Lattices and Fully Homomorphic Encryption

Nigel P. Smart

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB.

June 1, 2015

Lattices

A lattice (for this talk) will be an $\mathbb{Z}$-submodule of $\mathbb{R}^n$ of rank n.

Given a set of basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ as column vectors we define the matrix

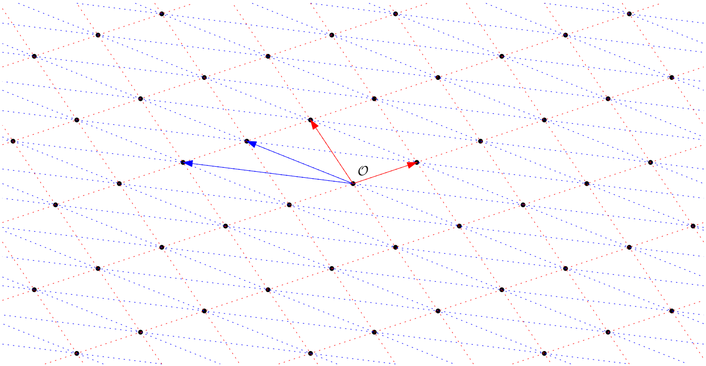
$$B = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{n \times n}.$$

The lattice generated by B is given by

$$\begin{aligned} \mathcal{L}(B) &= \{B \cdot \mathbf{z} : \mathbf{z} \in \mathbb{Z}^n\} \\ &= \{\sum_{i=1}^{n} z_i \cdot \mathbf{b}_i : z_i \in \mathbb{Z}\}. \end{aligned}$$

Lattice Basis

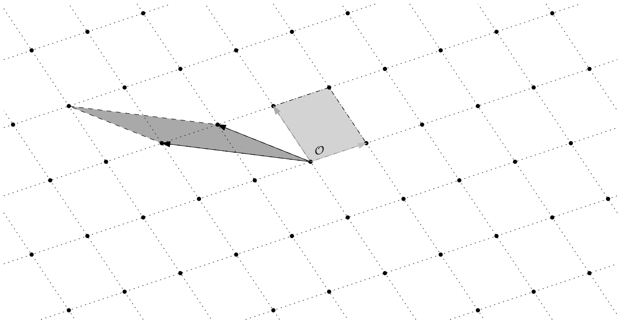
A lattice basis is *not unique*



The red basis is a nice one, the blue basis is a bit horrible

Parellelpiped

No matter what the basis we choose the volume of the region defined by the basis vectors is the same.

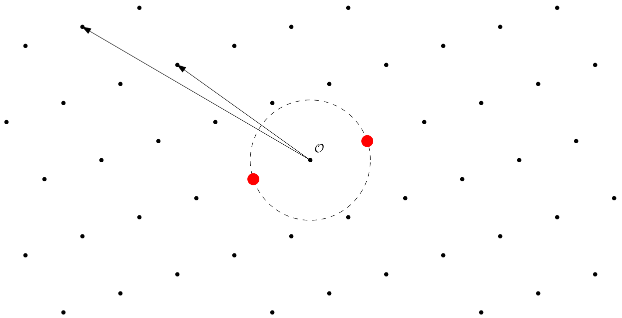


We call this value the lattice determinant

$$\Delta(\mathcal{L}(B)) = |\det(B)|.$$

Shortest Vector Problem

Since a lattice is discrete there is a well defined notion of a shortest non-zero vector

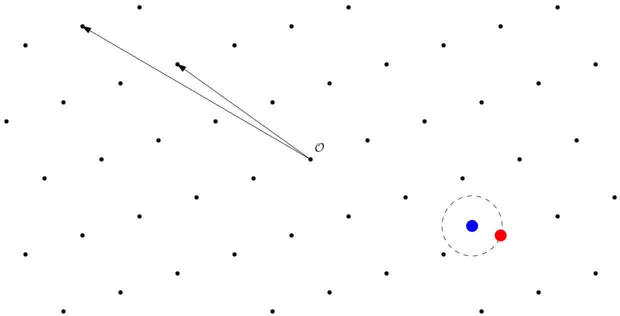


In general, for large enough dimension, finding even *a* short vector (as opposed to *the* shortest vector), is hard.

- Called the Shortest Vector Problem (SVP).
- Expected size of shortest vector $\approx \Delta^{1/n}$.

Closest Vector Problem

Given a general point (in blue) we can ask to find the closest lattice vector to that point (in red).



- Called the Closest Vector Problem (CVP).

Bounded Distance Decoding

In the CVP problem we are given

- A basis B
- A vector $\mathbf{v} \in \mathbb{R}^n$.

We are asked to find $\mathbf{x} \in \mathcal{L}(B)$ such that

$$|\mathbf{x} - \mathbf{v}|$$

is minimised.

- i.e. we need to find $\mathbf{z} \in \mathbb{Z}^n$ which minimises the size of the vector $\mathbf{e} = B \cdot \mathbf{z} - \mathbf{x}$.

Now suppose we are given a *promise* that such a $\mathbf{e}$ exists with $|\mathbf{e}| \leq \gamma$.

- This is the potentially easier Bounded Distance Decoding Problem (BDD)

Note, the smaller γ is then the easier this becomes.

Link to Coding Theory

Suppose now B was an integer matrix with more rows than columns

- $B \in \mathbb{Z}^{n \times m}$ with $n \gg m$,.

We can consider the *code* (modulo q) generated by B

$$\mathcal{C}(B) = \{B \cdot \mathbf{z} \pmod{q} : \mathbf{z} \in \mathbb{Z}_q^m\}.$$

This defines an m-dimensional lattice in $\mathbb{R}^n$.

Suppose we transmit a codeword $\mathbf{c} \in \mathcal{C}(B)$.

The codeword gets some error in transmission $\mathbf{x} = \mathbf{c} + \mathbf{e}$.

- Where $\mathbf{e}$ is “small”

We want to decode $\mathbf{x}$ to recover $\mathbf{c}$.

- Decoding problem for random linear codes modulo q.
- Essentially the BDD problem for the associated lattice.

Another Look At The Decoding Problem

We are given a matrix B (which we can think of as $n \times m$ with entries modulo q).

Someone gives us a value $\mathbf{x} = B \cdot \mathbf{z} \pmod{q}$ for $\mathbf{z} \in \mathbb{Z}_q^m$.

We can easily solve for **z** by standard Gaussian elimination

As soon as we are given $\mathbf{x} + \mathbf{e}$, for some small n-dimensional error vector **e**, it becomes hard

Called the Learning With Errors problem or LWE.

Error Distributions

At many points we shall want our error vectors **e** to come from some distribution.

We shall call this distribution $\mathcal{D}_n$, just to hide the details.

- In practice it could output vectors in $\mathbb{Z}^n$ with coefficients bounded by some small value γ
- Or vectors distributed like a discrete Gaussian with small standard deviation γ

In any case the distribution samples “small” vectors.

Cryptographic LWE

In cryptography we are interested more in decision problems,

Suppose we have a black box which executes the following code on input of q, n, m and $\mathcal{D}_n$

- Pick $A \in \mathbb{Z}_q^{n \times m}$.
- Pick $b \in \{0, 1\}$
- Pick $\mathbf{s} \in \mathbb{Z}_q^m$.
- Pick $\mathbf{r} \in \mathbb{Z}_q^n$
- Pick $\mathbf{e}$ according to distribution $\mathcal{D}_n$.
- If $b = 0$ then set $\mathbf{b} = A \cdot \mathbf{s} + \mathbf{e}$
- Else if $b = 1$ then set $\mathbf{b} = \mathbf{r}$.
- Output $(A, \mathbf{b})$.

The *decision LWE problem* is to work out whether the box has chosen $b = 0$ or $b = 1$.

Basic (secret key) LWE Encryption

The secret key is the value $\mathbf{s} \in \mathbb{Z}_q^m$.

To encrypt a message $\mathbf{m} \in \mathbb{Z}_p^n$, for some modulus $p \ll q$, we output $(A, \mathbf{b})$ where

- $A \in \mathbb{Z}_q^{n \times m}$ is random
- $\mathbf{b} = A \cdot \mathbf{s} + \mathbf{m} + p \cdot \mathbf{e} \pmod q$ where $\mathbf{e} \in \mathcal{D}_n$.

To decrypt we execute

$$(\mathbf{b} - A \cdot \mathbf{s} \pmod q) \pmod p.$$

This is semantic secure assuming LWE is hard

- Cannot distinguish a valid ciphertext $(A, \mathbf{b})$ from a random tuple $(A, \mathbf{r})$.

Adding Some Structure

Having big matrices is not very good in practice so instead we use polynomials as follows:

Can think of the set of polynomials with integer coefficients of degree less than n as defining the same lattice as $\mathbb{Z}^n$.

- A polynomial $a(X)$ corresponds to its vector of coefficients $\mathbf{a}$.

Now take the ring of polynomials modululo a fixed degree n polynomial

$$R = \mathbb{Z}[X]/F(X)$$

- Ring also forms a lattice in $\mathbb{Z}^n$.
- But now we can "multiply" lattice elements by each other

We can also take things modulo q, i.e. $R_q = \mathbb{Z}_q[X]/F(X)$ and still get an n-dimensional lattice.

Ring LWE

Given a polynomial $a(X) \in R$ (or R_q) we can associate the matrix M_a such that

$$a(X) \cdot b(X) = M_a \cdot \mathbf{b} \pmod{F(X)}.$$

Now think of LWE with A replaced by M_a, we can write it in terms of polynomials

Let $\mathcal{D}$ now pick *polynomials* with small coefficients.

Now we have an interactive problem, the adversary has a box holding secret (fixed) values

- $s \in R_q$
- $b \in \{0, 1\}$.

Adversary is asked to determine b given polynomially many calls to the box

Ring LWE Box

The box performs the following operations on each call

- $a \in R_q$.
- $r \in R_q$
- Pick e according to distribution $\mathcal{D}$ from R_q.
- If $b = 0$ then set $b = a \cdot s + e \pmod{F(X), q}$
- Else if $b = 1$ then set $b = r$.
- Output (a, b).

This is the polynomnial variant of LWE

Our encryption scheme now takes messages in R_p and encrypts via

$$b = a \cdot s + m + p \cdot e$$

Public Key Scheme

To make a public key scheme we give a public key which allows the encryptor to generate *many* encryptions of zero

KeyGen:

- Pick s and e according to $\mathcal{D}$ from R_q
- $a \in R_q$.
- $b = a \cdot s + p \cdot e$ in R_q
- Private key : s
- Public key : (a, b).

Note the public key *is* an encryption of zero.

Have selected s to be "small" for reasons to be seen later.

Public Key Scheme

To encrypt $m \in R_p$

Encryption:

- Pick v, e_0 and e_1 from $\mathcal{D}$.
- $c_0 = b \cdot v + p \cdot e_0 + m$.
- $c_1 = a \cdot v + p \cdot e_1$.

Think of v as “new” secret key

- By LWE assumption c_0 looks random in R_p, same for c_1.

Decryption

$$\begin{aligned} c_0 - s \cdot c_1 &= ((a \cdot s + p \cdot e) \cdot v + p \cdot e_0 + m) - s \cdot c_1 \\ &= a \cdot v \cdot s + p \cdot (e \cdot v + e_0) + m - s \cdot (a \cdot v + p \cdot e_1) \\ &= m + p \cdot (e \cdot v + e_0 - e_1 \cdot s) \\ &= m + p \cdot \text{“small”}. \end{aligned}$$

Works since s and v are “small”

Additively Homomorphic

Our scheme is additively homomorphic.

Let (c_0, c_1) encrypt $m \in R_p$ and (c_0', c_1') encrypt $m' \in R_p$

Define following operation on *ciphertexts*

$$(c_0, c_1) \oplus (c_0', c_1') = (c_0 + c_0', c_1 + c_1') = (d_0, d_1)$$

then (d_0, d_1) encrypts $m + m'$ in R_p since

$$\begin{aligned} d_0 - s \cdot d_1 &= (c_0 - s \cdot c_1) + (c_0' - s \cdot c_1') \\ &= (m + p \cdot \text{small}) + (m' + p \cdot \text{small}') \\ &= (m + m') + p \cdot (\text{small} + \text{small}'). \end{aligned}$$

Multiplicatively Homomorphic Version 1

Define the tensor product of the ciphertexts

$$(c_0, c_1) \otimes (c_0', c_1') = (c_0 \cdot c_0',\ c_0 \cdot c_1',\ c_1 \cdot c_0',\ c_1 \cdot c_1') = (d_0, d_1, d_2, d_3)$$

"Normal" decryption we can think of as a vector dot-product

$$(c_0, c_1) \cdot (1, -s)^{\mathsf{T}} = c_0 - s \cdot c_1$$

Form the tensor product of the"vector" secret key with itself

$$(1, -s) \otimes (1, -s) = (1, -s, -s, s^2).$$

Now decrypt the tensor ciphertext with the tensor secret key

$$\begin{aligned}(d_0, d_1, d_2, d_3) \cdot (1,\ -s,\ -s,\ s^2)^{\mathsf{T}} &= d_0 - s \cdot d_1 - s \cdot d_2 + s^2 \cdot d_3 \\ &= \ldots blah \ldots blah \ldots \\ &= m_0 \cdot m_1 + p \cdot \text{“small”}^2.\end{aligned}$$

Here we have assumed $p \ll q$ so the small really is small with respect to q.

Multiplicatively Homomorphic Version 2

But we have now got a four component ciphertext.

The first simplification is to reduce to a three component ciphertext, by replacing $\otimes$ with the operation

$$(c_0, c_1) \odot (c'_0, c'_1) = (c_0 \cdot c'_0,\ c_0 \cdot c'_1 + c_1 \cdot c'_0,\ c_1 \cdot c'_1) = (d_0, d_1, d_2)$$

This three component ciphertext will decrypt via the secret key vector $(1,\ -s,\ s^2)$, since

$$(d_0, d_1, d_2) \cdot (1,\ -s,\ s^2)^{\mathsf{T}} = m_0 \cdot m_1 + p \cdot \text{“small”}^2.$$

Multiplicatively Homomorphic Version 3

Now add to the secret key an "encryption" of s^2

$$(a', b') = (a', a' \cdot s + p \cdot e' + s^2)$$

This is a bit of a cheat

- Plaintext space is really R_p
- s is in R_q.
- We think of s^2 as lying in R_q
- So not even encrypting something in the plaintext space!

To define new ciphertext multiplication we take our three component ciphertext (d_0, d_1, d_2) and set

$$e_0 = d_0 + b' \cdot d_2$$
$$e_1 = d_1 + a' \cdot d_2.$$

Multiplicatively Homomorphic Version 3

Now we have

$$\begin{aligned} e_0 - s \cdot e_1 &= d_0 + \left(a' \cdot s + p \cdot e' + s^2\right) \cdot d_2 - d_1 \cdot s - a' \cdot d_2 \cdot s \\ &= \left(d_0 - d_1 \cdot s + d^2 \cdot s^2\right) + p \cdot e' \cdot d_2 \\ &= m_0 \cdot m_1 + p \cdot \text{“small”}^2 + p \cdot e' \cdot d_2 \end{aligned}$$

Problem is that $e' \cdot d_2$ is not “small”

Two Solutions:

- Use a bit decomposition to produce e_0 and e_1
- Temporarily replace q by a bigger modulus Q

The latter seems the more efficient (GHS CRYPTO 2012).

Multiplicatively Homomorphic Version 3

Basic idea is to set $Q = q \cdot P$ for a large integer P.

Define the encryption of s^2 as

$$(a', b') = (a', a' \cdot s + p \cdot e' + P \cdot s^2)$$

Which makes even less sense!

Now define

$$\begin{aligned} e_0 &= P \cdot d_0 + b' \cdot d_2 \\ e_1 &= P \cdot d_1 + a' \cdot d_2. \end{aligned}$$

Multiplicatively Homomorphic Version 3

Then we have

$$\begin{aligned} e_0 - s \cdot e_1 &= P \cdot d_0 + \left(a' \cdot s + p \cdot e' + P \cdot s^2\right) \cdot d_2 \\ &\qquad - P \cdot d_1 \cdot s - a' \cdot d_2 \cdot s \\ &= P \cdot \left(d_0 - d_1 \cdot s + d^2 \cdot s^2\right) + p \cdot e' \cdot d_2 \\ &= P \cdot \left(m_0 \cdot m_1 + p \cdot \text{“small”}^2\right) + p \cdot e' \cdot d_2 \end{aligned}$$

Then "scale" down by P resulting in error term of roughly

$$\left(p \cdot \text{“small”}^2\right) + \frac{p \cdot e' \cdot d_2}{P}.$$

Cool or What?

So we can perform arithmetic on ciphertexts

- Which are elements of R_q^2

This maps to arithmetic on messages

- Which are elements of R_p

Pick an $F(X)$ and a p such that $F(X)$ factors completely mod p

$$F(X) = (X - \alpha_1) \cdots (X - \alpha_n) \pmod{p}.$$

Then by the Chinese Remainder Theorem we have

$$R_p = \mathbb{F}_p \times \ldots \times \mathbb{F}_p$$

So arithemtic in R_p becomes parallel (a.k.a. SIMD) arithmetic in $\mathbb{F}_p^n$.

Even Cooler

Suppose $K = \mathbb{Q}[X]/F(X)$ is a Galois extension.

We can also define homomorphic Galois actions

Let $\sigma \in \mathcal{G}al(K/\mathbb{Q})$ then we can homomorphically apply σ to the plaintext.

The Galois group allows us to move around data between the SIMD slots in $\mathbb{F}_p^n$, since the Galois group acts modulo p as well.

There are all sorts of tricks like this one can apply

Any Questions ?