# Lecture 12: Lattice based crypto

Boaz Barak

## Lattice based encryption - some history and significance.

Lattice based public key encryption (and its cousins known as knapsack and coding based encryption) have almost as long a history as discrete logarithm and factoring based schemes. Already in 1976, right after the Diffie-Hellman key exchange was discovered (and before RSA), Ralph Merkle was working on building public key encryption from the NP hard *knapsack* problem (see Diffie's recollection). This can be thought of as the task of solving a linear equation of the form $Ax = y$ (where $A$ is a given matrix, $y$ is a given vector, and the unknown are $x$) over the real numbers but with the additional constraint that $x$ must be either 0 or 1. His proposal evolved into the Merkle-Hellman system proposed in 1978.

McEliece proposed in 1978 a system based on the difficulty of the decoding problem for general linear codes. This is the task of solving *noisy linear equations* where one is given $A$ and $y$ such that $y = Ax + e$ for a "small" error vector $e$, and needs to recover $x$. Crucially, here we work in a finite field, such as working modulo $q$ for some prime $q$ (that can even be 2) rather than over the reals or rationals. There are special matrices $A^*$ for which we know how to solve this problem efficiently: these are known as efficiently decodable error correcting codes. McEliece suggested a scheme where the key generator lets $A$ be a "scrambled" version of a special $A^*$ (based on the Goppa algebraic geometric code). So, someone that knows the scrambling could solve the problem, but (hopefully) someone that doesn't know it wouldn't. McEliece's system has so far not been broken.

In a 1996 breakthrough, Ajtai showed a *private key* scheme based on integer lattices that had a very curious property- its security could be based on the assumption that certain problems were only hard in the *worst case*, and moreover variants of these problems were known to be NP hard. This ignited hope that we could perhaps realize the old dream of basing crypto on the mere assumption that $P \neq NP$. Alas, we know understand that there are fundamental barriers to this approach. Nevertheless, Ajtai's work attracted signifincant interest, and within a year both Ajtai and Dwork, as well as Goldreich, Goldwasser and Halevi

came up with lattice based constructions for *public key* encryption (the former based also on *worst case* assumptions). At about the same time, Hoffstein, Pipher, and Silverman came up with their NTRU public key system which is based on stronger assumptions but offers better performance, and they started a company around it together with Daniel Lieman.

You may note that I haven't yet said what *lattices* are; we will do so later, but for now if you simply think of questions involving linear equations modulo some prime $q$, you will get enough of the intuition that you need. (The lattice viewpoint is more geometric, and we'll discuss it more below; it was first used to *attack* cryptosystems and in particular break the Merkle-Hellman knapsack scheme and many of its variants.)

Lattice based cryptography has captured a lot of attention recently from both theory and practice. In the theory side, many cool new constructions are now based on lattice based cryptography, and chief among them fully homomorphic encryption, as well as indistinguishability obfuscation (though the latter's security's foundations is still far less solid). On the applied side, the steady advances in the technology of quantum computers have finally gotten practitioners worried about RSA, Diffie Hellman and Elliptic Curves. While current constructions for quantum computers are nowhere near being able to, say, factor larger numbers that can be done classically (or even than can be done by hand), given that it takes many years to develop new standards and get them deployed, many believe the effort to transition away from these factoring/dlog based schemes should start today. The NSA has recently suggested that it plans to initiate the process to "transition to quantum resistant algorithms in the not too distant future"; see also this very interesting FAQ on this topic. Note that cryptography has the peculiar/unfortunate feature that if a machine is built that can factor large integers in 20 years, it can still be used to break the communication we transmit *today*, provided this communication was recorded. So, if you have some data that you expect you'd want still kept secret in 20 years (as many government and commercial entities do), you might have reasons to worry. Currently lattice based cryptography is the only real "game in town" for potentially quantum-resistant public key encryption schemes.

## A world without Gaussian elimination

The general approach people used to get a public key encryption is to obtain a hard computational problem with some mathematical *structure*. We've seen this in the *discrete logarithm* problem, where the task is to invert the map $a \mapsto g^a \pmod{p}$, and the integer factoring problem, where the task is to invert the map $a, b \mapsto a \cdot b$. Perhaps the simplest structure to consider is the task of solving linear equations. Pretend that we didn't know of Gaussian elimination, and that if we picked a "generic" matrix $A$ then the map $x \mapsto Ax$ would be hard to invert. (Here and elsewhere, our default interpretation of a vector $x$ is as a

*column* vector, and hence if $x$ is $n$ dimensional and $A$ is $m \times n$ then $Ax$ is $m$ dimensional. We use $x^\top$ to denote the row vector obtained by *transposing* $x$.) Could we use that to get a public key encryption scheme?

Here is a concrete approach. Let us fix some prime $q$ (think of it as polynomial size, e.g., $q$ is smaller than 1024 or so, though people can and sometimes do consider $q$ of exponential size), and all computation below will be done modulo $q$. The secret key is a vector $x \in \mathbb{Z}_q^n$, and the public key is $(A, y)$ where $A$ is a random $m \times n$ matrix with entries in $\mathbb{Z}_q$ and $y = Ax$. Under our assumption, it is hard to recover the secret key from the public key, but how do we use the public key to encrypt? The crucial observation is that even if we don't know how to solve linear equations, we can still combine several equations to get new ones. To keep things simple, let's consider the case of encrypting a single bit. We think of the public key as the set of equations $\langle a_1, x\rangle = y_1, \ldots, \langle a_m, x\rangle = y_m$ in the unknown variables $x$. The idea is that to encrypt the value 0 we will generate a new *correct* equation on $x$, while to encrypt the value 1 we will generate an *incorrect* equation. To decrypt a ciphertext $(a, \sigma) \in \mathbb{Z}_q^{n+1}$, we think of it as an equation of the form $\langle a, x\rangle = \sigma$ and output 1 if and only if the equation is correct.

How does the encrypting algorithm, that does not know $x$, get a correct or incorrect equation on demand? One way would be to simply take two equations $\langle a_i, x\rangle = y_i$ and $\langle a_j, x\rangle = y_j$ and add them together to get the equation $\langle a_i + a_j, x\rangle = y_i + y_j$. This equation is correct and so one can use it to encrypt 0, while to encrypt 1 we simply add some fixed nonzero number $\alpha \in \mathbb{Z}_q$ to the right hand side to get the incorrect equation $\langle a_i + a_j, x\rangle = y_i + y_j + \alpha$. However, even if it's hard to solve for $x$ given the equations, an attacker (who also knows the public key $(A, y)$) can try itself all pairs of equations and do the same thing. Our solution for this is simple- just add more equations! If the encryptor adds a random subset of equations then there are $2^m$ possibilities for that, and an attacker can't guess them all. Thus, at least intuitively, the following encryption scheme would be "secure" in the Gaussian-elimination free world of attackers that haven't taken freshman linear algebra:

- *Key generation*: Pick random $m \times n$ matrix $A$ over $\mathbb{Z}_q$, and $x \leftarrow_R \mathbb{Z}_q^n$, the secret key is $x$ and the public key is $(A, y)$ where $y = Ax$.
- *Encryption*: To encrypt a message $b \in \{0, 1\}$, pick $w \in \{0, 1\}^m$ and output $w^\top A, \langle w, y\rangle + \alpha b$ for some fixed nonzero $\alpha \in \mathbb{Z}_q$.
- *Decryption:* To decrypt a ciphertext $(a, \sigma)$, output 0 iff $\langle a, x\rangle = \sigma$.

(Please make sure that you see why this description corresponds to the previous one; as usual all calculations are done modulo $q$.)

## Security in the real world.

Like it or not (and cryptographers typically don't) Gaussian elimination *is* possible in the real world and the scheme above is completely insecure. However, the Gaussian elimination algorithm is extremely *brittle*.
Errors tend to be amplified when you combine equations. To see why, let us recall how Gaussian elimination works. Think of $m = n$ for simplicity. Given equations $Ax = y$ in the unknown variables $x$, the goal of Gaussian elimination is to transform them into the equations $Ix = y'$ where $I$ is the identity matrix (and hence the solution is simply $x = y'$). Recall how we do it: by rearranging and scaling, we can assume that the top left corner of $A$ is equal to 1, and then we add the first equation to the other equations (scaled apropriately) to zero out the first entry in all the other rows of $A$ (i.e., make the first column of $A$ equal to $(1, 0, \ldots, 0)$) and continue onwards to the second column and so on and so forth. Now, suppose that the equations were *noisy*, in the sense that we added to $y$ a vector $e \in \mathbb{Z}_q^m$ such that $|e_i| < \delta q$ for every $i$. (Because over $\mathbb{Z}_q$, we can think of $q - 1$ also as the number $-1$, and so on, if $a \in \mathbb{Z}_q$, we define $|a|$ to be the minimum of $a$ and $q - a$. This ensures the absolute value satisfies the natural property of $|a| = |-a|$.) Even ignoring the effect that our scaling has on this, simply adding the first equation to the rest would typically tend to increase their relative error from $\approx \delta$ to $\approx 2\delta$. Now, when we repeat the process, we increase the error from $\approx 2\delta$ to $\approx 4\delta$, and we see that by the time we're done dealing with all $n$ variables, we have equations either error level roughly $2^n\delta$. So, unless $\delta$ was truly tiny (and $q$ truly big, in which case the difference between working in $\mathbb{Z}_q$ and simply working with integers or rationals disappears), the resulting equations have the form $Ix = y' + e'$ where $e'$ is so big that we get no information on $x$.

The *Learning With Errors (LWE)* conjecture is that this is *inherent*:

**Conjecture (Learning with Errors, Regev 2005):** Let $q = q(n)$ and $\delta = \delta(n)$ be some functions. The Learning with Error (LWE) conjecture with respect to $q, \delta$, is that for every polynomial-time adversary $E$ and $m = poly(n)$, the probability that $E(A, Ax + e) = x$ is negligible, where $A$ is a random $m \times n$ matrix in $\mathbb{Z}_q$, $x$ is random in $\mathbb{Z}_q^n$, and $e \in \mathbb{Z}_q^m$ is a random noise vector with magnitude $\delta q$.[1] The *LWE conjecture* is that for ever polynomial $p(n)$ there is some polynomial $q(n)$ such that LWE holds with respect to $q(n)$ and $\delta(n) = 1/p(n)$.[2]

[1] One can think of $e$ as chosen by simply letting every coordinate be chosen at random in $\{-\delta q, -\delta q + 1, \ldots, +\delta q\}$. For technical reasons, we sometimes consider other distributions and in particular the *discrete Gaussian* distribution which is obtained by letting every coordinate of $e$ be an independent Gaussian random variable with standard deviation $\delta q$, conditioned on it being an integer. (A closely related distribution is obtained by picking such a Gaussian random variable and then rounding it to the nearest integer.)

[2] People sometimes also consider variants where both $p(n)$ and $q(n)$ can be as large as exponential.

## Search to decision

It turns out that if the LWE is hard, then it is even hard to distinguish between random equations and nearly correct ones:

**Theorem (Search to decision reduction):** If the LWE conjecture is true then for every $q = poly(n)$ and $\delta = 1/poly(n)$ and $m = poly(n)$, the following two distributions are computationally indistinguishable:

- $\{(A, Ax + e)\}$ where $A$ is random $m \times n$ matrix in $\mathbb{Z}_q$, $x$ is random in $\mathbb{Z}_q^n$ and $e \in \mathbb{Z}_q^m$ is random noise vector of magnitude $\delta$.
- $\{(A, y)\}$ where $A$ is random $m \times n$ matrix in $\mathbb{Z}_q$ and $y$ is random in $\mathbb{Z}_q^m$

**Proof:** Suppose that we had a decisional adversary $D$ that succeeds in distinguishing the two distributions above with bias $\epsilon$. For example, suppose that $D$ outputs 1 with probability $p + \epsilon$ on inputs from the first distribution, and outputs 1 with probability $p$ on inputs from the second distribution. We will show how we can use this to obtain a polynomial-time algorithm $S$ that on input $m$ noisy equations on $x$ and a value $a \in Z_q$, will learn with high probability whether or not the first coordinate of $x$ equals $a$. Clearly, we can repeat this for all the possible $q$ values of $a$ to learn the first coordinate exactly, and then continue in this way to learn all coordinates.

Our algorithm $S$ gets as input the pair $(A, y)$ where $y = Ax + e$ and we need to decide whether $x_1 = a$. Now consider the instance $A + (r\|0^m\|\cdots\|0^m), y + ar$, where $r$ is a random vector in $\mathbb{Z}_q^m$ and the matrix $(r\|0^m\|\cdots\|0^m)$ is simply the matrix with first column equal to $r$ and all other columns equal to 0. Note that if $A$ is random then $A + r\|0^m\|\cdots\|0^m)$ is random as well. Now note that $Ax + (r|0^m \cdots \|0^m)x = Ax + x_1 r$ and hence if $x_1 = a$ then we still have an input of the same form. However, if $x_1 \neq a$ then this amounts to adding a non-zero multiple of the random vector $r$ to the noise vector, and hence we get an instance of the form $(A', y')$ with random $A', y'$.
Hence if we send this input to our the decision algorithm $D$, then we would get 1 with probability $p + \epsilon$ if $x_1 = a$ and an output of 1 with probability $p$ otherwise. Now the crucial observation is that if our decision algorithm $D$ requires $m$ equations to succeed with bias $\epsilon$, we can use $100mn/\epsilon^2$ equations (which is still polynomial) to invoke it $100n/\epsilon^2$ times. This allows us to distinguish with probability $1 - 2^{-n}$ between the case that $D$ outputs 1 with probability $p + \epsilon$ and the case that it outputs 1 with probability $p$ (this follows from the Chernoff bound we discussed in the mathematical backgroudn handout; can you see why?). Hence by using polynomially more samples than the decision algorithm $D$, we get a search algorithm $S$ that can actually recover $x$. QED

## An LWE based encryption scheme

We can now show the secure variant of our original encryption scheme:

- *Parameters:* Let $\delta(n) = 1/n^4$ and let $q = poly(n)$ be a prime such that LWE holds w.r.t. $q, \delta$. We let $m = n^2 \log q$.
- *Key generation:* Pick $x \in \mathbb{Z}_q^n$. The private key is $x$ and the public key is $(A, y)$ with $y = Ax + e$ with $e$ a $\delta$-noise vector and $A$ a random $m \times n$ matrix.
- *Encrypt:* To encrypt $b \in \{0,1\}$ given the key $(A, y)$, pick $w \in \{0,1\}^m$ and output $w^\top A, \langle w, y \rangle + b\lfloor q/2 \rfloor$.
- *Decrypt:* To decrypt $(a, \sigma)$, output 0 iff $|\langle a, x \rangle - \sigma| < q/10$.

Unlike our typical schemes, here it is not immediately clear that even get the right answer, but we do:

**Claim:** With high probability, the decryption of the encryption of $b$ equals $b$.

**Proof:** $\langle w^\top A, x \rangle = \langle w, Ax \rangle$. Hence, if $y = Ax + e$ then $\langle w, y \rangle = \langle w^\top A, x \rangle + \langle w, e \rangle$. But since every coordinate of $w$ is either 0 or 1, $|\langle w, e \rangle| < \delta m q \ll q < q/10$.[3] So, we get that if $a = w^\top A$ and $\sigma = \langle w, y \rangle + b\lfloor q/2 \rfloor$ then $\sigma - \langle a, x \rangle = \langle w, e \rangle + b\lfloor q/2 \rfloor$ which will be smaller than $q/10$ iff $b = 0$. QED

We now claim that this scheme is CPA secure. For a public key encryption scheme with messages that are just bits, this just means that an encryption of 0 is indistinguishable from an encryption of 1, even given the public key. This will follow from the following lemma:

**Lemma:** Assuming the LWE, the distribution $(A, y = Ax + e), (w^\top A, \langle w^\top, y \rangle)$ (where all values are chosen as above) is indistinguishable from the distribution $(A, y), (a, \sigma)$ where $y$ is completely random in $\mathbb{Z}_q^m$, $a$ is random and independent in $\mathbb{Z}_q^n$ and $\sigma$ is random and independent in $\mathbb{Z}_q$.

(I leave to the reader to verify that this lemma implies security; the idea is that it shows that the concatenation of the public key and encryption of 0 is indistinguishable from something that is completely random, and you can use it to show that the concatenation of the public key and encryption of 1 is indistinguishable from the same thing, and then finish using the hybrid argument. )

**Proof:** By the search to decision reduction, the distribution above is indistinguishable from the distribution where $y$ is completely random. However, in this case I claim that if we choose $w$ at random in $\{0,1\}^m$ and let $(a, \sigma) = w^\top (A \| y)$ then $(a, \sigma)$ would be a (close to) completely uniform and independent vector in $\mathbb{Z}_q^{n+1}$. We will not do the whole proof (which uses the mod $q$ version of the *leftover hash lemma* we mentioned before) but the idea is simple. Let $A' = (A \| y)$ which in our case is a completely random matrix. This mans that the map $w \mapsto w^\top A'$ is essentially a *pairwise independent* hash function mapping $\mathbb{Z}_q^m$

[3] In fact, due to the fact that the *signs* of the error vector's entries are different, we expect the errors to have significant cancellations and hence we would expect $|\langle w, e \rangle|$ to only be roughly of magnitude $\sqrt{m}\delta q$, but this is not crucial for our discussions.

to $\mathbb{Z}_q^{n+1}$. Now when we choose $w$ at random in $\{0,1\}^m$, it is coming from a distribution with $m$ bits of entropy. If $m \gg (n+1)\log q$, then because the output of this function is so much smaller than $m$, we expect it to be completely uniform. QED

This proof is quite subtle and requires some thought. To read more about this, you can look at the survey of Oded Regev, "On the Learning with Error Problem" Sections 3 and 4.

## But what are lattices?

You can think of a lattice as a discrete version of a subspace. A lattice $L$ is simply a subset of $\mathbb{R}^n$ such that if $u, v \in L$ and $a, b$ are integers then $au + bv \in L$. A lattice is given by a basis which simply a matrix $B$ such that every vector $u \in L$ is obtained as $u = Bx$ for some vector of integers $x$. It can be shown that we can assume without loss of generality that $B$ is full dimensional and hence it's an $n$ by $n$ invertible matrix. Note that given a basis $B$ we can generate vectors in $L$, as well as test whether a vector $v$ is in $L$ by testing if $B^{-1}v$ is an integer vector. There can be many different bases for the same lattice, and some of them are easier to work with than others.

Some classical computational questions on lattices are:

- *Shortest vector problem:* Given a basis $B$ for $L$, find the nonzero vector $v$ with smallest norm in $L$.
- *Closest vector problem:* Given a basis $B$ for $L$ and a vector $u$ that is *not* in $L$, find the closest vector to $u$ in $L$.
- *Bounded distance decoding:* Given a basis $B$ for $L$ and a vector $u$ of the form $u = v + e$ where $v$ is in $L$, and $e$ is a particularly short "error" vector (so in particular no other vector in the lattice is within distance $\|e\|$ to $u$), recover $v$. Note that this is a special case of the closest vector problem.

In particular, if $V$ is a linear subspace of $\mathbb{Z}_q^n$, we can think of it also as a lattice $\hat{V}$ of $\mathbb{R}^n$ where we simply say that that a vector $\hat{u}$ is in $\hat{V}$ if all of $\hat{u}$'s coordinates are integers and if we let $u_i = \hat{u}_i \pmod q$ then $u \in V$. The learning with error task of recovering $x$ from $Ax + e$ can then be thought of as an instance of the bounded distance decoding problem for $\hat{V}$.

## Ring based lattices

One of the biggest issues with lattice based cryptosystem is the key size. In particular, the scheme above uses an $m \times n$ matrix where each entry takes $\log q$ bits to describe. (It also encrypts a single bit using a whole vector, but more efficient "multi-bit" variants are known.) Schemes using *ideal lattices* are an attempt to get more practical variants. These have very similar structure except that the matrix $A$ chosen is not completely random but rather can be described

by a single vector. One common variant is the following: we fix some polynomial $p$ over $\mathbb{Z}_q$ with degree $n$ and then treat vectors in $\mathbb{Z}_q^n$ as the coefficients of $n-1$ degree polynomials and always work modulo this polynomial $p()$. (By this I mean that for every polynomial $t$ of degree at least $n$ we write $t$ as $ps + r$ where $p$ is the polynomial above, $s$ is some polynomial and $r$ is the "remainder" polynomial of degree $< n$; then $t \pmod p = r$.) Now for every fixed polynomial $t$, the operation $A_t$ which is defined as $s \mapsto ts \pmod p$ is a linear operation mapping polynomials of degree at most $n-1$ to polynomials of degree at most $n-1$, or put another way, it is a linear map over $\mathbb{Z}_q^n$. However, the map $A_d$ can be described using the $n$ cofficients of $t$ as opposed to the $n^2$ description of a matrix. It also turns out that by using the Fast Fourier Transform we can evaluate this operation in roughly $n$ steps as opposed to $n^2$. The ideal lattice based cryptosystem use matrices of this form to save on key size and computation time. It is still unclear if this structure can be used for attacks; recent papers attacking principal ideal lattices have shown that one needs to be careful about this.