Lattices: Math for Cryptography II

They Have Played Us For Absolute Fools

@nebu @Hassam

sigpwny{squeamish_ossifrage}

Outline

- Meta
- Motivation
- Danger: Math Ahead

What this Presentation Is

- An overview of mathematical constructs that are important to cryptography (and common in modern CTFs).
- Intended for people with minimal mathematical background.

What this Presentation Isn't

- A complete guide on understanding linear algebra, lattices, or algorithms that relate to these topics.
- Particularly difficult...if you don't space out.

By the End

- Be able to solve simple cryptography CTF challenges.
- Start to see linear algebra, and lattices, everywhere.

Outline

Meta

Motivation

Danger: Math Ahead

Note on Notation and Assumptions

- $\mathbb{N}$: Natural numbers: $\{1, 2, 3, \ldots\}$
- $\mathbb{Z}$: Integers: $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- The $\prod$ thing we discussed last time, and $\sum$, which are multiplication and addition in for loops, respectively.
- We'll switch a lot between 2 dimensional examples and n-dimensional examples without proof, but trust us --- these generalizations do hold.

How do we attack crypto?

- Attack the implementation
- Attack the math

How do we attack crypto?

- Attack the implementation
- Attack the math

How do we attack crypto?

- Attack the implementation
- Attack the math

A Word of Warning...

Here’s a caricature of Legendre (legend-ray), a very famous number theorist --- he influenced Gauss, who first played with lattices.

Let's Get Started!

Recall that,

Definition

A lattice $\mathcal{L}$ is a discrete subgroup of H generated by all integer combinations of the vectors of some basis B, that is,

$$\mathcal{L} = \sum_{i=0}^{m} \mathbb{Z}\mathbf{b_i} = \left\{ \sum_{i=0}^{m} z_i \mathbf{b_i} \,\middle|\, z_i \in \mathbb{Z}, \mathbf{b_i} \in B \right\}$$

Got You.

Here's how we'll do it:

(These slides are borrowed from Thijs Laarhoven at TU/e)

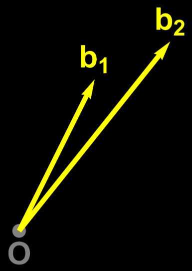
Lattices

What is a lattice?

TU/e

Lattices

What is a lattice?



Some notes about vectors

- A vector is a collection of numbers: $\mathbf{v} = \begin{bmatrix} 27 \\ 15 \end{bmatrix}$

Some notes about vectors

- A vector is a collection of numbers: $\mathbf{v} = \begin{bmatrix} 27 \\ 15 \end{bmatrix}$
- The dot product, $\mathbf{v} \cdot \mathbf{w}$ is the sum of the product of each element in v with each element in w. Example:

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix} = 3 \cdot 2 + 4 \cdot 5 = 26$$

Some notes about vectors

- A vector is a collection of numbers: $\mathbf{v} = \begin{bmatrix} 27 \\ 15 \end{bmatrix}$
- The dot product, $\mathbf{v} \cdot \mathbf{w}$ is the sum of the product of each element in v with each element in w. Example:

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix} = 3 \cdot 2 + 4 \cdot 5 = 26$$

- We define the "length" of a vector by it's norm: $\|\mathbf{v}\| = \sqrt{v \cdot v}$. This is a number, not a vector!

Some notes about vectors

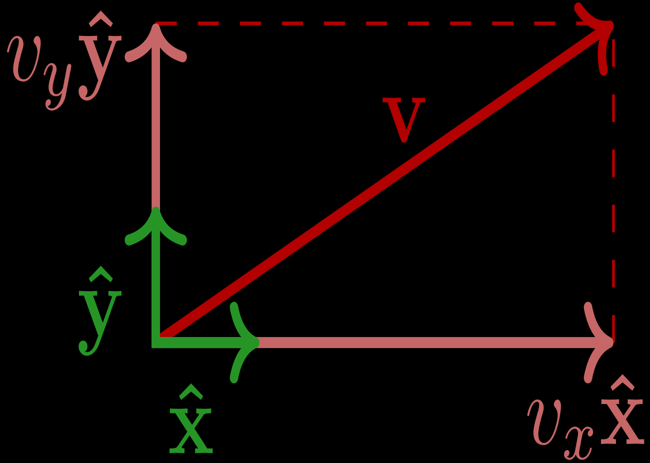
- A vector is a collection of numbers: $\mathbf{v} = \begin{bmatrix} 27 \\ 15 \end{bmatrix}$
- The dot product, $\mathbf{v} \cdot \mathbf{w}$ is the sum of the product of each element in v with each element in w. Example:

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix} = 3 \cdot 2 + 4 \cdot 5 = 26$$

- We define the "length" of a vector by it's norm: $\|\mathbf{v}\| = \sqrt{v \cdot v}$. This is a number, not a vector!
- Recall the Cauchy-Schwarz inequality: $\mathbf{v} \cdot \mathbf{w} \leq \|v\|\|w\|$

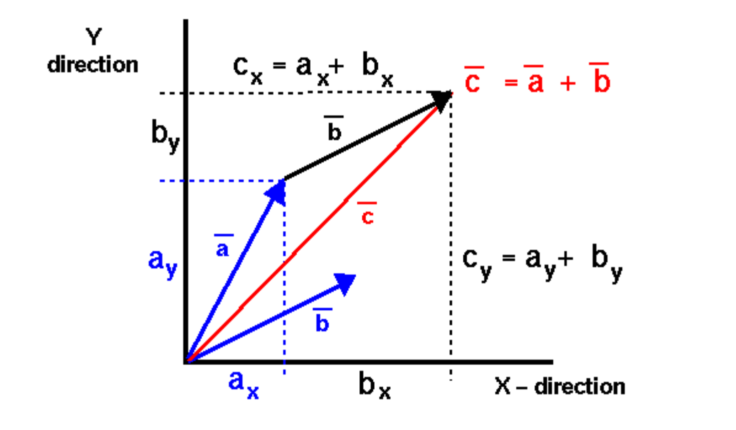
Some notes about vectors

We can split a vector up into its components:



Some notes about vectors

Adding vectors is the same as adding their components.



(Image is shamelessly stolen from NASA)

Some notes about vectors

- Often, we use vectors to represent polynomials.
- What is the result of $\begin{bmatrix} 1 \\ 3 \\ -4 \end{bmatrix} \cdot \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix}$?

Some notes about vectors

- Often, we use vectors to represent polynomials.
- What is the result of $\begin{bmatrix} 1 \\ 3 \\ -4 \end{bmatrix} \cdot \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix}$?
- $x^2 + 3x - 4$

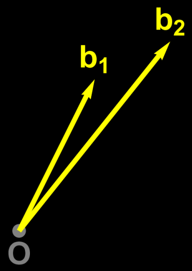
Some notes about vectors

- Often, we use vectors to represent polynomials.
- What is the result of $\begin{bmatrix} 1 \\ 3 \\ -4 \end{bmatrix} \cdot \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix}$?
- $x^2 + 3x - 4$
- Usually, we omit x vector for brevity.

TU/e

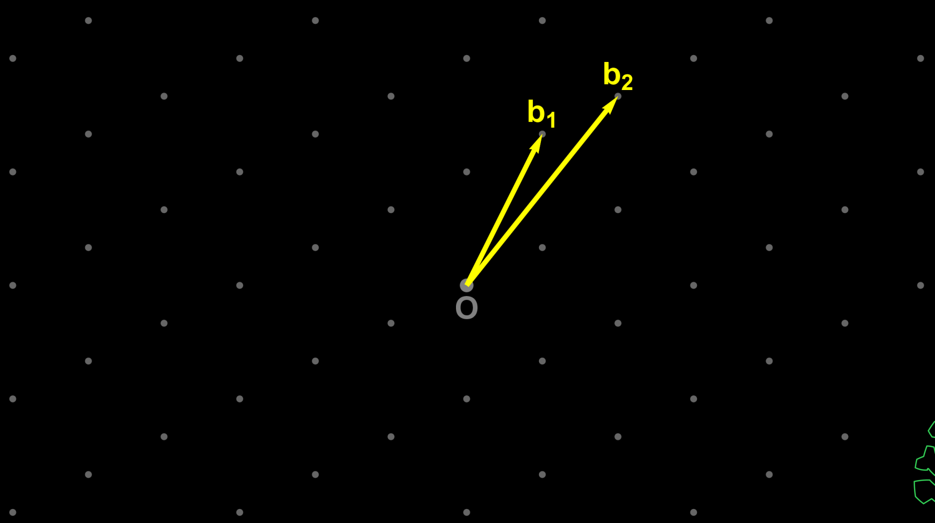
Lattices

What is a lattice?



Lattices

What is a lattice?

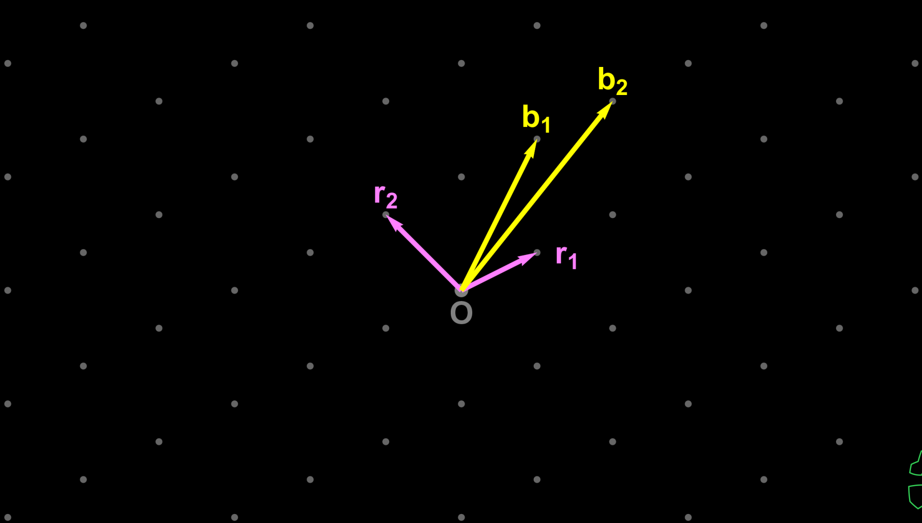


A Grid of Points, Yeah

1. A lattice is indeed an infinite grid of points.
2. It's defined by adding together integral multiples of subsets of the basis vectors.
3. It's a bit different than the n-dimensional space $\mathbb{R}^n$ since that space is *continuous* --- lattices are discrete n-dimensional spaces.
4. A set of basis vectors uniquely defines a lattice, but we can find other sets of bases that define the same lattice.

Lattices

Lattice basis reduction



A Problem for You

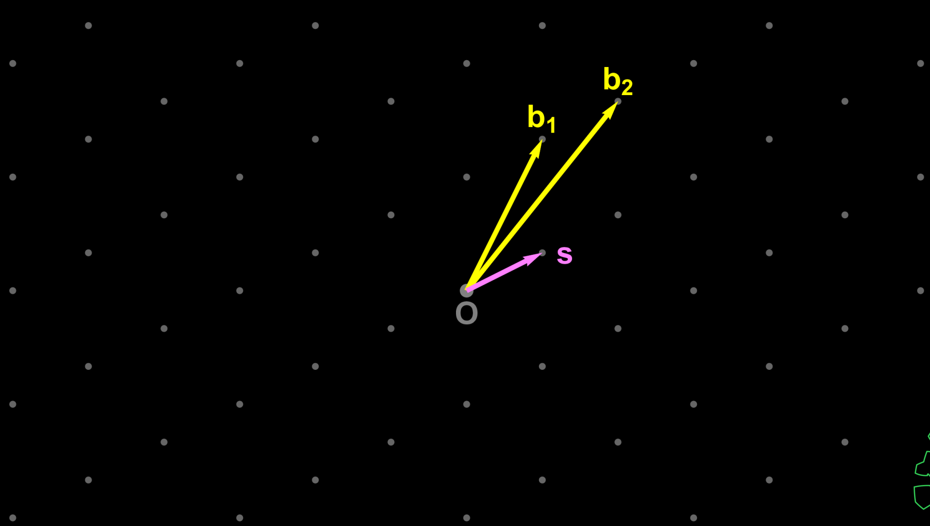
Given a lattice, find the shortest vector (starting at say, the origin) that ends at a grid point.

Note that the shortest vector problem and the smallest reduced basis problem are identical: the shortest of the bases is necessarily the shortest vector in the lattice, since it generates all the others.

TU/e

Lattices

Shortest Vector Problem (SVP)

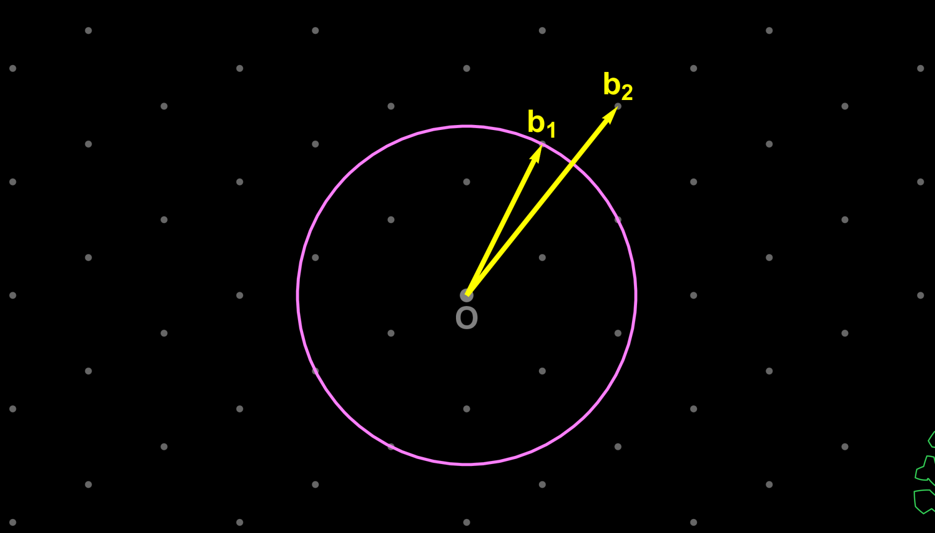


Er...

This problem is NP-complete. Let's explore the (exponential time) way to solve it.

Fincke-Pohst enumeration

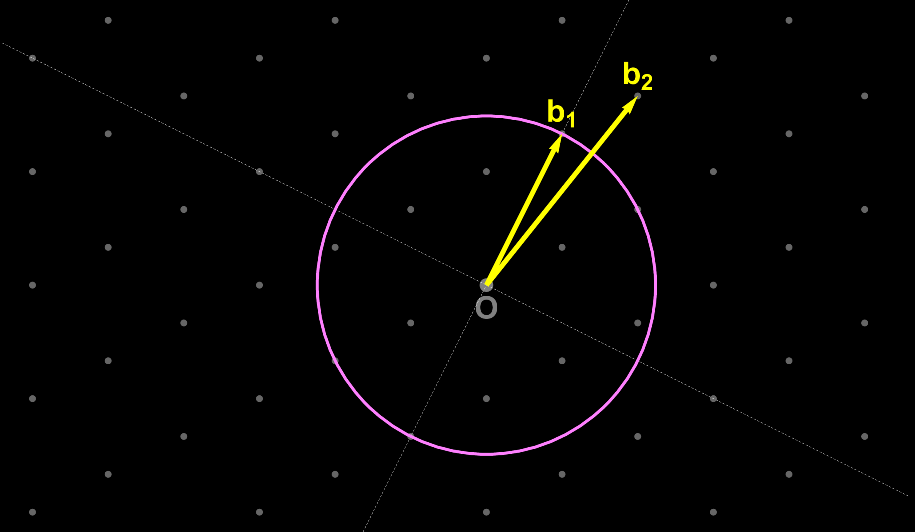
1. Determine possible coefficients of b_2



TU/e

Fincke-Pohst enumeration

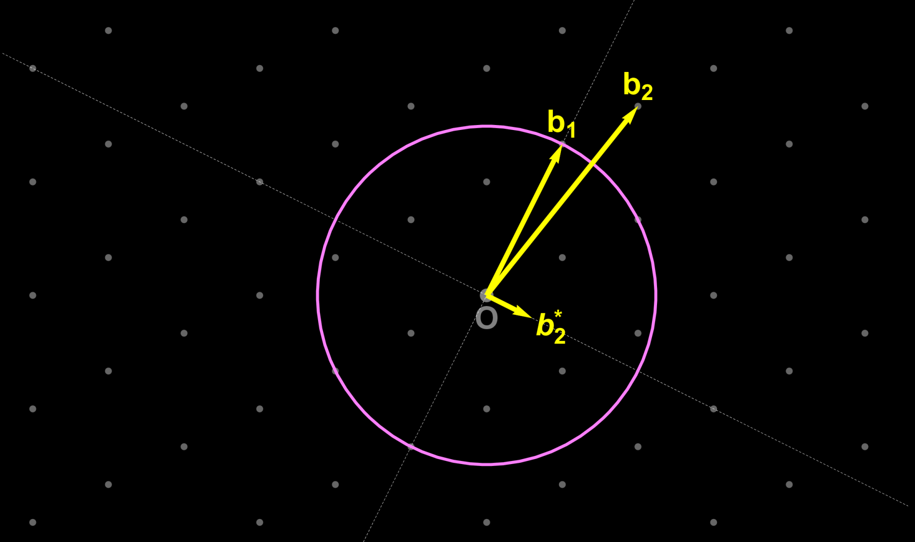
1. Determine possible coefficients of b_2



TU/e

Fincke-Pohst enumeration

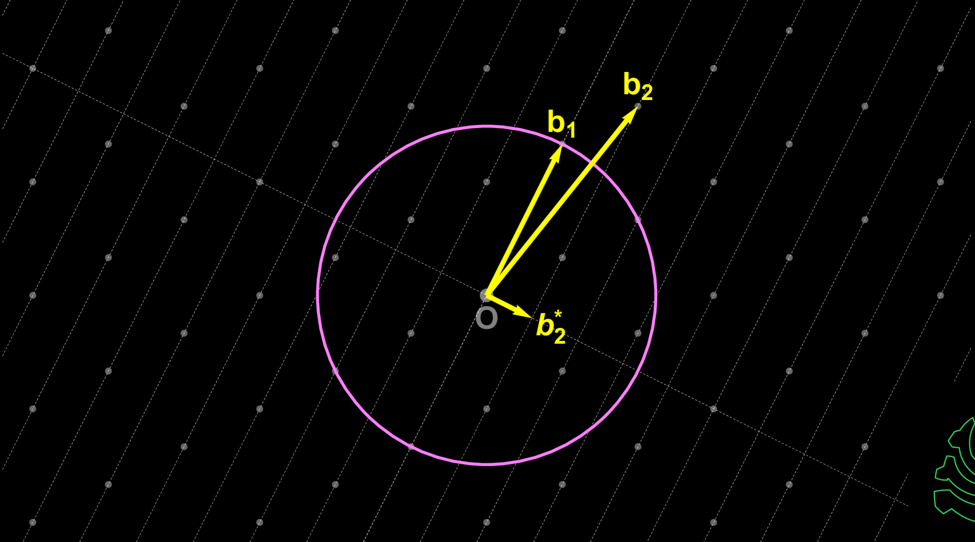
1. Determine possible coefficients of b_2



TU/e

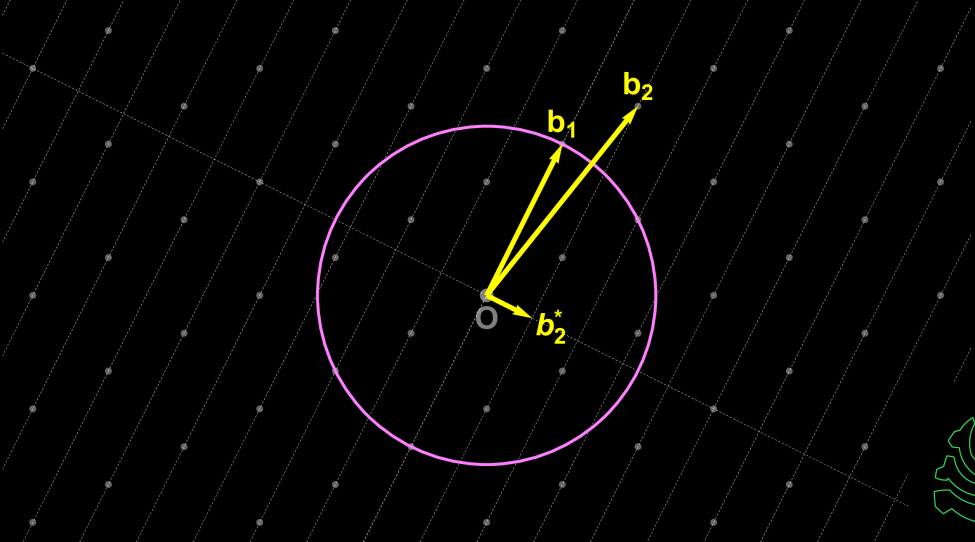
Fincke-Pohst enumeration

1. Determine possible coefficients of b_2



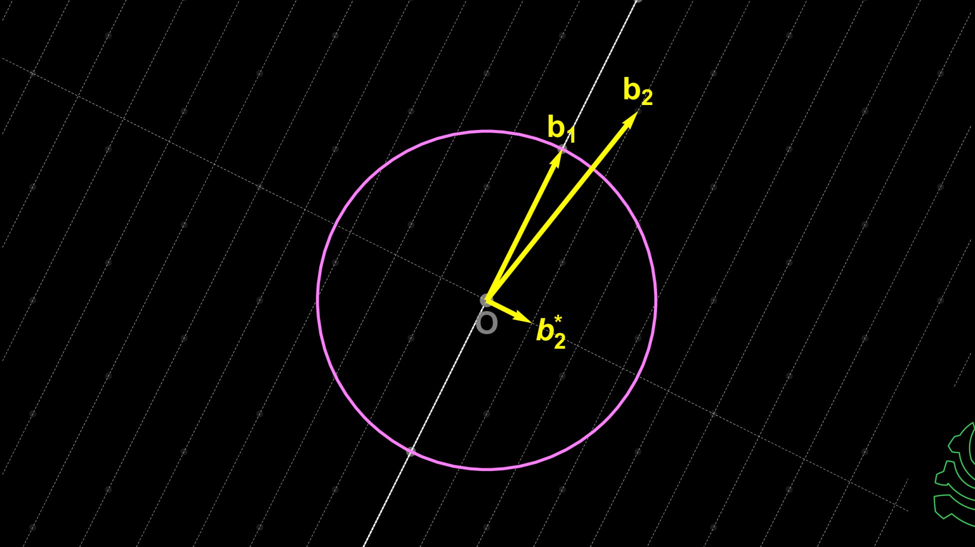
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



Fincke-Pohst enumeration

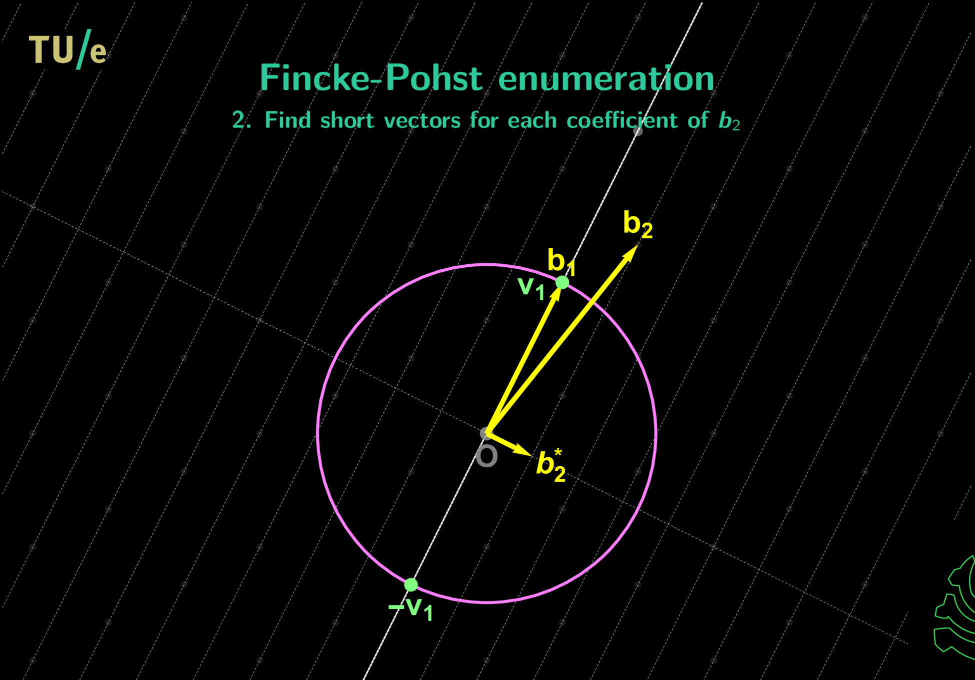
2. Find short vectors for each coefficient of b_2



TU/e

Fincke-Pohst enumeration

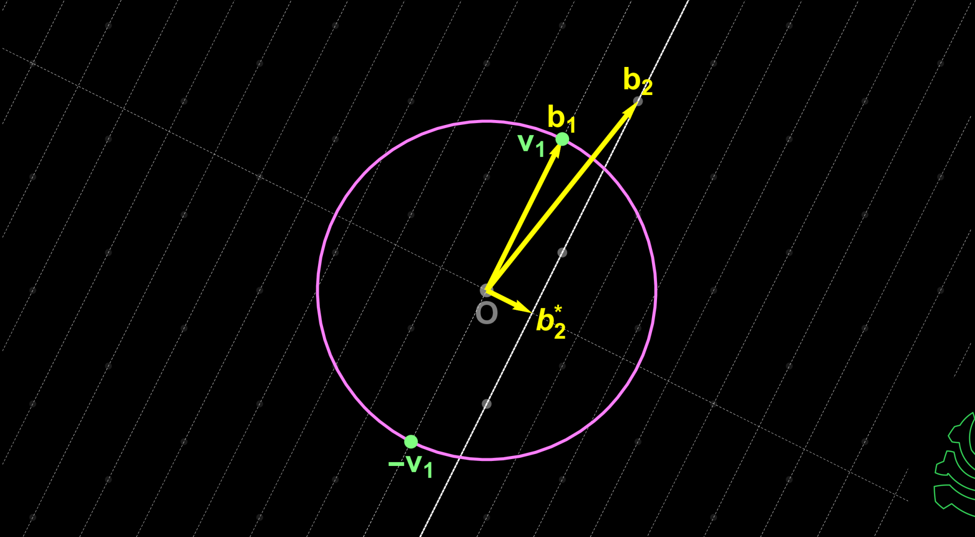
2. Find short vectors for each coefficient of b_2



TU/e

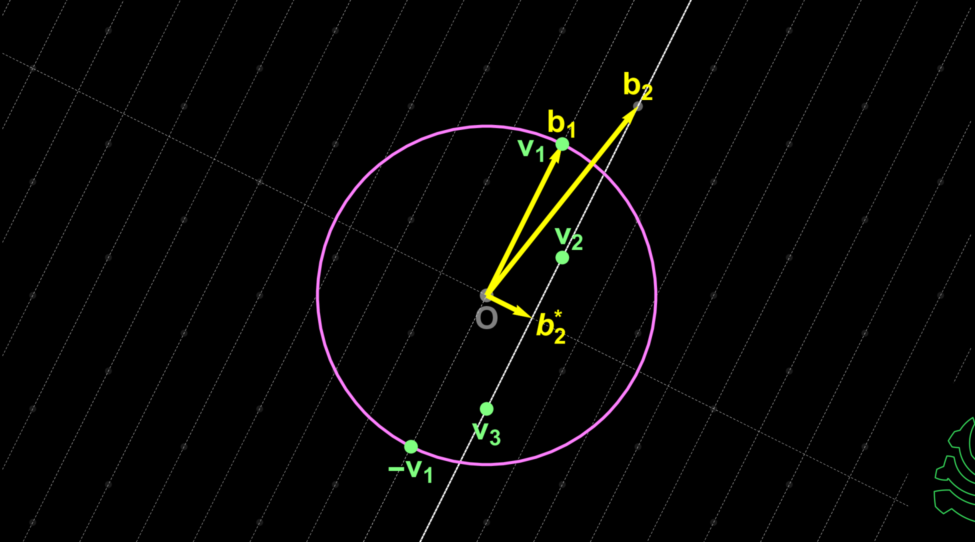
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



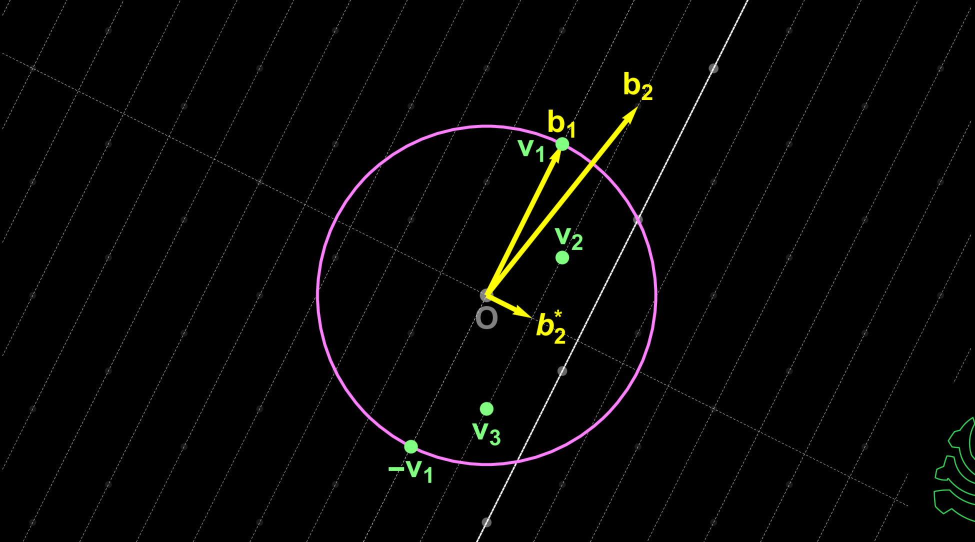
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



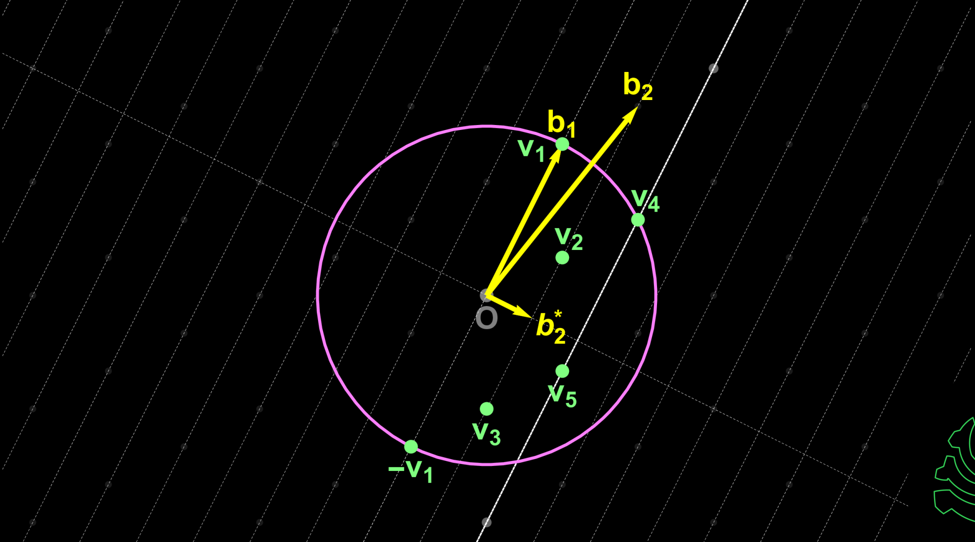
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



Fincke-Pohst enumeration

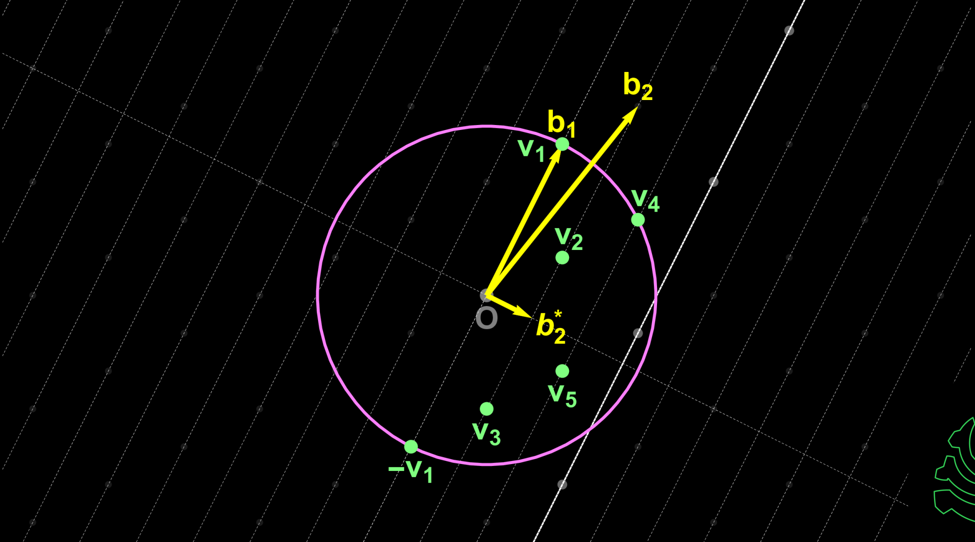
2. Find short vectors for each coefficient of b_2



TU/e

Fincke-Pohst enumeration

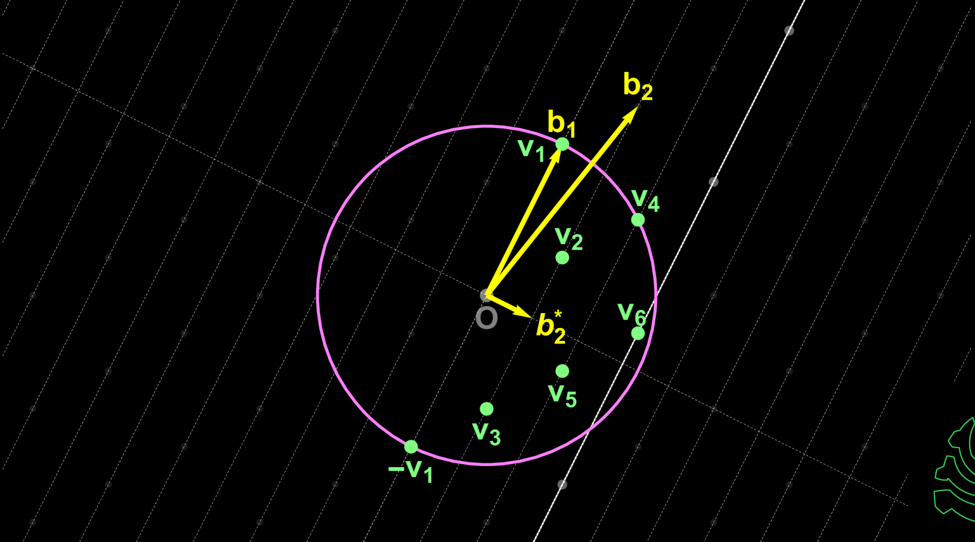
2. Find short vectors for each coefficient of b_2



TU/e

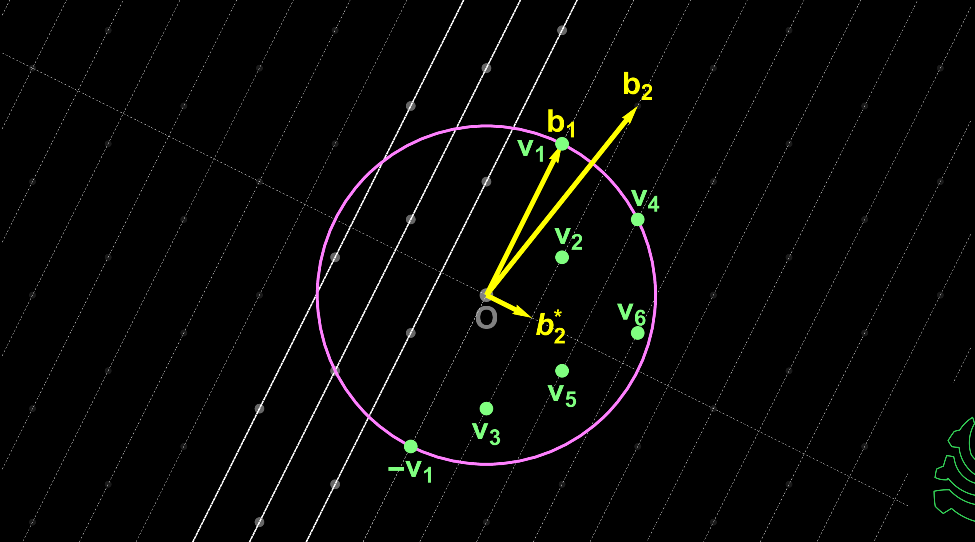
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



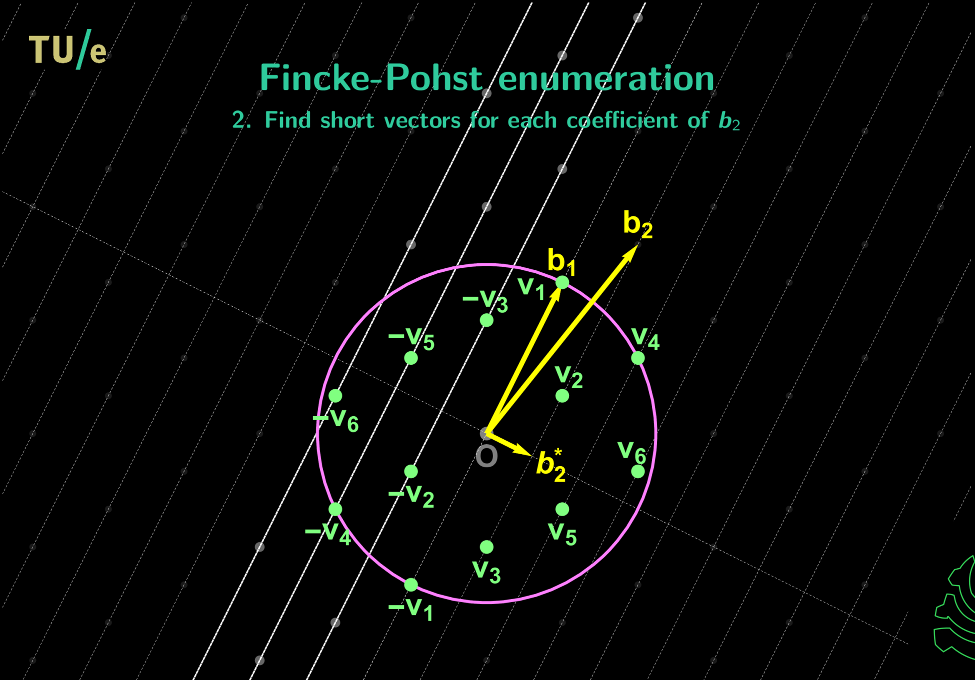
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



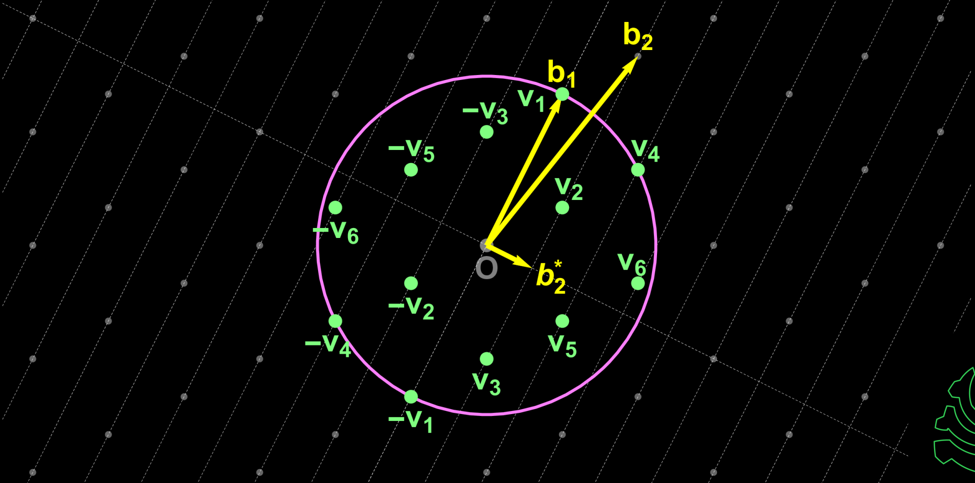
Fincke-Pohst enumeration

2. Find short vectors for each coefficient of b_2



Fincke-Pohst enumeration

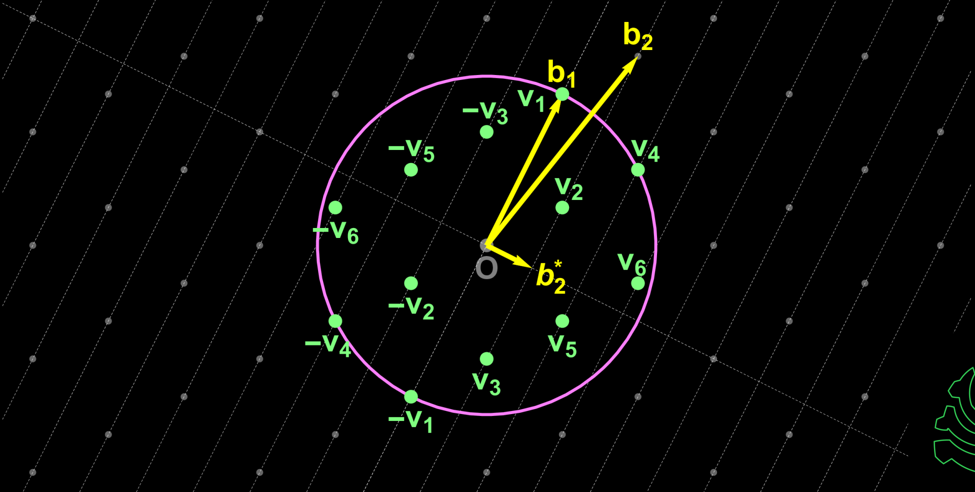
2. Find short vectors for each coefficient of b_2



TU/e

Fincke-Pohst enumeration

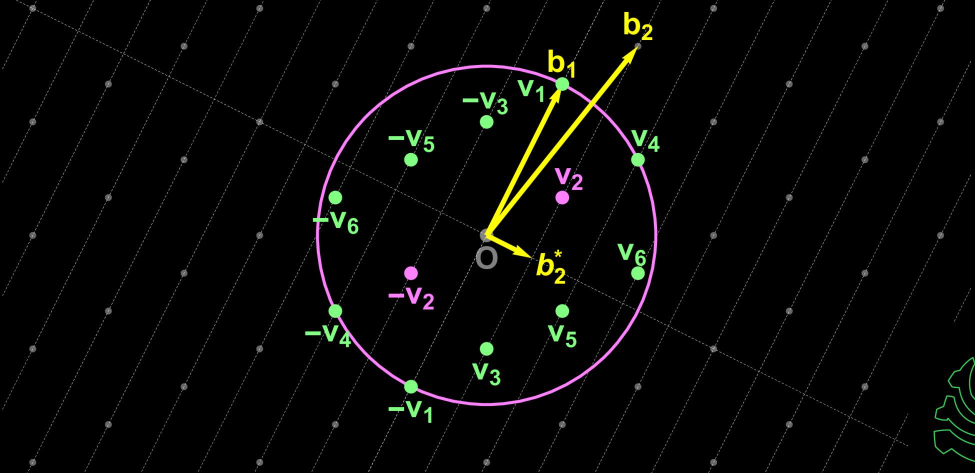
3. Find a shortest vector among all found vectors



TU/e

Fincke-Pohst enumeration

3. Find a shortest vector among all found vectors



Solved!

So we solved (albeit very inefficiently...) the SVP. This is an exponential time algorithm --- but like a lot of NP-complete problems, can be *approximated* in polynomial time.

In short: this hard problem can be approximated easily: we can get a vector that may not be the *shortest*, but is within a factor of the shortest.

LLL

That approximation algorithm for lattice basis reduction (=SVP) is called LLL (Lenstra-Lenstra-Lovász).

Okay, Nebu/Hassam, this is great, but what does it have to do with cryptography?

Sorry, we got carried away...back to some crypto.

Remember RSA? Good.

- We choose two primes (p, q), and compute $N = pq$, and $\phi(N) = (p-1)(q-1)$.
- Then generate (e, d) such that $d \equiv e^{-1} \pmod{\phi(N)}$
- We can encrypt a number m as

$$c \equiv m^e \pmod{N}$$

- We can decrypt a message c as

$$m \equiv c^d \pmod{N}$$

Why is This Secure Again?

All that is sent over is ((N, e), c). Given this tuple, an attacker cannot find m without factoring N (why?).

Let's Factor N

No, not the way @Husnain did it...

We Have Some Information Available

$N = pq$, a product of primes. Factor N.

1. Let's say you have p and q.
2. Oh, that's true, your problem's solved.

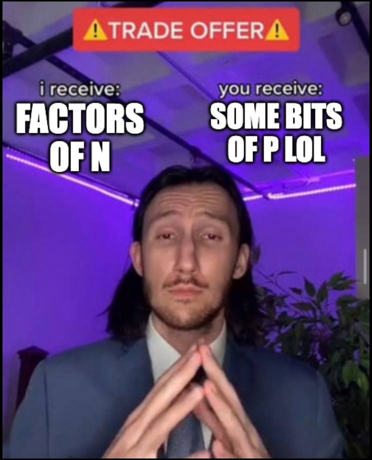
Hm, Maybe A Little Less Information

1. What if I give you just p.

Hm, Maybe A Little Less Information

1. What if I give you just p.
2. Hah! But q is just
$$q = \frac{N}{p}$$
3. You're getting good at this.

What About Now?



You get some bits (enough that you can't brute force, of course) of p --- not all, and nothing about q.

Let's See What We Can Do

Say we know the top a bits of p. Then,

$$p = a + x$$

where x is the unknown bits we have to solve for. As a polynomial:

$$p = f(x) = a + x$$

Can't enumerate all x's, so we pay Amazon to do it for us.

No. We Math.

$$f(x) = p = a + x$$

Let's reframe the polynomial as a modular one:

$$f(x) \equiv a + x \equiv 0 \mod p$$

Guess What?

We're actually much worse at finding roots of modular polynomials than ones over the integers.

Did we just shoot ourselves in the foot?

Outline

- Meta
- Motivation
- Danger: Math Ahead

We Want to Show You Some More Cool Math

We present the Howgrave-Graham (1997, Section 2) method of finding small roots of an irreducible monic modular polynomial.

Don't Get Notation Paralysis!

- polynomial: recall initial slides
- monic: leading coefficient 1 and univariate
- univariate: in one variable
- irreducible: no cheap factoring tricks
- modular: modulo some integer

So a polynomial that looks like this:

$$p(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_o \pmod{N}$$

That's exactly the kind of polynomial we're solving:

$$f(x) = x + a \pmod{p}$$

Howgrave-Graham

Let $p(x)$ be some polynomial

$$p(x) = \sum_{i=0}^{d-1} a_i x^i \pmod{b^k}$$

where $b, k \in \mathbb{N}$. We want to find a root x_0 that is less than a bound X. If the following two conditions are (both) true:

1. $p(x_0) = 0 \pmod{b^k}$
2. $\|p(xX)\| \leq b^k/\sqrt{d}$

Then $p(x_0) = 0$ over all the integers, $\mathbb{Z}$.

Proof

Define two vectors,

$$\mathbf{v} = (1, \quad x_0/X, \;\; x_0^2/X^2, \ldots, \;\; x_0^{d-1}/X^{d-1})$$
$$\mathbf{w} = (a_0, \;\; a_1 X, \quad a_2 X^2, \ldots, \quad a_{d-1} X^{d-1})$$

Confirm that the dot product $\mathbf{v} \cdot \mathbf{w}$ is just $p(x_0)$.

Proof

Define two vectors,

$$\mathbf{v} = (1, \quad x_0/X, \;\; x_0^2/X^2, \ldots, \;\; x_0^{d-1}/X^{d-1})$$
$$\mathbf{w} = (a_0, \;\; a_1 X, \quad a_2 X^2, \ldots, \quad a_{d-1} X^{d-1})$$

Also note that the 'maximum' possible $\mathbf{v}$ is

$$\mathbf{v} = (1, 1, 1, \ldots, 1)$$

Let's Use Cauchy-Schwarz on $\mathbf{v} \cdot \mathbf{w}$

$$\mathbf{v} = (1, \quad x_0/X, \; x_0^2/X^2, \ldots, \; x_0^{d-1}/X^{d-1})$$
$$\mathbf{w} = (a_0, \; a_1 X, \quad a_2 X^2, \ldots, \quad a_{d-1} X^{d-1})$$

$$|\mathbf{w} \cdot \mathbf{v}| < \|w\| \; \|v\|$$

Here,

$$\|w\| \leq \sqrt{1 + 1 + \cdots + 1} = \sqrt{d}$$

$$\|v\| = \sqrt{a_0^2 + a_1^2 X^2 + \cdots + a_{d-1}^2 X_{d-1}^2} = \|p(xX)\|$$

Let's Use Cauchy-Schwarz on $\mathbf{v} \cdot \mathbf{w}$

$$\mathbf{v} = (1, \quad x_0/X, \; x_0^2/X^2, \ldots, \; x_0^{d-1}/X^{d-1})$$
$$\mathbf{w} = (a_0, \; a_1 X, \quad a_2 X^2, \ldots, \quad a_{d-1} X^{d-1})$$

$$|\mathbf{w} \cdot \mathbf{v}| < \|w\| \; \|v\|$$

Here,

$$\|w\| \leq \sqrt{1 + 1 + \cdots + 1} = \sqrt{d}$$
$$\|v\| = \|p(xX)\| \leq b^k/\sqrt{d}$$

by condition (2).

Let's Use Cauchy-Schwarz on $\mathbf{v} \cdot \mathbf{w}$

$$|\mathbf{w} \cdot \mathbf{v}| < \|w\| \; \|v\|$$

$$\|w\| \leq \sqrt{d}$$

$$\|v\| = \|p(xX)\| \leq b^k/\sqrt{d}$$

Finally, using Cauchy-Schwarz:

$$|p(x_0)| = |\mathbf{w} \cdot \mathbf{v}| < \sqrt{d}\,\frac{b^k}{\sqrt{d}} = b^k$$

Wait, So What I'm Saying Is...

$p(x_0) \equiv 0 \pmod{b^k}$ and $|p(x_0)| < b^k$ mean that...

Wait, So What I'm Saying Is...

$p(x_0) \equiv 0 \pmod{b^k}$ and $|p(x_0)| < b^k$ mean that...$p(x_0)$ *must be* $= 0$ over all integers.

And *now*, this modular polynomial is easy to solve since it's in the integers.

Wait, But What Was The Point?

$f(x) \equiv 0 \mod p$ doesn't satisfy this theorem's bounds anyway.
Recall that for polynomial $p(x)$ being solved for,
$\|p(xX)\| \leq b^k/\sqrt{d}$, which f doesn't satisfy.

A possible solution

What if we can construct another polynomial $g(x)$ using $f(x)$ that has the same root x_0 we care about, but is "smaller" and therefore:

- Obeys the bounds of the theorem, and thus
- Roots are easy to enumerate since it's over the integers instead of a modulo.

How do we create $g(x)$?

We want a $g(x)$ such that $g(x_0) \equiv 0 \pmod{b^k}$ (and of course that it works with Howgrave-Graham). Or,

$$g(x) \in b^k\mathbb{Z}$$

Point is, none of these operations change the meaning of g, since $f(x_0) = 0 \mod b^k$ for the root we want.

How do we create $g(x)$?

We want a $g(x)$ such that $g(x_0) \equiv 0 \pmod{b^k}$ (and of course that it works with Howgrave-Graham). Or,

$$g(x) \in b^k\mathbb{Z}$$

$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z}$$

Point is, none of these operations change the meaning of g, since $f(x_0) = 0 \mod b^k$ for the root we want.

How do we create $g(x)$?

We want a $g(x)$ such that $g(x_0) \equiv 0 \pmod{b^k}$ (and of course that it works with Howgrave-Graham). Or,

$$g(x) \in b^k\mathbb{Z}$$
$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z}$$
$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z} + b^k x f(x)\mathbb{Z}$$

Point is, none of these operations change the meaning of g, since $f(x_0) = 0 \mod b^k$ for the root we want.

How do we create $g(x)$?

We want a $g(x)$ such that $g(x_0) \equiv 0 \pmod{b^k}$ (and of course that it works with Howgrave-Graham). Or,

$$g(x) \in b^k\mathbb{Z}$$
$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z}$$
$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z} + b^k x f(x)\mathbb{Z}$$
$$g(x) \in b^k\mathbb{Z} + b^k f(x)\mathbb{Z} + b^k x f(x)\mathbb{Z} + b^k x^2 f(x)\mathbb{Z} + \cdots$$

Point is, none of these operations change the meaning of g, since $f(x_0) = 0 \mod b^k$ for the root we want.

And Now, The Voilà

Remember the shortest vector problem?

We can put in coefficients to $(xX)^k f(x)$ as $k \in \mathbb{Z}$ bases for a lattice into LLL and find (one of) the shortest vector bases for the lattice.

Wait, what just happened?

Remember, vectors can represent polynomials.

We've made a bunch of polynomials that have the same properties as this ideal $g(x)$, but they're not the right size for Howgrave-Graham. But, using LLL, we can find a small enough vector that it might work!

To Summarize

- We know $r \leq 2^{\texttt{evil}} = X$, where 'evil' is the number of bits that were wiped out. This is an upper bound.
- Construct matrix M so that:

$$M = \begin{bmatrix} X^2 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & N \end{bmatrix}$$

- Run LLL on this matrix, interpreting rows as basis vectors, get output $g(xX)$.
- Use the shortest vector as coefficients of $g(x)$, which is *not* a modular polynomial.
- Enumerate roots r_i of $g(x)$ and check for $(a + r_i) \mid N$.

Whaddya Mean, That's Voilà?

Two observations:

1. Our lattice is made up of integral multiples of the basis vectors, which are coefficients of $f(xX)$. The shortest basis generated by LLL, interpreted as coefficients of $g(xX)$, will have a set of solutions on this lattice as well: so we retain all the information $f(x)$ gives us.
2. Recall that Howgrave-Graham requires that $\|g(xX)\| \leq b^k/\sqrt{d}$, which is true, since the norm of the shortest output vector basis from LLL is, well, short.[1]

[1] Not exactly accurate --- if you figure out the bounds correctly, it's short enough.

Whaddya Mean, That's Voilà?

Thus, we can take $g(Xx)$, remove the X terms to get $g(x)$. Since it satisfies Howgrave-Graham, solve it over the integers (easy), and check all the roots we get for dividing N. More precisely,...

To Summarize

The matrix M was constructed with values $xXf(xX)$, $f(xX)$, and $b^k\mathbb{Z} = N$. Recall $f(x) = a + x$. These written out are:

$$\begin{aligned} xXf(xX) &= (xX)^2 + Xxa + 0 \\ f(xX) &= 0 + xX + a \\ & \ 0 + 0 + N \end{aligned}$$

Taking the coefficients and putting them into a matrix:

$$M = \begin{bmatrix} X^2 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & N \end{bmatrix}$$

That's where M came from: it generates the lattice of the coefficients scaled by X^k.

Okay, how do we use this?

What if we can leak some information about (p, q)?

```
p, q, e = random_prime(2^512), random_prime(2^512), 17
n, tot = p*q, (p-1)*(q-1)
d = inverse_mod(e, tot)

print(p - p % 2^100, q - q % 2^100)
```

We can recover p and q even though we're missing a lot of their bits!