effective this method is even with very small lattices.

Input: $f(x) = a + x, N$

Output: $r < R$ s.t. $f(r) \equiv 0 \bmod p, \quad p | N, \quad p \geqslant N^{1/2}$

(1) We chose the polynomial basis $x(x+a), (x+a), N$.
(2) This corresponds to a lattice basis

$$\begin{bmatrix} R^2 & Ra & 0 \\ 0 & R & a \\ & & N \end{bmatrix}$$

$$\dim L = 3$$
$$\det L = R^3 N$$

(3) LLL will find us a vector of size about $|v| \approx \det L^{1/\dim L}$.
(4) The algorithm will find the root when we have

$$|Q(r)| \leqslant |v| \approx \det L^{1/\dim L} < p$$
$$(R^3 N)^{1/3} < N^{1/2}$$
$$R < N^{1/6}$$

4.2.1. Achieving the Howgrave-Graham bound $r < N^{\beta^2}$. The above example doesn't quite get to the full bound that Howgrave-Graham establishes. As in the the univariate Coppersmith's theorem in the previous section, to achieve the full bound, we need to generate a larger lattice as follows:

(1) Generate the lattice from subset of $\langle f(x), N\rangle^k$ so that any polynomial in the lattice vanishes modulo B^k.
(2) Allow higher degree polynomials.

The optimization of these parameters remains annoying, and we leave it to the enthusiastic reader, or take a look at [5].

Like the univariate Coppersmith's theorem modulo N, this result does not require better than an exponential approximation factor on the size of the vector found in the lattice.

4.2.2. Exercise Your exercise for this lecture is to implement factorization from partial knowledge using this method.

5. Hidden number problem

In the *hidden number problem* defined by Boneh and Venkatesan, there is a secret integer d and a public integer modulus n (it could be prime but does not need to be). An adversary chooses integers $t_1, \ldots, t_m$, computes a relation $a_i + r_i \equiv t_i d \bmod n$ for some a_i and r_i, and reveals the values $\{(a_i, t_i)\}_{i=1}^{n}$. The problem is to compute the secret integer d from this information.

To put this in a familiar framework, we can write down the problem as a system of linear equations in unknowns $x_1, \ldots, x_m, y$:

$$a_1 - t_1 y + x_1 \equiv 0 \bmod n$$

$$a_2 - t_2 y + x_2 \equiv 0 \bmod n$$
$$\vdots$$
$$a_m - t_m y + x_m \equiv 0 \bmod n$$

There are $m+1$ unknowns and m equations.

- If there are no size bounds on x_i or y, then there are n possible solutions. This is exponential in the size of the input (the bit length of n).
- If y can be chosen uniformly modulo n, then heuristically we expect to have a unique solution once the $|r_i| < n^{(m-1)/m}$.

Note that we can carry out a linear transformation to eliminate the variable y from our input linear relations and obtain a problem of the form

$$a'_1 - t'_1 x_m + x_1 \equiv 0 \bmod n$$
$$a'_2 - t'_2 x_m + x_2 \equiv 0 \bmod n$$
$$\vdots$$
$$a'_{m-1} - t'_{m-1} x_m + x_{m-1} \equiv 0 \bmod n$$

We will drop the $'$ in the following and assume this linear transformation has already been carried out.

5.1. Solving the hidden number problem with lattices

The input to the problem is a collection of linear relations

$$a_1 - t_1 x_m - x_1 \equiv 0 \bmod n$$
$$a_2 - t_2 x_m - x_2 \equiv 0 \bmod n$$
$$\vdots$$
$$a_{m-1} - t_{m-1} x_m - x_{m-1} \equiv 0 \bmod n$$

in unknowns $x_1, \dots, x_m$, where there is a desired solution $x_i = r_i$ with $|r_i| < R$.

Construct the lattice basis

$$B_s = \begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ -t_1 & -t_2 & \dots & -t_{m-1} & 1 & \\ a_1 & a_2 & \dots & a_{m-1} & & R \end{bmatrix} \tag{5.1.1}$$

The vector $v_r = (r_1, r_2, \dots, r_m, R)$ is in this lattice by construction.

We have:

$$\dim L = m + 1$$
$$\det L = Rn^{m-1}$$

The Gaussian Heuristic tells us that we expect the shortest vector in the lattice to satisfy

$$|v| \leqslant \sqrt{\frac{\dim L}{2\pi e}}(\det L)^{1/\dim L}$$

We are searching for a vector with length $|v_r| \leqslant \sqrt{m+1}R$.

Thus we expect that v_r will shorter than the expected shortest vector in the lattice and appear in a (fully) reduced basis when

$$\log R \leqslant \lfloor \log n(m-1)/m - (m-1)(\log(2\pi e))/(2m) \rfloor$$

In order to do well with many samples, we actually do need to solve the shortest vector problem, which can be done in practice using an algorithm like BKZ. This means that for large parameters, solving the hidden number problem at the limit is exponential time in the number of samples.

This tells us that as the number of samples m increases, the size of the bound R increases to n, but will not get all the way there.

Unlike Coppersmith's theorem, we are not guaranteed that any sufficiently short vector is a solution: there may be short vectors that do not solve the problem. To obtain a theorem, one needs to assume the randomness of the $\{t_i\}$ and give a statement that works with high probability. [3]

5.1.1. Application: Recovering ECDSA private keys from signatures with short nonces A useful application of the hidden number problem in cryptanalysis is to recover ECDSA secret keys from information about the signature nonces. This application is due to Nguyen and Shparlinski [14].

In this section we'll use some of the usual variable names in this application, which collide with some of the variables we've used in previous sections.

In the ECDSA signature scheme, global parameters include specification of an elliptic curve E with generator point G of order n. A secret key is an integer d modulo n, and the corresponding public key is the curve point dG. To generate a signature on a message hash h that we treat as an integer modulo n, the signer chooses a secret integer k. The signature has two components: $r = x(kG)$, the x-coordinate of the curve point, and $s = k^{-1}(h + dr) \bmod n$. Note that the relation defining the s-parts of the signaures is an integer linear relation modulo n.

In our key recovery scenario, the signer generates two signatures with short nonces k. Let $s_1 = k_1^{-1}(h_1 + dr_1) \bmod n$ and $s_2 = k_2^{-1}(h_2 + dr_2) \bmod n$ be the s-parts of the two signatures, and assume we know that $|k| < K$ for some bound K. We can eliminate d and rearrange terms to obtain

$$k_1 - s_1^{-1}s_2r_1r_2^{-1}k_2 + s_1^{-1}r_1r_2^{-1}h_2 - s_1^{-1}h_1 \equiv 0 \bmod n$$

Let $a_1 = -s_1^{-1}r_1r_2^{-1}h_2 - s_1^{-1}h_1$ and $t_1 = -s_1^{-1}s_2r_1r_2^{-1}$. Then we have a linear relation $a_1 - t_1k_2 - k_1 \equiv 0 \bmod n$ that we wish to solve for unknown small k_1 and k_2. We construct the lattice basis

$$B = \begin{bmatrix} n & & \\ -t_1 & 1 & \\ a_1 & & K \end{bmatrix}$$

It has determinant Kn and dimension 3. Ignoring approximation factors and constants, we expect there to be a target solution vector $v_k = (k_1, k_2, K)$ with $|v_k| \approx K$, and for there to be non-solution short vectors $v \approx (nK)^{1/3}$. If $|v_k| < (nK)^{1/3}$ then we hope to succeed with good probability. This is satisfied when $K < n^{1/2}$.

5.2. Writing down HNP as a coefficient embedding lattice Given the input problem as stated in Equation 5.1.1, it is natural to wonder whether we could use the polynomial coefficient-oriented approach to solve the problem, or improve over the bound above. Using the coefficient embedding, we could try to solve the problem by constructing the lattice basis

$$B_c = \begin{bmatrix} R & & \dots & t_1R & -a_1 \\ & R & \dots & t_2R & -a2 \\ & & \ddots & \vdots & \vdots \\ & & R & t_{m-1}R & -a_{m-1} \\ & & & nR & \\ & & & & n \end{bmatrix}$$

The rows of the lattice correspond to the scaled coefficient embedding of the linear equations in Equation 5.1.1, plus the coefficient embeddings of $nx_{m-1} \equiv 0 \bmod n$ and $n \equiv 0 \bmod n$.

Observe that $L(B_c) = nRL^*(B_s)$.

Then we would anticipate something like the following algorithm to be a faithful adaptation of the Howgrave-Graham/Coppersmith method for solving this problem:

(1) Run BKZ on this lattice to produce a fully reduced lattice.
(2) Each vector in the reduced lattice basis corresponds to a linear equation in the x_i. If the vector is sufficiently short, this relation is true over $\mathbb{Q}$ and not just modulo n.
(3) Each relation corresponding to a vector in the reduced lattice basis is linearly independent, because basis vectors are linearly independent.
(4) If we have m sufficiently short vectors, we solve the system of linear equations to obtain solutions for the x_i.

The main challenge in this approach is that we now require m short vectors in the reduced basis, instead of just one. There are bounds on the successive minima of a lattice, but in the generic case these are not able to give interesting bounds for m out of $m+1$ vectors in the reduced basis. One would need to analyze the structure of this lattice in particular. This can be done using Banaszczyk's transference theorems [2], but it is probably easier to just work with the solution embedding.

However, these dual formulations give a useful perspective on the lattice constructions: sometimes one or the other is easier to reason about or solve.

Given this perspective, it is also a natural question whether using higher degree polynomials might provide a benefit as it did in the Coppersmith/Howgrave case. Interestingly, it doesn't seem to.

5.2.1. Exercises Work out a variant of Coppersmith's method to solve for for multiple chunks of unknown bits.

5.2.2. Open problem Characterize when using higher degree polynomials provides a benefit for this type of problem.

6. Translation to $\mathbb{F}[z]$

In this section, we would like to translate these techniques from polynomials with integer coefficients to polynomials whose coefficients are polynomials in $\mathbb{F}[z]$.

6.1. Translating from $\mathbb{Z}$ to $\mathbb{F}[z]$. In this section, we'll review the various notions that we need to translate.

- We would like to replace the results over the ring of integers $\mathbb{Z}$ to the ring of polynomials with coefficients in a field $\mathbb{F}[z]$. We will use z as the variable for our polynomials.
- Where before we were looking for roots of polynomials with integer coefficients $\mathbb{Z}[x]$ that we wrote as $f(x) = f_d x^d + \dots f_0$ with $f_i \in \mathbb{Z}$, we will now have polynomials in $\mathbb{F}[z, x]$, where we can explicitly write down a polynomial $f(z, x) = f_d(z)x^d + \cdots + f_0(z)$ with $f_i(z) \in \mathbb{F}[z]$.
- Modular reduction works the way we would like in both cases. Where before we were looking for roots of some polynomial $f(x)$ modulo an integer N, now we will be looking for roots of some polynomial $f(z, x)$ modulo a polyomial $N(z) \in \mathbb{F}[z]$.
- Factorization also works the way we want to in both cases. Where in the integer case we have unique factorization into primes, in $\mathbb{F}[z]$ we have unique factorization into irreducible polynomials.
- For the norm of an element $f \in \mathbb{Z}$, we used the absolute value $|f|$. For polynomials, we will use the degree, $|f(z)| = \deg_z f$. This is our first major

difference: the degree is non-Archimedean, where the absolute value for integers is Archimedean.

- For a vector in $\mathbb{Z}^n$, explicitly we had $v = (v_1, \dots, v_n)$ with $v_i \in \mathbb{Z}$. Over the polynomials, our vectors will be in $\mathbb{F}[z]^n$, or explicitly $v = (v_1(z), \dots, v_n(z))$.
- For vector length over $\mathbb{Z}^n$, we used the ℓ_1 and ℓ_2 norms. For $\mathbb{F}[z]^n$ we will define the length to be the maximum degree of a vector $|v| = \max_i \deg v_i(z)$.
- Our integer lattices were $\mathbb{Z}$-modules. For polynomials, we will translate to $\mathbb{F}[z]$-modules. Explicitly, our polynomial lattice over $\mathbb{F}[z]^n$ will be generated by a set of basis vectors $B = \{b_1, \dots, b_n\}$ with $b_i \in \mathbb{F}[z]^n$ (so $b_i = (b_{i,1}(z), \dots, b_{i,n}(z))$ that are linearly independent over $\mathbb{F}[z]$. An element $v \in L(B)$ is any $v = \sum_{i=1}^{n} a(z) b_i(z)$, $a(z) \in \mathbb{F}[z]$.
- We can translate lattice notions fairly straightforwardly. For example, the determinant of a polynomial lattice can be computed as $\det B$ for a basis $B \in \mathbb{F}[z]^{n \times n}$ of L.

6.1.1. Lattice problems over $\mathbb{F}[z]$-modules

Definition 6.1.1. $\mathbb{F}[z]$**-module**: all vectors generated by polynomial combinations of some basis vectors $\{b_1, \dots, b_n\} \in \mathbb{F}[z]^n$.

It turns out that lattice problems over lattices with non-Archimedean norms are much easier than the case of Archimedean norms. In particular, SVP can be solved in polynomial time for a lattice with a non-Archimedean norm. [16]

For polynomial lattices, there are multiple notions of reduction. The definitions below are convenient for our purposes.

Definition 6.1.2. Let the pivot of each row vector be the leftmost element that achieves maximal degree in that vector. A basis is **reduced** if its pivots are all in different columns.

Theorem 6.1.3. *If* $\{b_i\}$ *is a reduced basis for* L, $\deg \det L = \sum_i \deg b_i$.

Theorem 6.1.4 (von zur Gathen, Mulders and Storjohann)**.** *A reduced basis for a polynomial lattice contains* $v \in L$ *of length* $\deg v \leqslant (1/\dim L)(\deg \det L)$. *Such a reduced basis can be computed in polynomial time in the maximum degree of the input basis and the dimension.*

Algorithm 6.1.5 (Lattice Basis Reduction)**.** Repeat until basis is reduced:

(1) Find a pair of vectors with pivots in the same column.
(2) Perform a row operation to reduce the degree of one of them.

Theorem 6.1.6 (Giorgi, Jeannerod, Villard)**.** *Polynomial lattice basis reduction can be performed in*

$$n^{\omega + o(1)} D$$

field operations. ($D = \max_i \deg b_i$, b_i *in input basis,* $\omega \leqslant 2.373$*)*

Theorem 6.1.7. *A reduced basis contains a shortest vector of* L.

- All of this works for any non-Archimedean norm as length.

6.2. Polynomial common divisors Now we can try to translate the problem of finding roots of polynomials modulo divisors to polynomials over $\mathbb{F}[z]$. Let $N(z)$ have total degree n. Then let $f(x) = a(z) + x$. We wish to find a low-degree $r(z)$ such that $f(r(z)) \equiv 0 \bmod p(z)$, with $p(z)|N(z)$.

Theorem 6.2.1 ([5]). *Given* $f(x) = x^d + \cdots + f_0(z)$ *with coefficients in* $\mathbb{F}[z]$, $N(z)$ *of degree* n*, can find all* $r(z)$ *such that*

$$\deg \gcd(f(r(z)), N(z)) \geqslant n\beta$$

$$\deg r(z) \leqslant n\beta^2/d$$

The proof sketch will be as before, translated to polynomials. Let R be the degree bound on the root r that we wish to find, so that $\deg r(z) \leqslant R$.

(1) Choose a set of basis polynomials from $\{x^j f(x)^i N(z)^{k-i}\}$.

(2) For any polynomial $h(x) = h_d(z)x^d + h_{d-1}(z)x^{d-1} + \cdots + h_0(z)$, the vector v_h representing the scaled coefficient embedding of $h(x)$ can be written as $v_h = (h_d(z)z^{dR}, h_{d-1}(z)z^{(d-1)R}, \ldots, h_0(z))$. Then $\deg h(r(z)) \leqslant |v_h|$.

(3) Construct the polynomial lattice basis B of scaled coefficient embeddings of the basis polynomials.

(4) Apply polynomial lattice basis reduction to the lattice basis.

(5) Map the shortest vector in the lattice to a polynomial $Q(x)$.

(6) Bound $\deg Q(r(z)) \leqslant |v_Q| \leqslant (1/\dim L(B)) \deg \det L(B)$. If we can show that $(1/\dim L(B)) \deg \det L(B) < kn\beta$ where $n\beta \leqslant \deg p(z)$ so $n\beta k \leqslant \deg p(z)^k$, then since $Q(r(z)) \equiv 0 \bmod p(z)^k$ and $\deg Q(r(z)) < \deg p(z)^k$, then $Q(r(z)) = 0$, so that $r(z)$ can be found by factoring $Q(x)$.

(7) Factor Q over $\mathbb{F}[z]$ to find its roots.

To establish the theorem, we choose a maximum degree (in x) m for our polynomial basis, and a multiplicity k for our roots. Then we can choose a good set of basis polynomials, compute the determinant and dimension of the lattice basis, and optimize the choice of k and m with respect to β and d.

6.3. Application: Reed-Solomon decoding For our sample application of this theorem, we will give a (polynomial lattice-based) algorithm for decoding Reed-Solomon codes.

Input: $\{(z_1, y_1), \ldots, (z_n, y_n)\}$

Problem: Find all polynomials r of degree less than R such that

$$r(z_i) = y_i$$

for βn pairs of inputs, where $0 < \beta \leqslant 1$.

First, we will put the problem into our framework. Let $N(z) = \prod_i (z - z_i)$. Let $a(z)$ be constructed through interpolation so that $a(z_i) = y_i$ for each i. Then for an $r(z)$ such that $r(z_i) = a(z_i)$, $a(z) - r(z) \equiv 0 \bmod (z - z_i)$. Let S be the set of i where this is the case. Then $a(z) - r(z) \equiv 0 \bmod \prod_{i \in S}(z - z_i)$.

Theorem 6.3.1 (Unique decoding)**.** *For unique decoding, the number of errors that can be tolerated is* $e = n(1 - \beta) < (n - R)/2$.

We will show this with a simplified version of the lattice construction above. For our set of basis polynomials, we will use $f(x) = x - a(z)$, and $N(z)$.

Our explicit lattice basis will be

$$B = \begin{bmatrix} z^R & -a(z) \\ & N(z) \end{bmatrix}$$

Its dimension is $\dim L = 2$ and its determinant is $\det L = z^R N(z)$. Using lattice reduction, we can find a vector $v \in L$ satisfying $|v| \leqslant (R + n)/2$. The degree of the common divisor, which is the number of points matching the codeword, is βn, and the number of non-matching (errored) points is $(1 - \beta)n$.

In order to obtain the theorem, we need to show

$$\begin{aligned} (R + n)/2 &< n\beta \\ (R - n)/2 &< n(\beta - 1) \\ (n - R)/2 &> n(1 - \beta) \end{aligned}$$

To decode beyond the unique decoding limit, we can apply Theorem 6.2.1 and re-obtain the list decoding bound of Guruswami and Sudan [10].

Theorem 6.3.2 (Guruswami Sudan[10])**.** *There is an efficient algorithm to decode up to* $e = (1 - \beta)n < n - \sqrt{nR}$.

These applications illustrate the usefulness of these theorems even in the case when factoring is “easy”. In the Reed-Solomon code application, we have the complete factorization of the modulus $N(z)$, but the difficult part is to find *which* factor our polynomial has a root over. These bounds are also constructive: since we are guaranteed to find all roots modulo any divisor satisfying the theorem, this shows that there are only polynomially many root/divisor combinations satisfying the bounds.

References

[1] D. Aggarwal, J. Li, P. Q. Nguyen, and N. Stephens-Davidowitz, *Slide reduction, revisited - filling the gaps in SVP approximation*, Advances in cryptology – CRYPTO 2020, part ii, August 2020, pp. 274–295. ←4

[2] W. Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Mathematische Annalen **296** (1993), no. 1, 625–635. ←16

[3] D. Boneh and R. Venkatesan, *Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes*, Advances in cryptology – CRYPTO'96, August 1996, pp. 129–142. ←14

[4] T. Chinburg, B. Hemenway, N. Heninger, and Z. Scherr, *Cryptographic applications of capacity theory: On the optimality of Coppersmith's method for univariate polynomials*, Advances in cryptology – ASIACRYPT 2016, part i, December 2016, pp. 759–788. ←10

[5] H. Cohn and N. Heninger, *Ideal forms of coppersmith's theorem and guruswami-sudan list decoding*, Advances in Mathematics of Communications **9** (2015), no. 3, 311. ←9, 12, 18

[6] D. Coppersmith, *Small solutions to polynomial equations, and low exponent RSA vulnerabilities*, Journal of Cryptology **10** (September 1997), no. 4, 233–260. ←5

[7] D. Coppersmith, *Finding small solutions to small degree polynomials*, International cryptography and lattices conference, 2001, pp. 20–31. ←5

[8] N. Gama and P. Q. Nguyen, *Finding short lattice vectors within Mordell's inequality*, 40th annual ACM symposium on theory of computing, May 2008, pp. 207–216. ←4

[9] N. Gama and P. Q. Nguyen, *Predicting lattice reduction*, Advances in cryptology – EUROCRYPT 2008, April 2008, pp. 31–51. ←4

[10] V. Guruswami and M. Sudan, *Improved decoding of reed-solomon and algebraic-geometric codes*, Proceedings 39th annual symposium on foundations of computer science (cat. no.98cb36280), 1998, pp. 28–37. ←19

[11] N. Howgrave-Graham, *Finding small roots of univariate modular equations revisited*, 6th ima international conference on cryptography and coding, December 1997, pp. 131–142. ←5

[12] A. K Lenstra, H. W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen **261** (1982), 515–534. ←3, 4

[13] D. Micciancio and M. Walter, *Practical, predictable lattice basis reduction*, Advances in cryptology – EUROCRYPT 2016, part i, May 2016, pp. 820–849. ←4

[14] P. Q. Nguyen and I. E. Shparlinski, *The insecurity of the elliptic curve digital signature algorithm with partially known nonces*, Designs, Codes and Cryptography **30** (2003), no. 2, 201–217. ←14

[15] P. Q Nguyen and D. Stehlé, *LLL on the average*, International algorithmic number theory symposium, 2006, pp. 238–256. ←4

[16] J. Von zur Gathen, *Hensel and newton methods in valuation rings*, Mathematics of Computation **42** (1984), no. 166, 637–661. ←17

UC San Diego
Email address: `nadiah@cs.ucsd.edu`