CS 294. Worst-case to Average-case Reduction for LWE

In this lecture, we will show a worst-case to average-case reduction for LWE.

1 Decision to Search Reduction for LWE

The first step is to come up with a way to reduce the search version of LWE to the decision version (which is the basis of cryptographic schemes, e.g., the public-key encryption schemes we already saw in Lecture 1). Later, we will show a reduction from worst-case lattice problems to search LWE, completing the chain of reductions.

1.1 Worst-case vs. Average-case Secret

We start with the simple observation that solving LWE with a worst-case secret $\mathbf{s}$ is just as easy as solving it with a uniformly random secret $\mathbf{s}$. That is, it is easy to re-randomize the secret $\mathbf{s}$. The key observation is that $\mathbf{A}$ is public and that everything here is additive.

Indeed, given an LWE input $(\mathbf{A}, \mathbf{b}_{wc}^T := \mathbf{s}_{wc}^T\mathbf{A} + \mathbf{e}^T)$ with an arbitrary secret $\mathbf{s}_{wc}$, the re-randomization algorithm (the reduction) computes

$$\mathbf{b}^T := \mathbf{b}_{wc}^T + \mathbf{s}_r^T\mathbf{A}$$

for a uniformly random vector $\mathbf{s}_r \leftarrow \mathbb{Z}_q^n$. Now, note that

$$\mathbf{b}^T := (\mathbf{s}_{wc} + \mathbf{s}_r)^T\mathbf{A} + \mathbf{e}^T$$

which is an LWE input with the uniformly random secret $\mathbf{s} := \mathbf{s}_{wc} + \mathbf{s}_r$. Clearly, if there is an algorithm that finds $\mathbf{s}$ given $(\mathbf{A}, \mathbf{b})$, the reduction can recover $\mathbf{s}_{wc} := \mathbf{s} - \mathbf{s}_r$.

1.2 A Simple Reduction

We now show a reduction from search LWE to decisional LWE. Before we begin, a few words about average-case reductions. These are quite tricky to get right. A typical reduction solves a distinguishing problem, such as coming up with an algorithm (typically probabilistic polynomial-time) that distinguishes between two probability distributions $\mathcal{D}_0$ and $\mathcal{D}_1$. Such an algorithm is said to be a (T, ϵ)-distinguisher if it runs in time T and has a (distinguishing) advantage of ϵ:

$$|\Pr[x \leftarrow \mathcal{D}_0; \mathsf{Dist}(x) = 1] - \Pr[x \leftarrow \mathcal{D}_1; \mathsf{Dist}(x) = 1]| \leq \epsilon$$

Equivalently,

$$1/2 - \epsilon/2 \leq \Pr[b \leftarrow \{0,1\}; x \leftarrow \mathcal{D}_b; \mathsf{Dist}(x) = b] \leq 1/2 + \epsilon/2$$

Theorem 1. *If there is a (T, ϵ)-distinguisher for decisional $\mathsf{LWE}_{n,m,q,\chi}$, then there is a time $T' = \widetilde{O}(T \cdot nq/\epsilon^2)$-time algorithm that solves search $\mathsf{LWE}_{n,m',q,\chi}$ with probability $1 - o(1)$, where $m' = \widetilde{O}(nmq/\epsilon^2)$, where $\widetilde{O}(\cdot)$ hides polylogarithmic factors in n.*

Proof. Our approach to solve search $\mathsf{LWE}_{n,m',q,\chi}$ will be to "guess" the secret, one coordinate at a time. Let $s_1, \ldots, s_n \in \mathbb{Z}_q$ denote the coordinates of $\mathbf{s}$, that is, $\mathbf{s} = (s_1, \ldots, s_n)$. Consider the algorithm which, on input $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$, for each $i \in [m]$, guesses the i^{th} coordinate of $\mathbf{s}$

Algorithm 1 "Guess" the i^{th} coordinate of $\mathbf{s}$

```
For $j = 0, \ldots, q-1$:
  • Let $g_i := j$.
  • For $\ell = 1, \ldots, L = \widetilde{O}(1/\epsilon^2)$:
      – Choose a fresh block of the search LWE challenge, call it $(\mathbf{A}_\ell, \mathbf{b}_\ell)$.
      – Sample a random vector $\mathbf{c}_\ell \leftarrow \mathbb{Z}_q^m$, and let $\mathbf{C}_\ell \in \mathbb{Z}_q^{n \times m}$ be the matrix whose $i$-th row is $\mathbf{c}_\ell$, and
        whose other entries are all zero.
      – Let $\mathbf{A}'_\ell := \mathbf{A}_\ell + \mathbf{C}_\ell$, and $\mathbf{b}'_\ell = \mathbf{b}_\ell + g_i \cdot \mathbf{c}_\ell$.
      – Run the distinguisher $\mathcal{D}$ on input $(\mathbf{A}'_\ell, \mathbf{b}'_\ell)$ and let the output of $\mathcal{D}$ be called $d_\ell$.
  • If $\mathsf{maj}(d_1, \ldots, d_L) = 1$ (meaning that the distinguisher guesses "LWE") then output $g_i$. Else, continue
    to the next iteration of the loop.
```

as described in Algorithm 1 below. First of all, the algorithm partitions the columns of $\mathbf{A}$ into $n \cdot q \cdot \widetilde{O}(\frac{m}{\epsilon^2})$ parts – n for the number of coordinates of $\mathbf{s}$; q for the number of possible guesses for each coordinate; and the rest is what a single iteration of the guessing algorithm uses.

If a guess g_i is correct, i.e. $s_i = g_i$, then the inputs $(\mathbf{A}'_\ell, \mathbf{b}'_\ell)$ given to $\mathcal{D}$ are fresh LWE samples, since

$$\begin{aligned}
\mathbf{b}'_\ell = \mathbf{b}_\ell + s_i \cdot \mathbf{c}_\ell &= \mathbf{s}^T \mathbf{A}_\ell + \mathbf{e}_\ell^T + s_i \cdot \mathbf{c}_\ell^T && \text{(expanding } \mathbf{b}_\ell\text{)} \\
&= (\mathbf{s}^T \mathbf{A}_\ell + s_i \cdot \mathbf{c}_\ell^T) + \mathbf{e}_\ell^T && \text{(rearranging)} \\
&= \mathbf{s}^T (\mathbf{A}_\ell + \mathbf{C}_\ell) + \mathbf{e}_\ell^T && \text{(by construction of } \mathbf{C}_\ell\text{)} \\
&= \mathbf{s}^T \mathbf{A}'_\ell + \mathbf{e}_\ell^T . && \text{(by definition of } \mathbf{A}'_\ell\text{)}
\end{aligned}$$

On the other hand, if the guess g_i is wrong, i.e. $s_i \neq g_i$, then the inputs $(\mathbf{A}'_\ell, \mathbf{b}'_\ell)$ given to $\mathcal{D}$ are uniformly random, since

$$\begin{aligned}
\mathbf{b}'_\ell = \mathbf{b}_\ell + g_i \cdot \mathbf{c}_\ell &= \mathbf{s}^T \mathbf{A}_\ell + \mathbf{e}_\ell^T + g_i \cdot \mathbf{c}_\ell^T \\
&= (\mathbf{s}^T \mathbf{A}_\ell + g_i \cdot \mathbf{c}_\ell^T) + \mathbf{e}_\ell^T \\
&= \mathbf{s}^T \mathbf{A}'_\ell + \mathbf{e}_\ell^T + (g_i - s_i) \cdot \mathbf{c}_\ell,
\end{aligned}$$

and the term $(g_i - s_i) \cdot \mathbf{c}_\ell$ is random and independent of the rest of the terms since (1) $g_i - s_i$ is nonzero and we are assuming that q is prime; and (2) $\mathbf{c}_\ell$ is random and independent of $\mathbf{A}'_\ell$, $\mathbf{s}$ and $\mathbf{e}_\ell$.

It follows that $\mathcal{D}$ will output 1 with probability at least $1/2 + \epsilon$, in the case that $s_i = g_i$. Since we run $\mathcal{D}$ many times, namely $L = c \log n / \epsilon^2$ times (for a sufficiently large constant c), it follows from a Chernoff bound that with probability $1 - 1/n^2$: if the majority of the outputs $d_1, \ldots, d_\ell$ from $\mathcal{D}$ are equal to 1, then we are in the case where $s_i = g_i$, and if not, we are in the case where $s_i \neq g_i$.

Hence, by a union bound, with overwhelming probability, namely at least $1 - 1/n$, Algorithm 1 guesses *all* coordinates of $\mathbf{s}$ correctly. Therefore, applying Algorithm 1 to each coordinate of $\mathbf{s}$ will, with overwhelming probability, correctly output all coordinates $s_1, \ldots, s_n$ of $\mathbf{s}$. □

Improvements.

- Sample-preserving reduction of Micciancio and Mol: Achieve $m' \approx m$. The key is to use ideas from the Goldreich-Levin and Impagliazzo-Naor search to decision reductions which work with pairwise independence as opposed to full independence as we did.
- Runtime scaling with $\text{poly} \log q$: A major problem with the reduction is that the runtime scales linearly with q, which could make the reduction meaningless for large $q \approx 2^n$, even when the LWE problem is likely hard, e.g., when the error has magnitude $q/\mathsf{poly}(n)$. We will sketch a modification of the above reduction which works even when q is large but of a specific form, e.g., $q = 2^k$ is a power of two, or $q = q_1 q_2 \dots q_k$ is a product of many small primes in which case the runtime will scale with $\max_i q_i$.
- Direct reduction from worst-case by Peikert, Regev and Stephens-Davidowitz: This is more relevant in the context of Ring-LWE which we will discuss later in the course.

1.3 A Reduction with $\mathsf{poly}(\log q)$ Runtime

Assume that $q = 2^k$. We show how to make the runtime scale with k rather than 2^k. The key idea (due to Micciancio and Peikert) is to guess each number $s_i \in \mathbb{Z}_q$ (coordinate of the secret vector $\mathbf{s}$) bit by bit, rather than make one guess for every possible value of s_i.

In particular, we will modify the guessing algorithm as follows. Unlike the previous algorithm, this one will employ the following *iterative procedure* for each coordinate, to guess each bit of it in turn, starting from the least significant bit.

- Define distributions $\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_k$ where $\mathcal{D}_i$ produces

  $$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e + r \cdot 2^j \pmod q)$$

  where, as above, $q = 2^k$ and r is uniformly random mod q. Note that $\mathcal{D}_0$ is uniformly random and $\mathcal{D}_k$ is LWE. Since the decisional LWE adversary can distinguish between $\mathcal{D}_0$ and $\mathcal{D}_k$ with a $1/\mathsf{poly}(n)$ advantage, there is a $j \in [k]$ such that it distinguishes between $\mathcal{D}_{j-1}$ and $\mathcal{D}_j$ with advantage at least $1/k \cdot 1/\mathsf{poly}(n)$. Focus on such a j.

- We will now use the distinguisher to learn the LSB of s_1 (and analogously, that of all other s_i) as follows. Given an LWE sample $(\mathbf{a}, b)$, create a sample

  $$(\mathbf{a}', b') = (\mathbf{a} + r \cdot 2^{j-1} \cdot \mathbf{u}_1, b)$$

  where $\mathbf{u}_1$ is the unit vector with 1 in the first coordinate and 0 elsewhere.

  If the LSB of s_1 is 0, then this looks like

  $$(\mathbf{a}', b' = \langle \mathbf{a}', \mathbf{s} \rangle + e + r \cdot 2^j \pmod q)$$

  where $\mathbf{a}'$ and r are uniformly random and independent. On the other hand, if the LSB of s_1 is 1, this looks like

  $$(\mathbf{a}', b' = \langle \mathbf{a}', \mathbf{s} \rangle + e + r \cdot 2^{j-1} \pmod q)$$

  where again, $\mathbf{a}'$ and r are uniformly random and independent.

  A distinguisher that tells these two apart also helps us determine the LSB of s_1 (and analogously, of all the s_i).

- We now proceed in two steps. First, we observe that this can be used to recover the successive bits of $\mathbf{s}$, up to a certain point. We first transform the given LWE sample $(\mathbf{a}, b)$ so that it corresponds to a secret whose LSBs are 0. For example, to go from predicting the LSB to the second least significant bit, we transform $(\mathbf{a}, b) \to (\mathbf{a}, b - \langle \mathbf{a}, \mathsf{LSB}(s_1) \cdot \mathbf{u}_1 \rangle)$.

  From then on, to recover the k-th least significant bit, we do:

$$(\mathbf{a}', b') = (\mathbf{a} + r \cdot 2^{j-k} \cdot \mathbf{u}_1, b)$$

  This ends up being either $\mathcal{D}_{j-1}$ or $\mathcal{D}_j$ depending on whether the k-th LSB of s_1 is either 1 or 0 (respectively).

- However, we can only recover up to j LSBs this way. What do we do with the rest? The key idea is to make sure that j is not too small. To do this, consider the modified distributions $\mathcal{D}'_j$ which output

$$(\mathbf{a}, b + r \cdot 2^j + e' \pmod q)$$

  where $(\mathbf{a}, b)$ is an LWE sample with noise rate αq and e' is a fresh Gaussian with noise rate about αq.

  The effect of doing this is that the distributions $\mathcal{D}'_0, \mathcal{D}'_1, \ldots, \mathcal{D}'_{2^{j'} \approx \alpha q}$ are *statistically indistinguishable*. Thus, the j in question for which the distinguisher succeeds in distinguishing $\mathcal{D}'_{j-1}$ and $\mathcal{D}'_j$ is necessarily larger than j'. This lets us recover j' LSBs of all the s_i. The remaining space has size about $q/2j' \approx 1/\alpha = \mathsf{poly}(n)$.

  To recover this part of the secret, observe the following: if we only had the MSB of the secret to recover and the error was small enough, we would be done. Indeed, b then is a multiple of $q/2$ plus a small amount of noise. From this, we can recover exactly the multiple of $q/2$ which by Gaussian elimination will tell us the MSB of s_1. The key is to extend this argument to recover sufficiently many MSBs, in fact $k - j'$ of them (everything that we couldn't recover by the procedure above).

1.4 A Better Reduction: A Sketch

We will show how to reduce LWE mod q to LWE mod $p \ll q$, with a commensurate noise rate, in two steps. The (very rough) intuition is that the hardness of LWE (for a fixed n) depends on the ratio between the noise magnitude and the modulus, and not on the modulus itself. This suggests that it *should* be possible to scale q while keeping the noise-to-modulus ratio the same. We will show a (sketch of a) formal version of this intuition.

We will proceed in steps.

Idea 1. From LWE to binary secret LWE. We will use an idea of Goldwasser, Kalai, Peikert and Vaikuntanathan [GKPV10]. The rough idea is as follows: look at an LWE input $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$ where $\mathbf{s} \in \{0,1\}^n$. Suppose $\mathbf{A}$ were decomposable into $\mathbf{BC}$ where $\mathbf{B} \in \mathbb{Z}_q^{n \times k}$ and $\mathbf{C} \in \mathbb{Z}_q^{k \times m}$ are uniformly random. The reader should think of $k \approx n/\log q = H_\infty(\mathbf{s})$, the min-entropy of the vector $\mathbf{s}$. Then,

$$\mathbf{s}^T\mathbf{A} + \mathbf{e}^T = \mathbf{s}^T\mathbf{BC} + \mathbf{e}^T = (\mathbf{s}^T\mathbf{B})\mathbf{C} + \mathbf{e}^T$$

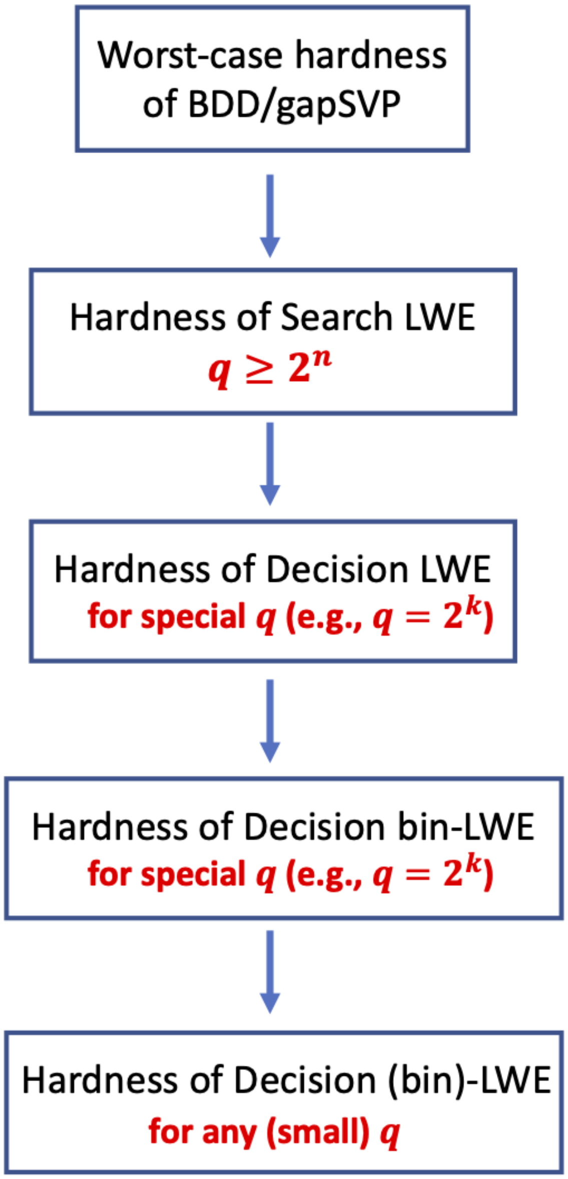
Worst-case hardness of BDD/gapSVP
Hardness of Search LWE $q \geq 2^n$
Hardness of Decision LWE for special q (e.g., $q = 2^k$)
Hardness of Decision bin-LWE for special q (e.g., $q = 2^k$)
Hardness of Decision (bin)-LWE for any (small) q

Figure 1: The Sequence of Reductions from Worst-case BDD/gapSVP to decision LWE for small modulus.

In other words, one can think of this as an LWE input w.r.t. the public matrix $\mathbf{C}$ with the secret being $\mathbf{s}^T\mathbf{B}$. The key point is that multiplication by $\mathbf{B}$ extracts randomness from $\mathbf{s}$ and makes $\mathbf{s}^T\mathbf{B}$ (statistically close to) uniformly random by the leftover hash lemma (LHL). (Clealy, we are omitting details such as the slack between the min-entropy and the output length that LHL needs, but they are not very important to this outline.)

In other words, this says that the LWE input with a binary secret $\mathbf{s}$ w.r.t. $\mathbf{A}$ looks statistically close to an LWE input with a uniformly random secret $\mathbf{s}' := \mathbf{B}^T\mathbf{s}$ which, in turn, is pseudorandom. QED.

If this argument did work, it will prove the hardness of LWE where the secret comes from *any* distribution with sufficient min-entropy (eg $H_\infty(\mathbf{s})/\log q \geq \lambda$ for some security parameter λ.)

There is a major glitch in this argument, however: a matrix of the type $\mathbf{BC}$ has rank at most k, whereas a random matrix $\mathbf{A}$ has rank $n \approx k \log q$. In other words, they are *very distinguishable*.

Goldwasser et al. [GKPV10] nevertheless show how to fix this idea in the following way: assume that $\mathbf{A} = \mathbf{BC} + \mathbf{N}$ where $\mathbf{N}$ is an LWE error matrix. Such a matrix is computationally close to uniform under LWE (with the uniformly random secret matrix $\mathbf{B}$.) Now let's do the calculation again.

$$\mathbf{s}^T\mathbf{A} + \mathbf{e}^T = \mathbf{s}^T(\mathbf{BC} + \mathbf{N}) + \mathbf{e}^T = (\mathbf{s}^T\mathbf{B})\mathbf{C} + (\mathbf{s}^T\mathbf{N} + \mathbf{e}^T)$$

$\mathbf{s}^T\mathbf{B}$ is statistically close to uniform by the argument above. However, the error term is different and it raises two problems: (1) it potentially leaks information about $\mathbf{s}$, ruining the LHL; and (2) it makes the error distribution wonky. A cheap way to get around this problem is to ensure that $||\mathbf{s}^T\mathbf{N}||$ is small, say $\mathsf{poly}(n)$, for example by ensuring that $\mathbf{s}$ is binary and $\mathbf{N}$ has $\mathsf{poly}(n)$-bounded entries, and using the so-called *noise flooding* trick, setting $\mathbf{e}^T$ to be a Gaussian with a superpolynomially larger standard deviation. This ensures that $\mathbf{s}^T\mathbf{N} + \mathbf{e}^T$ looks statistically like a fresh Gaussian, independent of $\mathbf{s}^T\mathbf{N}$. This kills both problems in one shot.

Unfortunately, this means that q has to be larger than the error, ie at least $2^{\omega(\log n)}$ and one has to assume LWE where the noise-to-modulus ratio is $2^{-\omega(\log n)}$. This issue has been resolved in a subsequent work of Brakerski et al. [BLP+13]. Nevertheless, the following question is still open:

Open Problem 4.1. For which distributions of the secret $\mathbf{s}$ does the LWE assumption hold (assuming LWE with uniform secrets holds)?

The most recent development along these lines is the very recent work of Dottling and Brakerski [BD20]. A more concrete question that, to the best of the instructor's knowledge, remains open is the following:

Open Problem 4.2. Does LWE remain hard if the secret vector is a random 0-1 vector with at most $\log n$ ones?

Idea 2. Modulus Reduction. Now, we utilize a technique called "modulus reduction" invented by Brakerski and Vaikuntanathan [BV11] in the context of fully homomorphic encryption.

The rough idea is as follows: Assume that you are given LWE samples $(\mathbf{A}, \mathbf{b})$ with a 0-1 secret relative to a matrix $\mathbf{A}$ mod q. We would like to produce LWE samples modulo p in such a way that solving LWE mod p gives us a solution mod q. Consider computing

$$\left(\left\lfloor \frac{p}{q}\mathbf{A}\right\rceil, \left\lfloor \frac{p}{q}\mathbf{b}\right\rceil\right)$$

The matrix $\mathbf{A'} := \lfloor p/q \cdot \mathbf{A}\rceil$ is uniformly random mod p (modulo boundary issues which can be taken care of with some work.) Now,

$$(p/q)\mathbf{b} = (p/q)\cdot(\mathbf{s}^T\mathbf{A} + \mathbf{e}^T + q\mathbf{z}^T) = \mathbf{s}^T\mathbf{A'} + \mathbf{s}^T\{p/q\mathbf{A}\} + (p/q)\mathbf{e}^T + p\mathbf{z}^T$$

where $\mathbf{z}$ is some integer vector and $\{\cdot\}$ denotes the fractional part of a number (or each number in a matrix). This is almost LWE mod p. Let us analyze the error term. $(p/q)\mathbf{e}^T$ is a Gaussian with parameter αp if $\mathbf{e}$ is Gaussian with paramter αq. Assuming p is quasipolynomially large, one can use the noise-flooding lemma to "absorb" the error $\mathbf{s}^T\{p/q\mathbf{A}\}$ which has polynomially bounded norm. This completes the proof sketch.

We remark that much better versions of this gameplan has been executed successfully by Brakerski, Langlois, Peikert, Regev and Stehlë [BLP$^+$13]. We refer the reader to their paper for more details.

2 Bounded Distance Decoding and LWE

The bounded distance decoding (BDD) problem is a promise variant of the closest vector problem (CVP) on lattices, where the target point is guaranteed to be so close to the lattice that there is a *unique* closest vector. In other words, in the c-BDD problem for a $c \in [0, 1/2)$, one is given a basis $\mathbf{B} \in \mathbb{Z}^{m\times m}$ of a lattice $\mathcal{L}(\mathbf{B})$ and a target vector $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathcal{D}(t, \mathcal{L}(\mathbf{B})) \le c \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$, and the goal is to find the lattice vector that is closest to $\mathbf{t}$.

BDD and LWE are very closely related as the reader may have noticed already. In particular, LWE can be seen as an average-case version of BDD in the following way. Define the LWE lattice

$$\Lambda(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \exists\ \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{z} = \mathbf{s}^T\mathbf{A} \pmod q\}$$

(Note that $q\mathbb{Z}^m \subseteq \Lambda(\mathbf{A}) \subseteq \mathbb{Z}^m$.) It is not hard to show that the minimum distance of $\Lambda(\mathbf{A})$ for a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is $c'q^{1-n/m}$ with high probability. (We will leave this calculation as an exercise.)

LWE is then the regime where the secret $\mathbf{s}$ (which defines the closest vector) is uniquely determined given $\mathbf{s}^T\mathbf{A} + \mathbf{e}^T$.

3 Discrete Gaussians

As we saw in the last lecture, the Gaussian function

$$\rho_s(\mathbf{x}) := e^{-\pi||\mathbf{x}||^2/s^2}$$

from $\mathbb{R}^n$ to $\mathbb{R}$ can be turned into a probability distribution over $\mathbb{R}^n$ by normalizing with $\int_{\mathbb{R}^n}\rho_s(\mathbf{x})d\mathbf{x} = s^n$. Henceforth, we will call this the (n-dimensional) Gaussian distribution N_s. Thus,

$$N_s(\mathbf{x}) = \frac{1}{s^n}\cdot e^{-\pi||\mathbf{x}||^2/s^2}$$

Given a lattice $\mathcal{L}$, we will define the discrete Gaussian distribution $D_{\mathcal{L},s}$ as the probability distribution that assigns the value 0 to all $\mathbf{x} \notin \mathcal{L}$ and the values

$$D_{\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathcal{L})}$$

for every $\mathbf{x} \in \mathcal{L}$. Here, $\rho_s(\mathcal{L}) := \sum_{\mathbf{v}\in\mathcal{L}} \rho_s(\mathbf{v})$.

The latter definition can be generalized to any discrete set; for example, we will let $D_{\mathcal{L}+\mathbf{c},s}$ denote the discrete Gaussian over the lattice coset $\mathcal{L} + \mathbf{c} = \{\mathbf{v} + \mathbf{c} : \mathbf{v} \in \mathcal{L}\}$ which assigns the Gaussian mass (normalized appropriately) to each vector in $\mathcal{L} + \mathbf{c}$ and 0 to all other vectors.

We will also define off-centered versions of these quantities $\rho_{s,\mathbf{c}}$, $N_{s,\mathbf{c}}$ and $D_{\mathcal{L},s,\mathbf{c}}$; for example, $\rho_{s,\mathbf{c}}(\mathbf{x}) := e^{-\pi||\mathbf{x}-\mathbf{c}||^2/s^2}$, and so on.

When s exceeds the smoothing parameter of the lattice $\eta_\epsilon(\mathcal{L})$, the discrete Gaussian over $\mathcal{L}$ starts having a number of nice regularity properties that make it behave essentially as if it were a continuous Gaussian distribution. Some examples follow.

Lemma 2. *For any* $\mathbf{c} \in \mathbb{R}^n$, *and* $s \geq \eta_\epsilon(\mathcal{L})$,

$$\rho_s(\mathcal{L} + \mathbf{c}) \in [1 - 2\epsilon, 1 + 2\epsilon] \cdot \rho_s(\mathcal{L})$$

Proof. Let $\mathbf{c}'$ denote the shortest vector in the lattice coset $\mathcal{L} + \mathbf{c}$. Then,

$$\begin{aligned}
\rho_s(\mathcal{L} + \mathbf{c}) &= \rho_{s,-\mathbf{c}}(\mathcal{L}) \\
&= \mathsf{det}(\mathcal{L}^*) \cdot \widehat{\rho_{s,-\mathbf{c}}}(\mathcal{L}^*) \\
&= \mathsf{det}(\mathcal{L}^*) \cdot \sum_{\mathbf{z}\in\mathcal{L}^*} \widehat{\rho_{s,-\mathbf{c}}}(\mathbf{z}) \\
&= \mathsf{det}(\mathcal{L}^*) \cdot \sum_{\mathbf{z}\in\mathcal{L}^*} e^{2\pi i\langle\mathbf{c},\mathbf{z}\rangle} \rho_{1/s}(\mathbf{z}) \\
&= \mathsf{det}(\mathcal{L}^*) \cdot \left(1 + \sum_{\mathbf{z}\in\mathcal{L}^*\setminus\{\mathbf{0}\}} e^{2\pi i\langle\mathbf{c},\mathbf{z}\rangle} \rho_{1/s}(\mathbf{z})\right) \\
&\in [1 - \epsilon, 1 + \epsilon] \cdot \mathsf{det}(\mathcal{L}^*)
\end{aligned}$$

The claim follows. □

A direct corollary is the following statement about discrete Gaussians modulo sublattices. It says that if you choose a vector from a discrete Gaussian over a dense (rank n) lattice $\mathcal{L}$ and reduce it modulo a sparser (also rank n) lattice $\mathcal{L}' \subseteq \mathcal{L}$, you get a uniformly random element of the finite group $\mathcal{L}/\mathcal{L}'$. This will be instantiated later in the lecture where $\mathcal{L}$ will be an arbitrary lattice and $\mathcal{L}' = q\mathcal{L}$ will be a scaling of it. Here, $\mathcal{L}/\mathcal{L}' \cong \mathbb{Z}_q^n$.

Lemma 3 (Discrete+Continuous Convolution)**.** *Let $\mathcal{L}$ be a lattice. Consider the distribution obtained by sampling a vector $\mathbf{v}$ from the discrete Gaussian $D_{\mathcal{L},s}$ and a vector $\mathbf{w}$ from the continuous Gaussian N_r and adding them together, where $s, r \geq \eta_\epsilon(\mathcal{L}) \cdot \sqrt{2}$ (where ϵ is a negligible function of n). Then, the resulting distribution is statistically close to the continuous Gaussian $N_{\sqrt{r^2+s^2}}$.*

Proof. Consider the distribution Y obtained by adding up the two vectors. Let $t = \sqrt{r^2+s^2}$.

$$\begin{aligned}
Y(\mathbf{x}) &= \sum_{\mathbf{v}\in\mathcal{L}} \Pr_{D_{\mathcal{L},s}}[\mathbf{v}] \cdot \Pr_{N_r}[\mathbf{x}-\mathbf{v}] \\
&= \frac{1}{\rho_s(\mathcal{L})\cdot r^n} \sum_{\mathbf{v}\in\mathcal{L}} \rho_s(\mathbf{v}) \cdot \rho_r(\mathbf{x}-\mathbf{v}) \\
&= \frac{1}{\rho_s(\mathcal{L})\cdot r^n} \sum_{\mathbf{v}\in\mathcal{L}} e^{-\pi||\mathbf{v}||^2/s^2} \cdot e^{-\pi||\mathbf{x}-\mathbf{v}||^2/r^2} \\
&= \frac{1}{\rho_s(\mathcal{L})\cdot r^n} \sum_{\mathbf{v}\in\mathcal{L}} e^{-\pi\left(||\mathbf{v}||^2\cdot(t^2/r^2s^2) - 2\langle\mathbf{x},\mathbf{v}\rangle/r^2 + ||\mathbf{x}||^2/r^2\right)} \\
&= \frac{e^{-\pi||\mathbf{x}||^2\cdot\frac{1}{r^2}\cdot(1-\frac{s^2}{t^2})}}{\rho_s(\mathcal{L})\cdot r^n} \sum_{\mathbf{v}\in\mathcal{L}} e^{-\pi\left(||\mathbf{v}||^2\cdot(t^2/r^2s^2) - 2\langle\mathbf{x},\mathbf{v}\rangle/r^2 + ||\mathbf{x}||^2\cdot(s^2/t^2r^2)\right)} \\
&= \frac{e^{-\pi||\mathbf{x}||^2/t^2}}{\rho_s(\mathcal{L})\cdot r^n} \sum_{\mathbf{v}\in\mathcal{L}} e^{-\pi||\mathbf{v}-s^2/t^2\cdot\mathbf{x}||^2/(rs/t)^2} \\
&= \frac{\rho_t(\mathbf{x})}{t^n} \cdot \frac{t^n}{r^n} \cdot \frac{\rho_{rs/t,s^2/t^2\cdot\mathbf{x}}(\mathcal{L})}{\rho_s(\mathcal{L})} \\
&\in \big[1-\epsilon, 1+\epsilon\big] \cdot \frac{\rho_t(\mathbf{x})}{t^n} \cdot \frac{t^n}{r^n} \cdot \frac{\rho_{rs/t}(\mathcal{L})}{\rho_s(\mathcal{L})}
\end{aligned}$$

where we used Lemma 2 on the numerator since $rs/t \geq \eta_\epsilon(\mathcal{L})$.

By Proposition 4, we have $\frac{\rho_{rs/t}(\mathcal{L})}{\rho_s(\mathcal{L})} \in [1-2\epsilon, 1+2\epsilon]\cdot(r/t)^n$. Put together with the above, we have

$$Y(\mathbf{x}) \in \big[1-3\epsilon, 1+3\epsilon\big] \cdot N_t(\mathbf{x})$$

from which it follows that the statistical distance between the two distributions in question is at most 3ϵ. □

Proposition 4. *Assume that $s_1, s_2 \geq \eta_\epsilon(\mathcal{L})$. Then,*

$$\frac{\rho_{s_1}(\mathcal{L})}{\rho_{s_2}(\mathcal{L})} \in [1-2\epsilon, 1+2\epsilon] \cdot \left(\frac{s_1}{s_2}\right)^n$$

Proof. We have

$$\rho_s(\mathcal{L}) = \mathsf{det}(\mathcal{L}^*) \cdot s^n \rho_{1/s}(\mathcal{L}^*) \in [1-\epsilon, 1+\epsilon] \cdot s^n \cdot \mathsf{det}(\mathcal{L}^*)$$

where the first equality uses Poisson summation and the fact that $\widehat{\rho_s} = s^n\rho_{1/s}$, and the second the definition of the smoothing parameter and the fact that $s \geq \eta_\epsilon(\mathcal{L})$. Thus,

$$\frac{\rho_{s_1}(\mathcal{L})}{\rho_{s_2}(\mathcal{L})} \in [1-2\epsilon, 1+2\epsilon] \cdot \left(\frac{s_1}{s_2}\right)^n$$

□

3.1 Poor Person's Discrete Gaussian Sampling

For the first step of our reduction in the next section, we need an algorithm to sample from the discrete Gaussian distribution $D_{\mathcal{L},s}$ given s and some basis $\mathbf{B}$ of $\mathcal{L}$. Clearly, this is hard to do if $s < 1/\sqrt{n} \cdot \max_i ||\mathbf{b}_i||$ as it will then give us a way to make the vectors of $\mathbf{B}$ shorter, a computationally hard problem. However, one can hope that for significantly larger s, this is possible. Indeed, Gentry, Peikert and Vaikuntanathan [GPV08], following an algorithm of Klein [Kle00], show such a (polynomial-time) algorithm with $s \geq \omega(\sqrt{\log n}) \cdot \max_i ||\mathbf{b}_i||$ (in fact, something slightly stronger but it will not matter to us). Their algorithm samples from a distribution that is negligibly close (in statistical distance) to the discrete Gaussian.

Here, we will make do with something significantly weaker.

We will show a *very* simple algorithm $\mathsf{SimpleDGS}$ that samples from the discrete Gaussian $D_{\mathcal{L},s}$ where $s \geq 2^n \cdot \max_i ||\mathbf{b}_i||$. The algorithm simply samples a vector $\mathbf{v} \leftarrow N_s$ from the continuous Gaussian distribution with parameter s and "rounds" it modulo the parallelepiped $\mathcal{P}(\mathbf{B})$. That is, output

$$\mathbf{v}' = \mathbf{B}\lfloor \mathbf{v} \rceil \in \mathcal{L}(\mathbf{B})$$

To show that this is statistically close to $D_{\mathcal{L},s}$, we calculate the two probabilities:

- $\Pr[\mathbf{w} \sim D_{\mathcal{L},s}] = c \cdot \rho_s(\mathbf{w})$ for some constant normalization factor c.
- $\Pr[\mathbf{w} \sim \mathsf{SimpleDGS}] = c' \cdot \int_{\mathbf{x} \in \mathcal{P}(\mathbf{B})} \rho_s(\mathbf{w}+\mathbf{x}) d\mathbf{x}$.

The intuition is that $\rho_s(\mathbf{w}+\mathbf{x})$ is very close to $\rho_s(\mathbf{w})$ for all the *typical* vectors, that is, vectors of length at most $s\sqrt{n}$. Indeed,

$$\rho_s(\mathbf{w}+\mathbf{x}) = \rho_s(\mathbf{w}) \cdot e^{-\pi(2\langle \mathbf{w},\mathbf{x}\rangle + ||\mathbf{x}||^2)/s^2}$$

It suffices to show that $|2\langle \mathbf{w},\mathbf{x}\rangle + ||\mathbf{x}||^2)/s^2|$ is very small. Note that this quantity is at most $(2||\mathbf{w}||||\mathbf{x}|| + ||\mathbf{x}||^2)/s^2$ by Cauchy-Schwartz. Since $||\mathbf{w}|| \approx s\sqrt{n}$ is the length of the typical vectors (Exercise: Check this!) and $s \gg 2^n \max_i ||\mathbf{b}_i|| \geq 2^n ||\mathbf{x}||$, we are done.

A remark to a reader who might be wondering if this algorithm in fact performs better, i.e., with a smaller s, and if the large s is merely an artifact of our analysis. To show that it is not, the reader is recommended to let $\mathcal{L} = \mathbb{Z}$ and show that for small s, the rounded continuous Gaussian (our distribution) and the discrete Gaussian over $\mathbb{Z}$ are in fact statistically far.

4 From (Worst-case) BDD to (Average-case) LWE

We show the reduction from the worst-case bounded distance decoding problem, which we saw was morally the same as the LWE problem, to the average-case LWE problem.

We will produce LWE samples where the LWE noise are drawn from a continuous Gaussian. It is easy to discretize it and make the noise comes from the rounded continuous Gaussian distribution.

Claim 5. *The vectors $\mathbf{a}_i$ are statistically close to uniformly random in $\mathbb{Z}_q^n$ and independent.*

Proof. By inspection, we see that the probability of getting $\mathbf{a}_i$ is the probability that the discrete Gaussian $D_{\mathcal{L}^*,s}$ lands up in the set $q\mathcal{L}^* + \mathbf{B}^*\mathbf{a}_i$. This is precisely

$$\frac{\rho_s(q\mathcal{L}^* + \mathbf{c})}{\sum_{\mathbf{c}} \rho_s(q\mathcal{L}^* + \mathbf{c})} \tag{1}$$

Regev's BDD to LWE Reduction

```
Input: Lattice basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$, $\mathbf{t} = \mathbf{Bs} + \mathbf{e} \in \mathbb{Z}^n$.
    (For simplicity, we will assume that $||\mathbf{e}||$ is known.)
Output: LWE instance $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{y} \in \mathbb{Z}_q^m$.

Repeat $m$ times:
    ▶ Let $q \geq 2^{2n}$, where $s \geq q\sqrt{2} \cdot \eta_\epsilon(\mathcal{L}^*)$ and $r \geq \sqrt{2} \cdot ||\mathbf{x}|| \cdot \eta_\epsilon(\mathcal{L}^*)$.
    ▶ Sample a vector $\mathbf{v}_i \leftarrow D_{\mathcal{L}^*,s}$.
    ▶ Compute
        $\mathbf{a}_i := (\mathbf{B}^*)^{-1}\mathbf{v}_i = \mathbf{B}^T\mathbf{v}_i \pmod q$ and $b_i := \mathbf{t}^T\mathbf{v}_i + e'_i \pmod q$
    where $e'_i \leftarrow N_r$.

Run the LWE algorithm on input $(\mathbf{A}, \mathbf{b})$ where the columns of $\mathbf{A}$ are the $\mathbf{a}_i$, and output what it outputs.
```

Since $s \geq q\eta_\epsilon(\mathcal{L}^*) = \eta_\epsilon(q\mathcal{L}^*)$, we know by Lemma 2 that

$$\sum_{\mathbf{c}} \rho_s(q\mathcal{L}^* + \mathbf{c}) \in [1 - 2\epsilon, 1 + 2\epsilon] \cdot \rho_s(q\mathcal{L}^*) \cdot q^n$$

and

$$\rho_s(q\mathcal{L}^* + \mathbf{c}) \in [1 - 2\epsilon, 1 + 2\epsilon] \cdot \rho_s(q\mathcal{L}^*)$$

therefore, the ratio in equation 1 is in the range $\frac{1}{q^n} \cdot [1 - 4\epsilon, 1 + 4\epsilon]$. Consequently, the statistical distance is at most 4ϵ. □

Claim 6. *$b_i = \mathbf{s}^T\mathbf{a}_i + e_i$ and e_i is statistically close to a (1-dimensional) continuous Gaussian N_t where $t = ||\mathbf{x}|| \cdot \sqrt{2}\eta_\epsilon(\mathcal{L}^*)$.*

Proof. For the reduction and the proof, we will assume that $||\mathbf{e}||$ is known. This assumption can be removed with more care; we refer to [Reg09] for more details.

Start by noting that

$$\begin{aligned} b_i &= \mathbf{t}^T\mathbf{v}_i + e'_i \pmod q \\ &= (\mathbf{s}^T\mathbf{B}^T + \mathbf{e}^T)\mathbf{B}^*\mathbf{a}_i + e'_i \pmod q \\ &= \mathbf{s}^T\mathbf{B}^T\mathbf{B}^{-T}\mathbf{a}_i + \mathbf{e}^T\mathbf{v}_i + e'_i \pmod q \\ &= \mathbf{s}^T\mathbf{a}_i + e_i \pmod q \end{aligned}$$

where the second equality follows from the definition of $\mathbf{t} := \mathbf{Bs}+\mathbf{e}$ and that of $\mathbf{a}_i$, and $e_i := \mathbf{e}^T\mathbf{v}_i+e'_i$. It remains to analyze the distribution of e_i.

First, e'_i is distributed like $\mathbf{e}^T\mathbf{w}_i$ where $\mathbf{w}_i$ is a continuous Gaussian with parameter $\sqrt{2}\eta_\epsilon(\mathcal{L}^*)$. Thus,

$$e'_i = \mathbf{e}^T(\mathbf{v}+\mathbf{w}) = \mathbf{e}^T\mathbf{w}'$$

where $\mathbf{w}'$ is distributed like $N_{s'}$ by Lemma 3 with

$$s' \approx q \cdot ||\mathbf{x}|| \cdot \eta_\epsilon(\mathcal{L}^*) \leq cq\lambda_1(\mathcal{L})\eta_\epsilon(\mathcal{L}^*) \in cq \cdot [1, \sqrt{n}]$$

by Banaszczyk's theorem. In the worst case, if $c \ll 1/\sqrt{n}$, this gives us an LWE distribution with meaningfully bounded error. □

In summary, the reduction solves $1/\sqrt{n}$-BDD assuming an LWE solver with a constant factor noise-to-modulus ratio.

5 From (Worst-case) SIVP to (Worst-case) BDD

5.1 A Classical Reduction

We now present a classical reduction from gapSVP to BDD due to Peikert [Pei09]. We contrast this with Regev's quantum reduction from SIVP to BDD [Reg09].

The advantage of Peikert's reduction, of course, is that it is classical. However, it is a reduction from a decision problem (gapSVP) to a search problem (BDD), as opposed to Regev's quantum reduction that reduces from search SIVP. For classes of lattices such as ideal lattices, the gapSVP problem for small factors turns out to be easy making the (analogous) reduction vacuous, so it is important to find a reduction starting from a *search* problem. Thus, the following question is wide open.

Open Problem 4.1. Show a (worst-case) reduction from SIVP (or SVP or CVP) to BDD.

We sketch the idea behind Peikert's reduction which in turn draws inspiration from a beautiful *coAM* protocol for gapSVP due to Goldreich and Goldwasser. Let $\mathcal{L}$ be the input lattice with the promise that $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma$. Assume that we have access to a c-BDD solver, namely an algorithm that returns the closest lattice vector given the promise that the target point is within distance $c \cdot \lambda_1(\mathcal{L})$ from the lattice. The reduction works as follows.

- Pick a random lattice point $\mathbf{z} \in \mathcal{L}$ and add a random point $\mathbf{e}$ from a ball of radius $c \cdot \gamma$.
- Run the BDD solver with input $\mathbf{t} := \mathbf{z} + \mathbf{e}$.
- If the BDD solver produces a vector $\mathbf{z}' = \mathbf{z}$, output NO ("large λ_1") else output NO ("small λ_1").

On the one hand, if $\lambda_1(\mathcal{L}) > \gamma$, then the distance of $\mathbf{t}$ from the lattice is at most $c \cdot \lambda_1(\mathcal{L})$ and thus it satisfies the BDD promise. Consequently, the BDD solver will return $\mathbf{z}$. On the other hand, if $\lambda_1(\mathcal{L}) \leq 1$, the (uniform distribution on the) balls centered at $\mathbf{z}$ and $\mathbf{z}+\mathbf{u}$ where $||\mathbf{u}|| = \lambda_1(\mathcal{L})$ are statistically close, *if* $c\gamma \geq \sqrt{n}$. Therefore, a c-BDD algorithm helps us solve $\sqrt{n}/c$-gapSVP.

Putting this together with the worst-case to average-case reduction, we get a $O(n)$-gapSVP algorithm given an LWE solver with constant noise-to-modulus ratio.

References

[BD20] Zvika Brakerski and Nico Dttling. Hardness of lwe on general entropic distributions. Cryptology ePrint Archive, Report 2020/119, 2020. https://eprint.iacr.org/2020/119.

[BLP^{+}13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584. ACM, 2013.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 937–941. ACM/SIAM, 2000.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.