Day 5: Lattice-based Cryptography

The main question we've been considering all week actually has an official name:

Definition

The **Shortest Vector Problem** (SVP): Given a basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ for a lattice, find a nonzero lattice vector $\mathbf{l}$ with the shortest possible length.

Yesterday we proved the following fact:

Day 4, Proposition 6

Suppose $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is an LLL-reduced basis for a lattice $\mathcal{L}$. If λ_1 denotes the length of the shortest nonzero vector of $\mathcal{L}$, then $\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{3}}\right)^n \lambda_1$.

So for relatively small-dimensional lattices, $\mathbf{b}_1$ will be pretty close to being the shortest vector. But if we make the dimension big enough, the bound becomes much worse. For example, for $n = 100$, $\mathbf{b}_1$ can be 1.7 million (!!!) times the length of the shortest vector. So eventually even LLL-reduction isn't enough to help us find the shortest vectors!

Brute Force Search

Suppose we have an LLL-reduced basis, and want to find the shortest vector; we can exhaustively search through all possible coefficients of the basis vectors. The key idea is that $\text{Rem}_{S_i}(\mathbf{v})$ is always shorter than $\mathbf{v}$, so if we want $\mathbf{v}$ to be shorter than $\mathbf{b}_1$, then this forces us to make $\text{Rem}_{S_i}(\mathbf{v})$ shorter than $\mathbf{b}_1$ for all i.

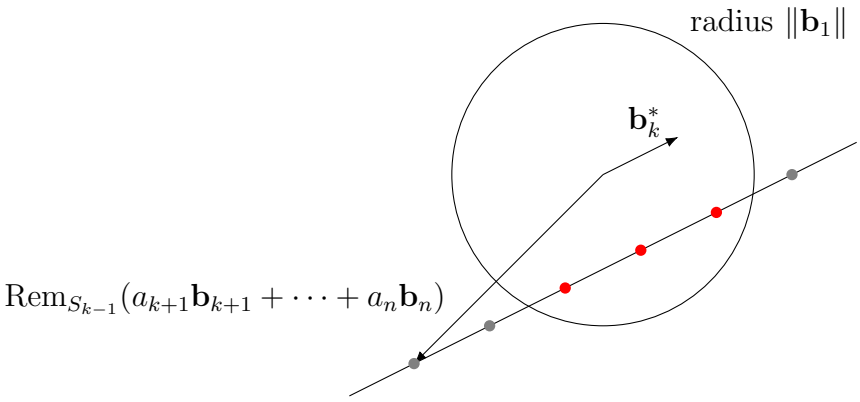
1. Find all integer values of a_n such that $\text{Rem}_{S_{n-1}}(a_n\mathbf{b}_n)$ has length less than $\|\mathbf{b}_1\|$.
2. For each choice of a_n, find all integer values of a_{n-1} such that $\text{Rem}_{S_{n-2}}(a_{n-1}\mathbf{b}_{n-1}+a_n\mathbf{b}_n)$ has length less than $\|\mathbf{b}_1\|$.

$\vdots$. Continue as above: for each choice of $a_n, a_{n-1}, \ldots, a_{k+1}$, find all integer values of a_k such that $\text{Rem}_{S_{k-1}}(a_k\mathbf{b}_k + a_{k+1}\mathbf{b}_{k+1} + \cdots + a_n\mathbf{b}_n)$ has length less than $\|\mathbf{b}_1\|$.

$n+1$. For each of the lists $a_n, \ldots, a_1$ of coefficients found, compute the length of the vector $a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n$, and record the shortest length.

Exploration 1

For any given choice of $a_n, a_{n-1}, \ldots, a_{k+1}$, prove that there are at most $2\|\mathbf{b}_1\|/\|\mathbf{b}_k^*\|$ valid choices of a_k to consider. (Hint: Interpret the diagram below.)



Exploration 2

Use Day 4 Exploration 5 to prove $\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{3}}\right)^{k-1} \|\mathbf{b}_k^*\|$. Use this to prove that the total number of vectors to check by brute force is bounded above by

$$2^n \left(\frac{2}{\sqrt{3}}\right)^{n(n-1)/2}.$$

Exploration 3

This bound is *extremely* bad. Try plugging in $n = 100$ to see how many vectors we would need to check in order to guarantee we find the shortest vector.

In summary, let $\mathbf{v}$ denote the vector returned by the algorithm:

Algorithms that Find Short Vectors	number of steps is polynomial in n	number of steps is exponential in n or worse
$\|\mathbf{v}\|/\lambda_1$ is polynomial in n	**impossible?**	**brute force** (also sieving, etc)
$\|\mathbf{v}\|/\lambda_1$ is exponential in n or worse	**LLL** (also HKZ, BKZ, etc)	(lol who cares)

Importantly, these results don't improve even if we use quantum computers! There is no known quantum algorithm that can find short vectors any faster than a non-quantum algorithm.

Public-Key Cryptography

Suppose Alice owns an online store, and Bob wants to buy something from it. Bob needs to let Alice know his credit card number, but if he just sends the number over the internet in plaintext, anyone would be able to read it! He needs some way to hide his information in such a way that no one but Alice can read it.

To do this, Alice sets up a **public-key cryptosystem**. We will use red for things that Alice doesn't want anyone else to know, and blue for things that Bob doesn't want anyone except Alice to know. The following are known to the public:

- A set M of possible messages (for example, the set of all possible credit card numbers).
- A set C of possible ciphertexts.
- An encryption process[1] that inputs an element of M and outputs an element of C. This process should have the property that a given ciphertext $c \in C$ could have come from at most one $m \in M$.

In addition to these, Alice also has a decryption process that she keeps a secret.

To use the cryptosystem, Bob takes his message m, and encrypts it to a ciphertext c, which is sent through the internet. Alice can use her decryption process to recover the message m. Anyone can read the ciphertext c, but in a good public-key cryptosystem, it should be computationally infeasible[2] for anyone to recover m from c if they don't have the secret decryption process.

Examples:

- RSA: Alice publishes a product $N = pq$ of two primes p and q, but keeps p and q secret. Encryption involves some modular arithmetic mod N. Decryption depends on knowing the factorization of N.
- Elliptic Curve Cryptography (ECC): Alice publishes an elliptic curve E, a point P on E, and a point $Q = kP$ on E, but keeps k secret. Encryption involves using the elliptic curve group law. Decryption depends on knowing what multiple Q is of P (the "discrete log problem").

[1] This may not be a function; there could be a randomized component, so a given message could result in different ciphertexts.

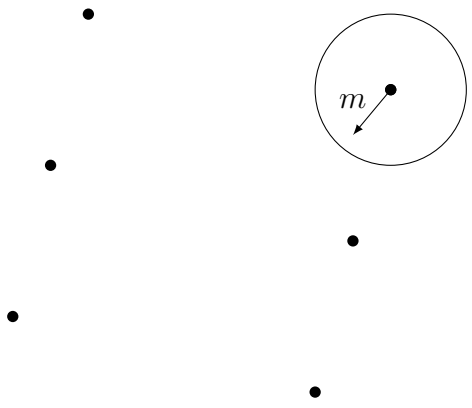
[2] Interpret this as "takes longer than the lifetime of the universe for the fastest supercomputers in the world today."

Essentially the entire internet uses these two cryptosystems.[3] Unfortunately, there is a polynomial time algorithm that can break both of them, if given a decently powerful quantum computer. This has led to a search for new cryptosystems which are secure against quantum attacks.

Lattice-Based Cryptography

Here's an example of a lattice-based public-key cryptosystem.

- Alice generates a lattice $\mathcal{L}$, together with an orthogonal basis $\mathbf{u}_1, \ldots, \mathbf{u}_n$. She keeps $\mathbf{u}_1, \ldots, \mathbf{u}_n$ secret.
- Alice finds a different (very far from orthogonal) basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ generating $\mathcal{L}$, and publishes this basis. She also publishes a number ℓ, which is less than half the length of any $\mathbf{u}_i$.
- Message space M: vectors $\mathbf{m} \in \mathbb{R}^n$ with length shorter than ℓ.
- Ciphertext space C: vectors $\mathbf{c} \in \mathbb{R}^n$.
- Encryption: Bob chooses random coefficients $a_1, \ldots, a_n \in \mathbb{Z}$, and sends $\mathbf{c} = a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n + \mathbf{m}$ (not a lattice vector!)



In order to decrypt $\mathbf{c}$, Alice finds the element $\mathbf{l} \in \mathcal{L}$ closest to $\mathbf{c}$, and computes $\mathbf{l} - \mathbf{c}$.

Exploration 4

How do we know that the closest element to $\mathbf{c}$ in $\mathcal{L}$ must be Bob's lattice vector $a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n$?

[3] To find which system is used by a given website on Google Chrome, you can click on the lock symbol next to a website's URL, click "Certificate," go to "Details," and find the "Public Key" field.

Exploration 5

Given an orthogonal lattice basis $\mathbf{u}_1, \ldots, \mathbf{u}_n$ generating a lattice $\mathcal{L}$, and a vector $\mathbf{v} \in \mathbb{R}^n$ (not in the lattice), explain how to find the vector in $\mathcal{L}$ that is closest to $\mathbf{v}$. (This shows that Alice, knowing the secret basis $\mathbf{u}_1, \ldots, \mathbf{u}_n$, is able to decrypt.)

On the other hand, suppose an eavesdropper comes along, and knows $\mathbf{b}_1, \ldots, \mathbf{b}_n$ and $\mathbf{c}$. Would such an eavesdropper be able to recover $\mathbf{m}$? Doing so would require knowing how to solve the following problem:

Definition

The **Closest Vector Problem** (CVP): Given a basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ for a lattice, and a vector $\mathbf{v}$, find the lattice vector $\mathbf{l} \in \mathcal{L}$ closest to $\mathbf{v}$.

We will now see a *reduction* from SVP to CVP. In other words, breaking the cryptosystem (which requires solving CVP) is *at least as hard* as finding the shortest vector in a lattice.

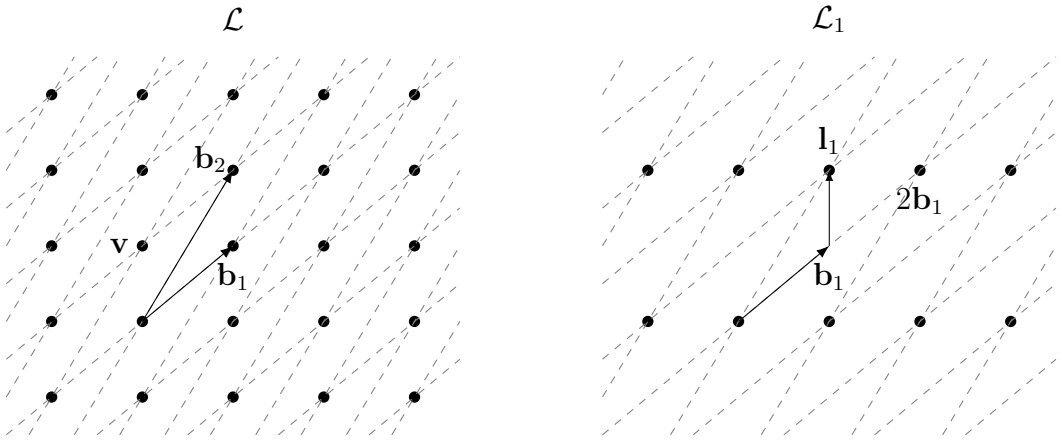
Proposition 6

If someone knows how to solve CVP for an arbitrary basis in $\mathbb{R}^n$, then they can solve SVP for an arbitrary basis in $\mathbb{R}^n$ by solving n instances of CVP.

Proof. Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a basis generating a lattice $\mathcal{L}$, and suppose we want to solve SVP for this basis. For each $1 \leq i \leq n$, consider the modified basis

$$\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n.$$

This generates a new lattice $\mathcal{L}_i$ that is a subset of $\mathcal{L}$, but does not contain $\mathbf{b}_i$. If we can solve CVP, then for each i, we can find the vector $\mathbf{l}_i \in \mathcal{L}_i$ such that $\mathbf{l}_i - \mathbf{b}_i$ (which is a vector in $\mathcal{L}$) is as small as possible. We'll show that if we take the shortest value of $\mathbf{l}_i - \mathbf{b}_i$, this will in fact be the shortest vector in $\mathcal{L}$.



Now suppose $\mathbf{v} = a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n$ is the shortest vector in $\mathcal{L}$ (so $\|\mathbf{v}\| \leq \|\mathbf{l}_i - \mathbf{b}_i\|$ for all i). Then some coefficient a_j must be odd, or else we could divide it by 2 to obtain a shorter vector. We can then write

$$\mathbf{v} = \left(a_1\mathbf{b}_1 + \cdots + a_{j-1}\mathbf{b}_{j-1} + \frac{a_j + 1}{2}(2\mathbf{b}_j) + a_{j+1}\mathbf{b}_{j+1} + \cdots + a_n\mathbf{b}_n\right) - \mathbf{b}_j.$$

That is, $\mathbf{v}$ equals the difference between some lattice vector in $\mathcal{L}_j$ and $\mathbf{b}_j$. But our solution to CVP tells us that $\mathbf{l}_j - \mathbf{b}_j$ is the shortest possible difference between an element of $\mathcal{L}_j$ and $\mathbf{b}_j$. Hence $\|\mathbf{l}_j - \mathbf{b}_j\| \leq \|\mathbf{v}\|$. Thus $\mathbf{l}_j - \mathbf{b}_j$ is the shortest vector in $\mathcal{L}$. □

Since SVP appears to be intractably hard, this proposition suggests that our lattice-based cryptosystem is likely to be secure! This gives us an example of **post-quantum cryptography**: cryptography that is safe even in a world with quantum computers.