

Lecture 3: Lattice Cryptography (Part 1)

Instructor: Fermi Ma

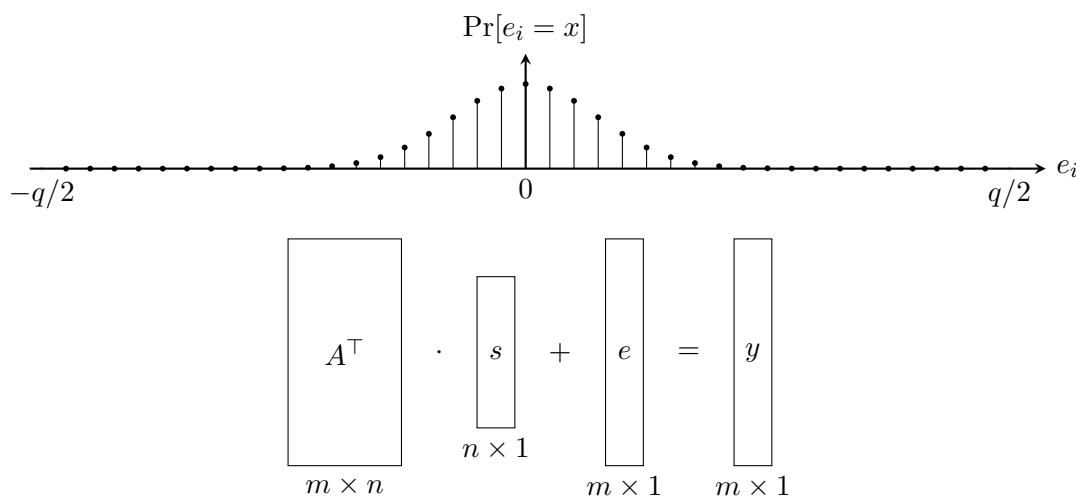
September 28, 2026

In the last two lectures, we covered Shor's algorithm, which breaks RSA and many other public-key cryptosystems. Progress toward building large-scale quantum computers has thus made post-quantum cryptography increasingly important. In response, the National Institute for Standards and Technology (NIST) finalized new **post-quantum cryptography** standards in 2024, recommending new cryptosystems designed to be secure against quantum attacks. NIST's primary recommendations are lattice-based cryptosystems built from (variants of) the **Learning with Errors (LWE)** problem.

Today, we'll cover the LWE problem, Regev's public-key encryption scheme built from LWE, and the basics of lattices. In the next lecture, we'll cover Regev's reduction, which shows that any quantum poly-time algorithm for LWE can be turned into a polynomial-time quantum algorithm for certain worst-case lattice problems.

1 The Learning with Errors Problem

The LWE problem asks you to solve a system of noisy random linear equations over $\mathbb{Z}_q$. Formally, let $A \leftarrow \mathbb{Z}_q^{n \times m}$ be a uniformly random matrix where $m \gg n$ (and $m = \text{poly}(n)$), let $s \leftarrow \mathbb{Z}_q^n$ be a uniformly random n -dimensional column vector, and let e be a random m -dimensional vector where each entry e_i is sampled according to a bounded error distribution, e.g., uniformly random in $[-B, B]$ for some $B \ll q$, or more commonly, a **discrete Gaussian** distribution. The discrete Gaussian is a Gaussian-shaped probability distribution supported on the integers mod q :



(Thanks to ChatGPT for the above diagrams)

To ensure the problem is information-theoretically solvable, the width of the discrete Gaussian is set to be much smaller than $q/2$. (Technically, the discrete Gaussian is not bounded, but you can truncate it with negligible change to the distribution.)

The LWE problem comes in both search and decision versions:

- In the **search** version of the LWE problem, the input is $(A, y = A^\top \cdot s + e)$, and the goal is to find s . The search LWE assumption is that no (quantum) polynomial-time algorithm can output s with non-negligible probability.
- In the **decision** version of the LWE problem, you are given *either* $(A, y = A^\top \cdot s + e)$, or (A, u) , where u is a uniformly random vector in $\mathbb{Z}_q^m$, and the task is to determine which case you are in. The decision LWE assumption is that no (quantum) polynomial-time algorithm can guess which case it is in with probability better than $1/2 + \text{negl}(n)$.

Roughly speaking, the search and decision versions of LWE are equally hard. One direction is obvious: if you can solve the search LWE, you can easily solve decision LWE. The other direction is less obvious, and we will not cover it today. If you're interested, I recommend reading <https://eprint.iacr.org/2011/521.pdf> to learn how the classical search-to-decision reduction works.

1.1 Regev's Public-Key Encryption Scheme

Next, we'll show that assuming decision LWE is hard, we can build a public-key cryptosystem. The public-key will be (A, y) and the secret key will be s .

How do we encrypt? The key step in the encryption procedure is to take a *random subset sum* of the m LWE samples (once we understand how this works, it'll be easy to state the encryption algorithm).

In particular, let's think of (A, y) as a list of m LWE samples,

$$(a_1, y_1 = \langle a_1, s \rangle + e_1), \dots, (a_m, y_m = \langle a_m, s \rangle + e_m), \quad (1)$$

where $a_i \in \mathbb{Z}_q^n$ is the i -th column of A , and $y_i = \langle a_i, s \rangle + e_i$ (which is a scalar) is the i th coordinate of y . Then, by sampling random bits $r_1, \dots, r_m \leftarrow \{0, 1\}$, and setting $a' = \sum_{i \in [m]} r_i a_i$ and $y' = \sum_{i \in [m]} r_i y_i$, we get a new LWE sample (a', y') with larger error. This is a well-formed LWE sample because

$$y' = \sum_{i \in [m]} r_i y_i = \sum_{i \in [m]} r_i (\langle a_i, s \rangle + e_i) = \underbrace{\langle \sum_{i \in [m]} r_i a_i, s \rangle}_{=a'} + \underbrace{\sum_{i \in [m]} r_i e_i}_{=e'} = \langle a', s \rangle + e'. \quad (2)$$

That is, (a', y') is a new LWE sample (with respect to the original secret vector s), although the noise e' is now larger. If e was originally bounded between $[-B, B]$, e' is in $[-Bm, Bm]$. So, for Regev's encryption to work, we'll need to assume the hardness of LWE when $B \ll q/m$ (which is standard).

But now observe: if you know the secret key s , you can easily distinguish (a', y') from random! You simply compute $\langle a', s \rangle - y'$ and recover e' , which is small compared to q .

This suggests the following encryption algorithm. To encrypt a bit b , sample uniformly random bits $r_1, \dots, r_m \leftarrow \{0, 1\}^m$, and then compute a', y' as above. Let $z = y' + b\lfloor q/2 \rfloor$, and output (a', z) as your ciphertext. To decrypt, compute $\langle a', s \rangle - z$. If the resulting value is close to 0, then message was $b = 0$, and if it's close to $q/2$, the message was $b = 1$.

Why is this secure? We need to show that an adversary given the public key (A, y) cannot distinguish between

- an encryption of 0, which is distributed as $(A \cdot r, y^\top \cdot r)$, r is a uniformly random vector from $\{0, 1\}^m$, and
- an encryption of 1, which is distributed as $(A \cdot r, y^\top \cdot r + \lfloor q/2 \rfloor)$, r is a uniformly random vector from $\{0, 1\}^m$.

Equivalently, the task is to distinguish between

- Distribution 0: $(A, y, A \cdot r, y^\top \cdot r)$
- Distribution 1: $(A, y, A \cdot r, y^\top \cdot r + \lfloor q/2 \rfloor)$.

Let us define the following two distributions:

- Distribution 0': $(A, u, A \cdot r, u^\top \cdot r)$. This is identical to distribution 0, except that we've replaced the $y = A^\top \cdot s + e$ with a uniformly random vector u .
- Distribution 1': $(A, u, A \cdot r, u^\top \cdot r + \lfloor q/2 \rfloor)$. This is identical to distribution 1, except that we've replaced the $y = A^\top \cdot s + e$ with a uniformly random vector u .

By the Decision LWE assumption, Distribution 0 is indistinguishable from Distribution 0', and similarly Distribution 1 is indistinguishable from Distribution 1'. Indeed, given a candidate public key (either (A, y) or (A, u)) we can sample r and compute the ciphertext ourselves without knowing s , so any algorithm that distinguishes Distribution 0 from Distribution 0' (or Distribution 1 from Distribution 1') would also distinguish LWE samples from uniformly random samples.

Next, we claim that Distribution 0' and Distribution 1' are *information-theoretically* indistinguishable. To see this, think of $(A \cdot r, u^\top \cdot r)$ as a random subset sum of the m uniformly random vectors $(a_i, u_i) \in \mathbb{Z}_q^{n+1}$. One can show (by something called the Leftover Hash Lemma) that, provided m is sufficiently larger than $(n+1) \log_2 q$, this subset sum is statistically close to a fresh uniformly random vector, *even given* (A, u) . That is,

$$(A, u, A \cdot r, u^\top \cdot r) \approx \text{stat}(A, u, w), \quad (3)$$

where $w \leftarrow \mathbb{Z}_q^{n+1}$ is uniformly random and independent of (A, u) . Finally, since adding $\lfloor q/2 \rfloor$ to the last coordinate of a uniformly random vector leaves its distribution unchanged, it follows that Distribution 0' and Distribution 1' are both statistically close to the same distribution (A, u, w) , and hence statistically close to each other.

2 Lattices

As we mentioned at the beginning of class, Regev showed that solving LWE is as hard as solving certain lattice problems in the worst case. Towards understanding that reduction (which we will cover next class), we'll first need some background on lattices.

A lattice L is a discrete additive subgroup of $\mathbb{R}^n$. Recall that to be an additive subgroup, L must satisfy:

- $0 \in L$ (where 0 means the n dimensional all 0's vector),
- if $x, y \in L$, then $x + y \in L$, and
- if $x \in L$, then $-x \in L$.

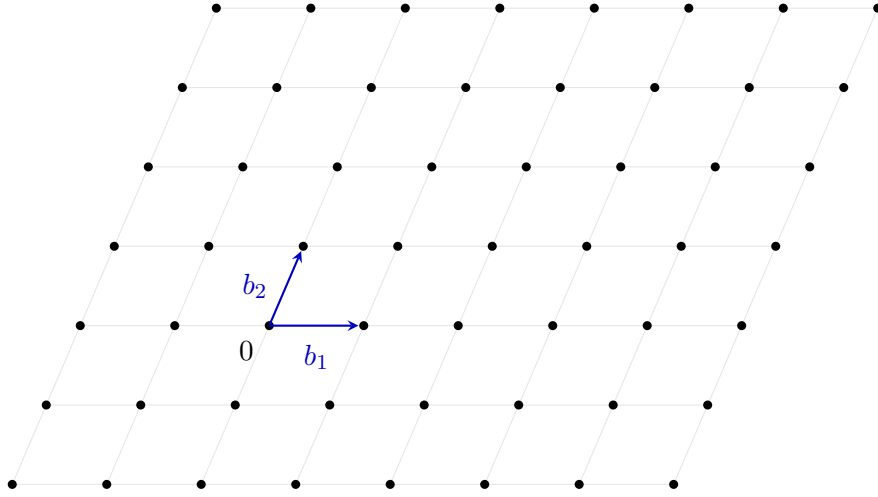
Discrete means that there exists $\varepsilon > 0$ such that for all non-zero $x \in L$, $\|x\|_2 > \varepsilon$, i.e., x is bounded away from 0.

We will restrict attention to full-rank lattices, i.e., lattices where any point in $\mathbb{R}^n$ can be constructed by taking a linear combination of lattice points (with real-valued coefficients).

With this restriction, we can define a lattice L to be all **integer** combinations of a set of linearly independent vectors $(\vec{b}_1, \dots, \vec{b}_n)$. We'll use the notation $B = (\vec{b}_1, \dots, \vec{b}_n)$ to denote the matrix whose columns are the basis vectors for the lattice, and

$$L(b) := B \cdot \mathbb{Z}^n = \left\{ \sum_{i \in [n]} x_i \cdot \vec{b}_i : x_i \in \mathbb{Z} \right\}. \quad (4)$$

Thanks to ChatGPT, here is a picture of a lattice where $n = 2$:



To make sure the definition of a lattice is clear, let's go through some examples and non-examples:

- Is $\mathbb{Z}^2$ a lattice?

Yes. This is a lattice with basis $b_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $b_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

- Is $\{a\sqrt{2} : a \in \mathbb{Z}\}$ a lattice?

Yes. This is a 1 dimensional-lattice spanned by the 1-D basis vector $b_1 = \sqrt{2}$.

- Is $\{a + b\sqrt{2} : a, b \in \mathbb{Z}\}$ a lattice?

No. This is not a lattice. Since $\sqrt{2}$ is irrational, there exist arbitrarily good rational approximations $p/q \approx \sqrt{2}$. As a result, you can find integers p and q to make $q\sqrt{2} - p$ arbitrarily small, which violates the requirement that lattices must be discrete.

- Is $\mathbb{Z}^2 + \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$ a lattice?

No. This is not a lattice since it doesn't contain 0.

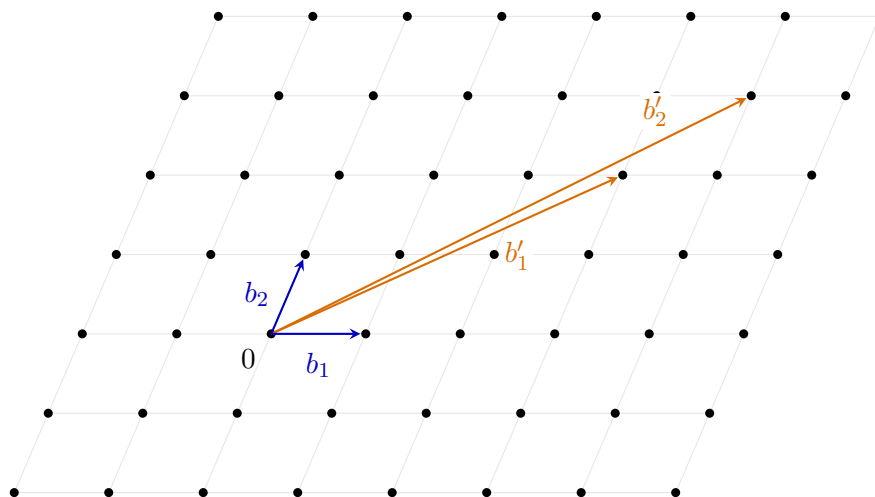
- Is $\{x \in \mathbb{Z}^m : Ax = 0 \bmod q\}$ a lattice?

Yes. This is discrete (since it's contained in $\mathbb{Z}^m$) and you can verify that the set of $x \in \mathbb{Z}^m$ s.t. $Ax = 0 \bmod q$ satisfy all the subgroup properties.

2.1 Lattice problems

Many important lattice problems ask us to find *short vectors* in a lattice. This might seem easy in the two-dimensional picture above, but the difficulty is believed to grow roughly exponentially with the dimension. One complication is that the same lattice has many different bases, and given a highly skewed basis, it may not clear whether the lattice contains much shorter vectors.

Here is another ChatGPT-generated picture of a skewed basis for the lattice above (but note that since $n = 2$, finding short vectors is still easy):



Here are two of the most well-known lattice problems:

- The Shortest Vector Problem (SVP): Given a basis B , find the shortest non-zero vector in $L(B)$.
- The Shortest Independent Vectors Problem (SIVP): Given a basis B , find n linearly independent vectors $v_1, \dots, v_n \in L(B)$ that minimize $\max_{i \in [n]} \|v_i\|_2$.

For any approximation ratio $\alpha > 1$, we can also consider approximation versions of these problems, where the goal is to output a solution whose length is at most α times the optimum. The corresponding *gap decision versions* ask us to distinguish whether the optimal length is at most d or greater than αd , given a threshold $d > 0$. The algorithm may answer arbitrarily when the optimal length falls in the gap between these thresholds.

There are roughly three regimes of difficulty worth highlighting:

- An $O(1)$ approximation ratio. In this regime, the problems are NP-hard, but their (worst-case) hardness is not known to be sufficient for cryptographic applications.
- A $\text{poly}(n)$ approximation ratio. Approximate SVP and SIVP are still believed to be hard, and this is where hardness has cryptographic implications, as we'll see next class. For sufficiently large polynomial factors, the gap decision versions are in $\text{NP} \cap \text{coNP}$ and are therefore unlikely to be NP-hard.
- A $2^{O(n \log \log n / \log n)}$ approximation ratio. For this approximation ratio, we have polynomial-time algorithms.