Lecture 7 - Lattice Based Cryptography

Gidon Rosalki
2026-06-17

Notice: If you find any mistakes, please open an issue in the github repository

1. Introduction

1.1. ...Why?

Lattices are a new mathematical basis for cryptography. It's easy to ask why we need this, we already have RSA, DLOG, DDH and so on, and last lecture we also learnt about bilinear groups. One reason is that non of the previously learnt methods are not secure when we have quantum computers. For example, Shor showed in 1996 that quantum computers can (relatively easily) break RSA, DLOG, and so on. Currently, lattices are plausibly resistant to quantum computers (just as DDH is plausibly resistant against traditional computers). It is also interesting, and brings us new capabilities, much like how bilinear groups brought new capabilities impossible just over RSA/DLOG.

1.2. What

Let us suppose that we have the following 3 inequalities:

$$\begin{aligned} 3x_1 + 4x_2 + x_3 &= 0 \\ 4x_1 + 2x_2 + 6x_3 &= 1 \\ x_1 + x_2 + x_3 &= 1 \end{aligned}$$

All over $\bmod 7$. If we want to find x_1, x_2, x_3, it is quite trivial, and we find them to be $1, -1, 1$. This is called Gaussian elimination (as taught in linear algebra 1). We can also construct this with simplifying matrices and vectors, $Ax = b$.
This initial idea is trivial, but if we add in some noise e, such that we are now trying to find x where $Ax + e = b$, then Gaussian elimination no longer works.

Let us now consider that we have $Ax \in \mathbb{Z}_q$, and some "little noise" e. If e is completely random, then it completely hides what x might be, and we do not have enough information to be able to establish the value of x. However, if we say that e is little, limited in some sort of range, then perhaps we can do more.

We can also consider when $A, x \in \mathbb{F}_2$, so they are binary. Then, we may define e where each part has a value of 1 with probability p. So, for each inner product $\langle a_i, x_i \rangle + e$, where for each value, we flip it with probability p. This is also impossible in Gaussian elimination.

We are going to focus on the $\mathbb{Z}_q$ option.

2. Lattices

We are used to $\mathbb{Z}_q = \{0, ..., q-1\}$. However, to make our life simpler here, we will instead define

$$\mathbb{Z}_q = \left\{-\frac{q}{2}, ..., \frac{q}{2}\right\}$$

Additionally, we will define

$$e = (e_1, ..., d_n) \in \mathbb{Z}_q$$
$$\|e\|_\infty = \max e_i$$

We will also define B-bounded, which means

$$\Pr[\|e\|_\infty \leq B] = 1$$

We will assume:

$$LWE(n, m, q, X_B)$$

Where X_B is a B bounded distribution. We will also sample $A \leftarrow \mathbb{Z}_q^{m \times n}, s \leftarrow \mathbb{Z}_q^n, e \leftarrow X_B$. We will define search LWE, where given $A, As + e)$, find s' such that

$$\|As' - (As - e)\|_\infty \leq B$$

The essential idea is given a set of inequalities, with some B bounded noise, to find a value $s' = s$, within the limit of the B bounding.

We are considering all possible s in some field $\mathbb{Z}_q^n$, and a mapping A to another field $\mathbb{Z}_q^m$. This little noise means instead of mapping to a single other value, it maps to some small *group* of values in the field $\mathbb{Z}_q^m$. Given a point in one of these circles in $\mathbb{Z}_q^m$ (which may be overlapped with other circles, and all other problems), we want to find the original s that leads to it.

Theorem: For every PPT algorithm $\mathcal{A}$,

$$\Pr[\mathcal{A}(A, As + e) = s'] \leq \text{negl}(n)$$

We generally prefer to define with indistinguishability, so instead: *Theorem*: LWE for all PPT adversary, it cannot distinguish between

$$(A, As + e) \approx (A, u) : u \in \mathbb{Z}_q^m$$

This theorem is not particularly young, but converting it to make public key cryptography was a relatively recent development, from a professor in Tel Aviv university.

There is a theorem that LWE is equivalent to search LWE. We will probably have to prove it in our homework.

3. Regev's LWE PK Encryption Scheme

Let

$$\text{KeyGen}(1^n) \to A \leftarrow \mathbb{Z}_q^{m \times n}$$
$$s \leftarrow \mathbb{Z}_q^n$$
$$c \leftarrow X_B^m$$
$$b = As + e$$
$$sk = s$$
$$pk = (A, b) \in \mathbb{Z}_q^{m \times (n+1)}$$

So, $\text{Enc}(pk, x \in \{0, 1\})$:

$$r \leftarrow \{0, 1\}^m$$
$$c_0 = r^T A$$

$$c_1 = r^T \cdot b + \left\lfloor \frac{q}{2} \right\rfloor \cdot x$$
$$ct = (c_0, c_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

So this r means we are taking a random subset of rows from A, a random subset of answers from b, and adding on to the second either 0, or a very large number.
Our objective in decoding is to approximately remove the $r^T b$ value, since if $x = 0$, then we get some small value, and if $x = 1$ we will get a very large value.

So, $\text{Dec}(sk, (c_0, c_1))$:

$$\tilde{x} = c_1 - c_0 \cdot s$$
$$\text{output} = \begin{cases} 0 \text{ if } \tilde{x} < \frac{q}{4} \\ 1 \text{ else} \end{cases}$$

Correctness:

$$\begin{aligned} c_1 = c_0 \cdot s &= r^T \cdot b + \left\lfloor \frac{q}{2} \right\rfloor \cdot x - r^T A \cdot s \\ &= r^T(As + e) + \left\lfloor \frac{q}{2} \right\rfloor x - r^T As \\ &= r^T As + r^T e + \left\lfloor \frac{q}{2} \right\rfloor x - r^T As \\ &= r^T e + \left\lfloor \frac{q}{2} \right\rfloor x \end{aligned}$$

and $r^T \cdot e$ is limited by the size of $mB < \frac{q}{4}$, so correctness holds with very high probability.

Security: We want to prove equivalence between $pk = (A, b = As + e)$, and $pk = (A, b = u)$, for a random uniform u. This is tricky, and we will need to use the Leftover Hash Lemma:
Lemma: Leftover Hash Lemma: Let

$$\begin{aligned} &m \geq 2n \log q \\ &H \\ &A \leftarrow \mathbb{Z}_q^{m \times n} \\ &x \leftarrow \{0,1\}^m \\ &y \leftarrow \mathbb{Z}_q^n \end{aligned}$$

So,

$$(A, x^T A) \underset{\text{statistically}}{\approx} (A, y)$$

So, we take a very large matrix, and sum together sum subset of its rows. We are asking if some random vector is in fact random, or came from the sum of random subset of rows.

If we now construct the three hybrids:

$$pk = (A, b = As + e), c_0 = r^T A, c_1 = r^T b + \left\lfloor \frac{q}{2} \right\rfloor \cdot x$$
$$pk = (A, b = u), c_0 = r^T A, c_1 = r^T u + \left\lfloor \frac{q}{2} \right\rfloor \cdot x$$
$$pk = (A, b = u), c_0 = u', c_1 = r^T u + \left\lfloor \frac{q}{2} \right\rfloor \cdot x$$

And we may see that it is statistically indistinguishable from the Leftover Hash Lemma.

4. Lattice Mathematics

We said earlier that LWE problems are "lattice problems", but what even is a lattice? We will discuss this in relatively high level terms, and we may read more if we so desire.

A lattice is a set of points in $\mathbb{Z}_n$, that are linear combinations of basis vectors $B = \left\{\vec{b}_1, ..., \vec{b}_n\right\}$. So we have n basis vectors, which we recombine into a lattice:

$$L = \left\{ \sum_{i=1}^{n} a_i \vec{b}_i : a_i \in \mathbb{Z} \right\}$$

We can now do all sorts of problems over this lattice, such as the Closest Vector Problem (CVP), where given a point $t \in \mathbb{Z}^n$, and we are asked to bring a vector in the lattice $v \in L(B)$, such that $\|v - t\|$ is minimal. There is also SVP, shortest vector problem, where we find the "shortest" vector $u \in L(B)$.
So we have problems like this, which appear to be hard, and we assume them to be hard in the worst case, much like how for satisfiability, where we assume that there is no algorithm for everything, only specific solutions. Here we assume that there is no algorithm for any basis, just specific bases.

We can also convert these to approximate algorithms (approximation of γ), where instead of finding the minimum, we find one of the solutions that is approximately minimal. For large γ this is possible, but for small constant γ, it remains NP-hard. What is interesting, is if we believe that the problem is also hard for $\gamma = \text{poly}(n)$ in the worst case, then so too is LWE (which is average case, not worst case). This is wonderful, because it means we can make the assumption for Regev under a worst case assumption, rather than average case.

5. FHE (Fully Homomorphic Encryption)

(Allow us to compute over encrypted data, without decrypting it)

Created in 2009 by Gentry in Stanford, first time anyone created any system capable of this. In 2011, B&V showed that Gentry's system may be done with LWE. In 2013, G(entry from earlier)SW created a new method with LWE, which is much simpler than that of BV.

Let us begin with what we have:

$$\begin{aligned} \text{KeyGen}(1^n) &\to sk, pk \\ \text{Enc}(pk, m) &\to ct \\ \text{Dec}(sk, ct) &\to m \\ \text{Eval}(f, ct_1, ..., ct - d) &\to ct' \end{aligned}$$

Correctness: $\text{Dec}(sk, \text{Eval}(f, ct_1, ..., ct_l)) = f(m_1, ..., m_l)$

Security: We will want to show that $\forall m_0, m_1, \text{Enc}(pk, m_0) \approx \text{Enc}(pk, m_1)$. From these two, we could simply have eval return its input, and decryption will decrypt, and then apply f. To avoid this, we will also define conciseness:

Conciseness: $|ct' \leq \text{poly}(n)|$

So, how do we construct this properly?

- Phase 1: f is bounded depth, and $\exists p(\cdot)$ such that the depth of f is $\leq \text{poly}(n)$. This is leveled homomorphic encryption
- Phase 2: move from leveled, to fully arbitrary f

5.1. Idea

We will build a scheme where the secret key is a vector $\vec{s}$, and encryptions are matrices C such that $Cs = \mu s$, where μ is a constant, so s is an eigenvector of C, with respect to the eigenvalue μ.
Decryption is simply finding the value of μ, since it is our message here.

This is useful, since if for message μ_1, we have the ciphertext C_1, and for μ_2 we have C_2, then we may compute the sum of the messages without decryption since

$$(C_1 + C_2)s = (\mu_1 + \mu_2)s$$

What about multiplication?

$$\begin{aligned}(C_1 \cdot C_2)s &= C_1 \cdot \mu_2 s \\ &= \mu_1 \cdot \mu_2 s\end{aligned}$$

5.1.1. Attempt 1 (Secret Key)

The definition of s below is just to make the writing easier.

$$\begin{aligned}\text{KeyGen} : \vec{s} &= \mathbb{Z}_q^{n-1} \\ \vec{s} &= \begin{pmatrix}\tilde{s} \\ -1\end{pmatrix} \in \mathbb{Z}_q^n \\ \text{Enc}(s, \mu) : A &\leftarrow \mathbb{Z}_q^{n\times(n-1)} \\ e &\leftarrow X_B^n \\ c &= \left(A, A\vec{\tilde{s}} + \vec{e}\right) + \mu I_n \in \mathbb{Z}_q^{n\times n} \\ \text{Dec}(\vec{s}, c) &= \text{Compute } C \cdot \vec{s} \\ &\text{output } \begin{cases}0 \text{ if } \|C \cdot s\| \text{ is small} \\ 1 \text{ else}\end{cases}\end{aligned}$$

Let us consider encryption:

$$\begin{aligned}C \cdot \vec{s} &= \left(A, A\vec{\tilde{s}} + e\right)\begin{pmatrix}\vec{\tilde{s}} \\ -1\end{pmatrix} + \mu I_n \cdot \vec{s} \\ &= A\vec{\tilde{s}} - \left(A\vec{\tilde{s}} + e\right) + \mu\vec{s} \\ &= \mu\vec{s} - e\end{aligned}$$

Let us now check this for evaluation:

$$\begin{aligned}(c_1 + c_2)s &= (\mu_1 s + e_1) + (\mu_2 s + e_2) \\ &= (\mu_1 + \mu_2)s + (e_1 + e_1) \\ (c_1 \cdot c_2) &= c_1(\mu_2 s + e_2) \\ &= \mu_2(\mu_1 s + e_1) + c_1 \cdot e_2 \\ &= (\mu_1 \cdot \mu_2 s) + (\mu_2 \cdot e_1 + c_1 \cdot e_2)\end{aligned}$$

Which is not good, since c_2 is a large value, times a small value, so it is not ignorable noise. We want to ensure that c_1 is small, and we will do that next week.

5.1.2. Attempt 2

We will create a variation such that the matrices are not only matrices with eigenvector s, but the values inside these matrices are incredibly small.
We may reduce the size of the used number by encoding all the numbers in binary. This way, all our numbers are small (0 or 1):

$$\hat{x} = \left(x_0, ..., x_{\log(q-1)}\right)$$

So, we may (with a linear function) compute $x = \sum_i x_i 2^i$. This may be extended to vectors by encoding each number in a flat vector, since we know how many values correspond to each x. To convert this vector, we may multiply by the matrix G, since converting each value of the vector is a linear operation. This G will have a column corresponding to each value in the vector, and at the correct height of this column will be the values $1, 2, 4, ..., \log(q-1)$.
It is easy to convince ourselves from here that there exists an inverse matrix G^{-1}.
We may similarly extend this from an input vector, to an input matrix.

Let us define:

$$\begin{aligned}
\text{KeyGen} : s &= \begin{pmatrix} \tilde{s} \\ -1 \end{pmatrix} \in \mathbb{Z}_q^n \\
\text{Enc}(s, \mu) : A &\leftarrow \mathbb{Z}^{m \times (n-1)} : m = n \log q \\
e &\sim X_B^n \\
c &= [(A \mid A\tilde{s} + e) + \mu I_n] \cdot G : G \in \mathbb{F}_{n \times n \log q} \\
\text{Dec}(s, c) &= C \cdot G^{-1} \cdot s \\
&= ((A \mid As + e) + \mu I) \cdot s
\end{aligned}$$

Correctness:

$$\begin{aligned}
C \cdot s &= (A \mid A\tilde{s} + e) \begin{pmatrix} \tilde{s} \\ -1 \end{pmatrix} + \mu G s \\
&= e + \mu G s
\end{aligned}$$

Additional homomorphism:

$$\begin{aligned}
(C_1 + C_2) \cdot s &= (A \mid A\tilde{s} + e_1)s + (A \mid A\tilde{s} + e_2)s + \mu_1 G s + \mu_2 G s \\
&= ((A \mid A\tilde{s} + e) + (\mu_1 + \mu_2)G)s
\end{aligned}$$

Multiplication homomorphism:

$$\begin{aligned}
C_1 \cdot C_2 s &= C_1 \cdot (e + \mu_2 G s) \\
&= C_1 e + \mu_2 C_1 G s
\end{aligned}$$

This did not really help, since we are still multiplying the big C_1 by the error, and we are multiplying C_1 by G. Let us define $h(C)G = C$, so h is a function that does the inverse operation of G, and converts a matrix to binary.

$$\begin{aligned}
h(C_1) \cdot C_2 s &= h(C_1) \cdot (e + \mu_2 G s) \\
&= h(C_1)e + \mu_2 h(C_1) G s
\end{aligned}$$

Since $h(C_1)$ is only zeroes and ones, $h(C_1)e$ remains small. Therefore:

$$\begin{aligned}
h(C_1) \cdot C_2 s &= h(C_1) \cdot (e + \mu_2 G s) \\
&= h(C_1)e + \mu_2 h(C_1) G s \\
&= h(C_1)e + \mu_2 C_1 s \\
&= h(C_1)e + \mu_2 (e + \mu_1 G s) \\
&= h(C_1)e + \mu_1 \mu_2 G s + \mu_2 e
\end{aligned}$$

The problem here is that the size of ε increases with the number of homomorphic encryptions. If we perform

$$\text{Enc}(1) \times ... \times \text{Enc}(1)$$

Then after the first encryption we have noise of ε, the second we have the same result, but noise $O(\varepsilon^2)$, and so on until we wind up with $O(\varepsilon^t)$, which is really quite large, and we have ruined the homomorphism.

What we may do now, is take an encrypted form of the secret key, and use it to homomorphically decrypt the message. As a result, we have created a new encryption with error ε of the final value, and from there we may continue.
This comes with the caveat that it is not exactly accurate that the encryption of the secret key does not bring us any information as an adversary. It is seemingly impossible to prove that security schemes are secure against knowledge of an encrypted form of the secret key, so we make this an assumption. This assumption is called *circular security.*