PQCrypto Course Notes
CO 789 - Winter 2024
University of Waterloo
Instructor: Sam Jaques

January 26, 2024

1 Learning With Errors

The goal here is to get to a point where you can read the submission specification and/or documentation for Kyber and Dilithium (lattice-based key encapsulation mechanism and signature scheme, respectively), and roughly understand all the parts. This means that unlike other notes on lattice cryptography, I will not take a historic route, nor will I cover much of the fundamentals of lattices. The outline is instead something like:

1. explore the ins and outs of the LWE problem and its parameters;
2. explain how one can use latttice solvers to attack LWE (and completely ignore the research on the reverse direction)
3. give all the important details that go into the construction of Kyber and Dilithium.

1.1 Basic Definitions

An LWE$(n, m, q, \chi_s, \chi_e)$ instance is formed by sampling a uniformly random $m \times n$ matrix A with entries in $\mathbb{Z}/q\mathbb{Z}$, a vector $s \in (\mathbb{Z}/q\mathbb{Z})^n$ from the distribution χ_s, and a vector $e \in (\mathbb{Z}/q\mathbb{Z})^m$ from the distribution χ_e, and outputting $(A, b := As + e \mod q)$.

The number m is sometimes referred to as the number of "samples".

The LWE$(n, m, q, \chi_s, \chi_e)$ *search* problem is, given (A, b) as sampled above, to recover s.

The LWE$(n, m, q, \chi_s, \chi_e)$ *decision* problem is: a bit $b' \in \{0, 1\}$ is drawn uniformly at random, and if $b' = 0$, then one is given an LWE sample (A, b) as above, and if $b' = 1$, then one is given (A, b) where A is a uniformly random $n \times m$ matrix and b is a uniformly random m-dimensional vector (both with entries in $\mathbb{Z}/q\mathbb{Z}$. The problem is to determine whether $b' = 0$ or $b' = 1$.

Throughout these notes I might equivocate between χ_s being a distribution, a random variable, or the distribution of individual components of s. Some common distributions for χ_s:

1. Uniform in $\mathbb{Z}_q^n$
2. Uniform with exactly t non-zero entries, either all 1 or $\{-1, 1\}$ randomly.
3. Discrete Gaussian (independently chosen in each coordinate)
4. Centered binomial (independently chosen in each coordinate)

Common error distributions are the same, except we do not use uniform error. If we allow uniformly random errors, then LWE is easy to solve: Given (A, b).

1.2 Pathological LWE

LWE has so many parameters, there is a huge space of problems that it captures. There are certain pathological cases which are easy to solve (for search) or impossible to solve (for decision). For example, if χ_e is uniform, then it is easy to solve search LWE: given (A, b), select a random s from χ_s, and declare that as a solution. It satisfies $As + e = b$ for *some* error e. I think you can show that if A is full rank then this has exactly the expected distribution, but intuitively, this is a valid solution.

To avoid such cases, I define "unique LWE", which is just something I made up, not a standard definition:

Definition 1.1. *An LWE problem LWE$(m, n, q, \chi_s, \chi_e)$ is "unique" if, for a matrix A sampled uniformly at random and two pairs (s, e) and (s', e') sampled independently from (χ_s, χ_e) such that $As + e = As' + e'$, then $s = s'$ and $e = e'$ with high probability.*

If the function $(s, e) \mapsto As + e$ is injective, then the LWE instance will be unique. When the values of s and e are strictly bounded, this can be shown, but uniqueness captures the same property when we use a distribution where, with low probability, s and e can take on arbitrary values.

Later we will show that, more or less, we need $\|s\| + \|e\| \leq O(\sqrt{n} q^{1 - \frac{n}{m+n}})$.

1.3 Parameter Reductions

We have five different parameters to vary for LWE. We can make a table of how they change things:

Parameter	Easier	Harder
m	Large	Small
n	Small	Large
q	Large	Small
χ_s	Narrow	Wide
χ_e	Narrow	Wide

Number of samples, m: The parameter m is called the "number of samples" for historical reasons, because one imagined a case where there is an oracle that will give you random vectors a and noisy inner products $\langle a, s \rangle + e$ upon request. So the complexity of the problem can be graded by how many samples you ask for. In practical cryptosystems, you only ever get a fixed number for a given secret s.

Intuitively, each sample gives you some information, so the more samples you get, the more information you have, and the easier the problem gets. Indeed, two simple edge cases: if $m = 1$ the problem is basically impossible (homework), if $m \gg q^n$ then you would expect to get the same row of A many times – each sample will be $\langle a, s \rangle + e$ for *different* e, so you can take the average of these samples and get $\langle a, s \rangle + \overline{e}$, and $\overline{e}$ should be computable from χ_e.

(Much tighter attacks are due to BKW and Arora-Ge. I *think* Arora-Ge gives a poly-time attack if $m = \Omega(n^2 \log^2 q)$, but their result is based on binary secrets and error and it's possible something goes wrong in the reduction).

To *prove* that smaller m is more difficult, we will give a reduction.

Lemma 1.1. *Suppose χ_e is a product distribution (i.e., each element of e is sampled independently and identically). Then LWE$(m, n, q, \chi_s, \chi_e)$ reduces*

to LWE$(m', n, q, \chi_s, \chi_e)$ for $m' \leq m$ as long as the second LWE problem is unique.

Proof. We're given (A, b) as a sample from LWE$(m, n, q, \chi_s, \chi_e)$. We simply give the first m' rows of A and the first m' elements of b to the LWE$(m', n, q, \chi_s, \chi_e)$ oracle.

Call the new problem (A', b'), Since A was uniformly random, so is A'. We also know that $b' = A's + e'$, where e' is the first m' elements of b, and by assumption on the structure of χ_e, we know it still has the right distribution. Thus, this follows the correct distribution for an LWE sample and the oracle will return s' from χ_s such that $As' + e'' = b'$ for some e'' from χ_e. Since the problem is unique, we know $s' = s$, and we solve the original problem. □

Notice that uniqueness is necessary for this proof to go through. We can't reduce anything to LWE with $m = 1$ because it's trivial to solve this (randomly sampled s and e have a $1/q$ chance of equalling b).

Secret dimension n: Intuitively, larger secrets should be harder to find. We can prove this with a reduction:

Lemma 1.2. *LWE$(m, n, q, \chi_s, \chi_e)$ reduces to LWE$(m, n', q, \chi_s \times \chi'_s, \chi_e)$ for any $n' \geq n$ and any distribution χ'_s on $\mathbb{Z}_q^{n'-n}$, as long as the second is unique.*

Proof. Here the notation $\chi_s \times \chi'_s$ means that we sample the first n coordinates from χ_s and the next $n' - n$ from χ'_s.

We're given (A, b) as a LWE$(m, n, q, \chi_s, \chi_e)$ sample. We generate $n' - n$ random columns A', and sample a random s' from χ'_s. Then we give $([A|A'], b + A's')$ to the LWE$(m, n', q, \chi_s \times \chi'_s, \chi_e)$ oracle.

Since A' was uniformly random, so is $[A|A']$. Then $b = As + e + A's'$, so it follows the right distribution as well, and the oracle will return some s''. By uniqueness, $(s, s') = s''$, so we can just take the first n coordinates and we solved the first problem. □

Modulus q: This one is a bit surprising, but larger q is easier to solve. One way to think of it is that if we hold the errors fixed, then the size of errors relative to the space that they live in will shrink. Later, lattice attacks will make this more precise.

We'll show just a simple reduction for this one. There are more general reductions, but they are approximate and annoying; I only want to give the flavour of things here.

Lemma 1.3. *LWE$(m, n, qp, \chi_s, \chi_e)$ reduces to LWE$(m, n', q, \chi_s, \chi_e)$ for any natural numbers q, p such that both problems are unique, and where χ_s and χ_e produce values that are bounded by q with all but negligible probability.*

Proof. Given (A, b) as an instance of the first problem, we just give $(A \mod q, b \mod q)$ to the second oracle. Since $b \equiv As + e \mod pq$, this algebraic relation survives the modular reduction: $b \equiv As + e \mod q$ as well.

Since $s = (s \mod q)$ and $e = (e \mod q)$ with high probability (by assumption on χ_s and χ_e), this new b satisfies the right distribution and the oracle will give us some $s' \in \mathbb{Z}_q^n$ such that $As' + e' \equiv b \mod q$. By uniqueness of the second problem, $(s', e') = (s, e) \mod q$ (since (s, e) is a valid solution to the problem mod q). But since $s = (s \mod q)$, then $s' = s$ and we are done. □

As an exercise, you could show that this also holds with uniform χ_s.

1.3.1 Probability Distributions

Intuitively, the more noise we add, the harder this problem should get. One edge case (no noise at all) is definitely easy. We'll prove this with the following lemma:

Lemma 1.4. *LWE$(m, n, q, \chi_s, \chi_e)$ reduces to LWE$(m, n, q, \chi_s + \chi'_s, \chi_e)$ for any distribution χ'_s such that the second problem is unique.*

Here I'm abusing notation somewhat: $\chi_s + \chi'_s$ means that we sample s from χ_s, then sample s' from χ'_s, then add them together.

Proof. Given $(A, b = As + e)$ from the first problem, we sample s' from χ'_s, and give $(A, b + As')$ to the second oracle. Clearly $b + As' = A(s + s') + e$, so it has the right distribution and we get a response s''. By uniqueness, $s'' = s + s'$; we subtract s' from s'' to get s. □

This was maybe the easiest proof, but it has many implications:

- The exact same proof works for e.
- We can shift any distribution by adding a constant distribution, and so we can assume $\mathbb{E}[\chi_s] = 0$.
- Symmetric distributions must be the hardest type, since we can just add $\chi_s + (-\chi_s)$ and this is symmetric about its mean (which is 0 WLOG)

- We cannot necessarily add all distributions, because then LWE might not be unique. But for tall LWE, this shows that uniform secrets are the hardest type of secret (adding a uniform secret distribution to any other distribution makes it uniform).
- A Gaussian distribution ought to be hardest. The reasoning is that if we take k independent and identically distributed random variables S_i, then $S_1 + \cdots + S_k$ converges to normal by the central limit theorem. Thus, for almost any distribution, there is a harder distribution which is normal (albeit wider). The only exception is if we do not have any room to make the error wider without losing uniqueness.

For a lot of our early analysis, a uniformly bounded error would be easiest to analyze (i.e., choose each component of e between $-\beta$ and $+\beta$ for some β). However, in practice we use Gaussians or approximations to Gaussians in most cases, mostly because the lattice reductions only work with Gaussians but partly because of the above reasoning.

1.4 Two main forms of LWE

There are two main forms of LWE that we use in practice.

Tall LWE: In this case we choose $m > n$ and (often) use $\chi_s = U$, the uniform distribution. The hardness in this case ends up being that the span of A is only an n-dimensional subspace of $\mathbb{Z}_q^m$, and we're given something near to that subspace.

Normal form LWE: Here we take $m = n$ and $\chi_s = \chi_e$. Since χ_e must always be short, this means both secret and error are short. Here we expect the image of A to cover the entire space, but the problem is that the preimage of most points is large.

It turns out that these are equivalent to each other.

Lemma 1.5. *Suppose χ_e is a product of iid variables in each component. Then LWE$(m, n, q, \chi_s, \chi_e)$ reduces to LWE$(m - n, n, q, \chi_e, \chi_e)$.*

Proof. We're given $(A, b = As + e)$, and we assume the first n rows of A are invertible without loss of generality (we are free to permute A, and we can make A have rank n with a reduction from the homework). Let $A = [A_0^T | A_1^T]^T$. This way we can write

$$b = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} s + \begin{pmatrix} e_0 \\ e_1 \end{pmatrix}. \tag{1}$$

Notice then that

$$A_1 A_0^{-1} b_0 = A_1(s + A_0^{-1} e_0) = A_1 s + A_1 A_0^{-1} e_0 \tag{2}$$

Which contains $A_1 s$, the first part of b_1! So we subtract b_1 from this:

$$b' := A_1 A_0^{-1} b_0 - b_1 = A_1 s + A_1 A_0^{-1} e_0 - A_1 s - e_1 = A_1 A_0^{-1} e_0 - e_1 \tag{3}$$

This looks like an LWE sample. Indeed, we can pass $(A_1 A_0^{-1}, b')$ to the second oracle. We see that $A_1 A_0^{-1}$ has dimensions $(m-n) \times n$, and it is uniformly random, and of course e_0 and e_1 are distributed according to χ_e. □

It turns out we can also reverse this. This was something I worked out with Romy Minko (we're slightly surprised it does not appear in the literature anywhere).

Lemma 1.6. *Full-rank LWE*$(n, n, q, \chi_s, \chi_e)$ *reduces to LWE*$(2n, n, q, U, \chi_s \times \chi_e)$.

Where "full-rank" means we restrict A to be full rank, and $\chi_s \times \chi_e$ means we sample the first n components of the error from χ_s, and the remaining n components from χ_e.

Proof. We will simply reverse the previous proof. We're given $(A, b = As + e)$. Since A is full-rank, we can select random A_0 as an invertible matrix and let A_1 be such that $A = A_1 A_0^{-1}$. Then we will select a uniformly random $b_1 \in \mathbb{Z}_q^n$, and we will declare by fiat that $b_1 = A_1 s' + e$ for some s' (more concretely, $s' = A_1^{-1}(b_1 - e)$; since b_1 is uniformly random, so is s').

For this s' that we defined, we want to find $A_0 s' + s$. To do this, we compute

$$\begin{aligned}
b_0 :=& A^{-1}(b - b_1) && (4)\\
=& A^{-1}(As + e - A_1 s' - e) && (5)\\
=& A^{-1}(As - A_1 s') && (6)\\
=& s - A^{-1} A_1 s' && (7)\\
=& s - A_0 A_1^{-1} A_1 s' && (8)\\
=& s - A_0 s' && (9)
\end{aligned}$$

Thus, if we set $A' = (-A_0^T | A_1^T)^T$, then we see that

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = A' s' + \begin{pmatrix} s \\ e \end{pmatrix} \tag{10}$$

which fits the distribution of the second problem. We pass $(A', b' = (b_0^T, b_1^T)^T)$ to the oracle, which returns some $\tilde{s}$ satisfies $A'\tilde{s} + e' \equiv b' \mod q$ for e' from $\chi_s \times \chi_e$.

Here we do not even need uniqueness, actually. More precisely, we know that $b_0 = -A_0\tilde{s} + e'_0$ and $b_1 = A_1\tilde{s} + e'_1$. This means that

$$Ab_0 = A_1A_0^{-1}b_0 = -A_1\tilde{s} + Ae'_0 \tag{11}$$

so

$$Ab_0 + b_1 = -A_1\tilde{s} + Ae'_0 + A_1\tilde{s} + e'_1 = Ae'_0 + e'_1 \tag{12}$$

But critically, we constructed $b_0 = A^{-1}(b - b_1)$, so $Ab_0 + b_1 = b$. Thus, $b = Ae'_0 + e'_1$, where e'_0 is distributed as χ_s and e'_1 is distributed as χ_e. This is exactly what we needed. □

1.5 Search and Decision

The search and decision versions of LWE reduce to each other. One direction is easy; for the other:

Lemma 1.7. *Search LWE$(m, n, q, \chi_s, \chi_e)$ reduces to $O(nq)$ calls to LWE$(m, n, q, \chi_s, \chi_e)$.*

Proof. Given a sample (A, b), we guess the first component of s: call it s'_1. Select a uniformly random vector v and add it to the first column of A, and we also subtract vs'_1 from b. Let A' be the new matrix, so that $A = A' + v(1, 0, \ldots, 0)$. Then since b is an LWE sample, we have

$$b = As + e = (A' + v(1, 0, \ldots, 0))s + e = A's - vs_1 + e \tag{13}$$

Thus, $(A', b + vs'_1) = (A', A's + e + v(s'_1 - s_1))$. If we guessed incorrectly, then the sample is uniformly random because v is uniformly random. If we guessed correctly, the terms cancel out and this is just a valid LWE. Thus, with only q guesses, we recover one component of s; after n repetitions of this we have all of s. □

Notice that this proof is not a polynomial time reduction if q itself is superpolynomial. Is this a problem? Well, we argued that the problem gets easier with large q, so this isn't a big problem. In Kyber for example, $q = 3329$.

1.6 LWE Encryption

We will describe a basic public key encryption method based on LWE. This is *not secure*, never use it directly.

A public key encryption scheme consists of three algorithms:

- $\mathsf{KeyGen}() \to (\mathbf{PK}, \mathbf{SK})$
- $\mathsf{Enc}(\mathbf{PK}, m) \to c$
- $\mathsf{Dec}(\mathbf{SK}, c) \to m$

Intuitively, **PK** is the public key, **SK** is the secret key, m is a plaintext message, and c is a ciphertext.

To make our LWE encryption, we have parameters $(n, q, \chi_s, \chi_e, \chi'_s, \chi'_e, \chi''_e)$.

For $\mathsf{KeyGen}()$, we sample s from χ_s, e from χ_e, and uniformly random $A \in \mathbb{Z}_q^{n \times n}$. We let s be our private key and $(A, b = As + e)$ be our public key.

For $\mathsf{Enc}(\mathbf{PK} = (A, b), m)$, we make a "transposed" LWE sample by sampling r from χ'_s, e' from χ'_e, and e'' from χ''_e, and make the ciphertext as $c = (c_1, c_2)$, where:

$$c_1 = r^T A + e'^T,\ c_2 = r^T b + e'' + m \left\lfloor \frac{q}{2} \right\rceil \tag{14}$$

(it was pointed out that $\left\lfloor \frac{q}{2} \right\rceil$ is sort of pointless because q is an integer; we could just take the floor, e.g. these choices don't really matter, as we will see).

For $\mathsf{Dec}(\mathbf{SK} = s, c = (c_1, c_2))$, we can basically think of $r^T b$ as a one-time pad: since b should look uniformly random (by hardness of decisional LWE), this should look uniformly random in $\mathbb{Z}_q$. But how to find $r^T b$? Well, consider that

$$r^T b = r^T A s + r^T e \tag{15}$$

The first part we can figure out, almost, since $c_1 s = (r^T A + e'^T)s = r^T A s + e'^T s$. Thus, we'll subtract $c_1 s$ from c_2 and see where that gets us:

$$c_2 - c_1 s = r^T A s + r^T e + e'' + m \left\lfloor \frac{q}{2} \right\rceil - r^T A s - e'^T s = \underbrace{r^T e - e'^T s + e''}_{(A)} + m \left\lfloor \frac{q}{2} \right\rceil. \tag{16}$$

We still need to get rid of (A). Actually, we won't, really: the key fact is that it's *small*. At least, as long as all the error distributions produce small

vectors, by the triangle inequality and Cauchy-Schwarz inequality, all of (A) is small.

Notice that $c_2 - c_1 s \in \mathbb{Z}_q$, i.e., it's a scalar. Thus, if (A) has absolute value less than $\frac{q}{4}$, the we can recover whether $m = 0$ or $m = 1$ by rounding $c_2 - c_1 s$ to either 0 or $\left\lfloor \frac{q}{2} \right\rceil$, whichever is closer. (If the error is larger than $\frac{q}{4}$, this fails!).

Unfortunately, this only allows us to send $m \in \{0, 1\}$, i.e., a single bit. That's technically enough for public key encryption, and in fact we can send many ciphertexts from the same public key. But this scheme is *big*! Notice the sizes to send k bits:

- Public key: $(n+1)n \lg(q)$ bits
- Ciphertext: $(n+1)k \lg(q)$ bits

And the computations are also bad:

- Encrypt: $O(kn^2)$ (we must do k multiplications of an n-dimensional vector against an n-dimensional matrix)
- Decrypt: $O(kn)$ (we must do k inner products of ciphertexts with our n-dimensional secret).

There are worse schemes, but there are also better schemes. Later we will work on improving the efficiency.

2 Lattices

Why is LWE called lattice cryptography? What is a lattice? Our goal in this section will be to explore this connection, but it will mainly be from an attacker's point of view. It turns out the best known way to attack LWE deployed in practice is by solving lattice problems (by analogy, the best way to attack good RSA schemes is by factoring). I will point out that there are many other ways to attack LWE; there is good survey paper at [**?**].

Definition 2.1. *A lattice is a set $L(B)$ formed by a set B of m vectors in $\mathbb{R}^n$, which are all linearly independent, defined as:*

$$L(B) = \left\{ \sum_{i=1}^{m} a_i b_i \mid a_i \in \mathbb{Z}, b_i \in B \right\} \tag{17}$$

In other words, it's like a vector space, but instead of arbitrary linear combinations, we're only allowed integer coefficients.

Another definition is a discrete additive subgroup of $\mathbb{R}^n$. It's complicated why they're equivalent: `https://www.ams.jhu.edu/~abasu9/RFG/lecture_notes.pdf`.

2.1 Short Vectors

Because it is a discrete subgroup, there exists a shortest vector (possibly non-unique). We can thus define:

$$\lambda_1(L) := \{\text{length of the shortest non-zero vector in } L\}. \tag{18}$$

This is hard to find!

Definition 2.2. *γ-SVP: Given a lattice basis B, find a vector v in $L(B)$ such that $|v| \leq \gamma\lambda_1(L)$.*

Why is this hard? If I'm given a basis B, why not just output the smallest vector in B? Generally this will not be good.

Theorem 2.1 (Minkowski)**.** $\lambda_1(L) \leq \sqrt{n}|\det(B)|^{1/n}$

This holds for any B. In fact:

Proposition 2.1. *If B and B' are two bases of the same lattice L, then $|\det(B)| = |\det(B')|$.*

Proof. Each vector in B' can be written as an integer linear combination of vectors in B. If we abuse notation and let B and B' be matrices where the vectors themselves are the columns, this means $B' = BU$, where U is matrix with integer coefficients.

But the same logic applies in reverse, since B' also generates L. Thus, $B = B'V$ for a matrix V with integer coefficients. Substituting, we get that $B = BUV$.

Since B and B' have full-rank (we could project to their span if not), $\det(B) \neq 0$, so $1 = \det(U)\det(V)$, meaning $\det(U) = \det(V)^{-1}$. But we also know $\det(V) \in \mathbb{Z}$, since it has integer entries, and there are only two invertible integers: 1 and -1. Thus, $\det(B) = \pm\det(B')$. □

From this we can define $\text{Vol}(L) = |\det(B)|$ for any basis, and this is well-defined.

Minkowski's theorem is true for any lattice. And we can further define the Hermite constant, γ_n, such that

$$\lambda_1(L) \leq \gamma_n \text{Vol}(L)^{1/n} \tag{19}$$

for any lattice L. But there are worst-case lattices out there, and we want to do a little better, so there is something called the "Gaussian heuristic": for a random lattice L,

$$\lambda_1(L) \approx \sqrt{\frac{n}{2\pi e}} \text{Vol}(L)^{1/n}. \tag{20}$$

There are measures you can use for lattices to define a "random" lattice such that this is asymptotically true.

There are many more variants of lattice problems! A big one which is important is SIVP:

Definition 2.3. *The ith successive minimum of a lattice L, denoted $\lambda_i(L)$, is*

$$\min \{\max\{\|v\| : v \in B\} \mid B \subseteq L \text{ has } i \text{ vectors which are LID over } \mathbb{R}\} \tag{21}$$

SIVP asks us to find the set B. SIVP is a funny problem because $\lambda_n(L)$ can be shorter than the length of the shortest basis. The classic example is the basis

$$\{e_1, e_2, \ldots, e_{n-1}, \frac{1}{2}(e_1 + \cdots + e_n)\} \tag{22}$$

Notice that e_n is in the lattice as well, so $\{e_1, \ldots, e_n\}$ are linearly independent and shorter than this basis (for $n \geq 5$), but do not span the lattice.

2.2 Close Vectors

For any $t \in \mathbb{R}^n$, it is well-defined to ask for $\|t - L\|$, since there is a minimum distance.

We can then define Bounded Distance Decoing:

Definition 2.4. *Bounded Distance Decoding problem: Given β, a lattice L, and a vector $t \in \mathbb{R}^n$, with the promise that $\|t - L\| \leq \beta$, find t.*

Notice how this "morally" is identical to LWE. The main differences are (a) the secret distribution; (b) doing things mod q.

2.3 Connection to LWE

There are two routes to connect LWE to lattices.

First, and what started LWE cryptography, was Regev's reduction from 2005 from SVP to LWE. That is, if we can solve LWE in polynomial time, then we can solve SVP in polynomial time. That's not quite right, and there's some issues with the reduction:

- More precisely, his reduction was from LWE$(m, n, q, \chi_s, \chi_e)$ where χ_e is a discrete Gaussian distribution of variance σ^2 and χ_s uniform, and requires $\sigma > 2\sqrt{n}$. Then it solves SIVP for approximation factor $O(nq/\sigma)$. Often one assumes $q = \Omega(n)$, so this is a $O(n^{3/2})$ approximation factor.

- In practice, we often choose $\sigma = O(1)$. Then the reduction simply fails.

- The reduction is *quantum*. That means if we have an LWE solver (classical or quantum), it only gives us a quantum SIVP solver. Granted, since this is post-quantum cryptography, we expect SIVP to hard for quantum computers anyway.

- There are huge tightness losses in this reduction. What this means is that if the runtime of our LWE solver is $t(n)$, there is some function $R(n)$ such that we get a $t(n)R(n)$-time algorithm for SIVP. If we assume that SIVP is hard, and would take some time $T(n)$, then our LWE runtime is bounded by $T(n)/R(n)$. However, $R(n)$ is so large that this bound is meaningless in practice except for spectacularly large, impractical n. A good paper to discuss this is [**?**].

The second route is to reduce LWE to SVP: we use an SVP solver to attack LWE. The route will be something like Figure 1.

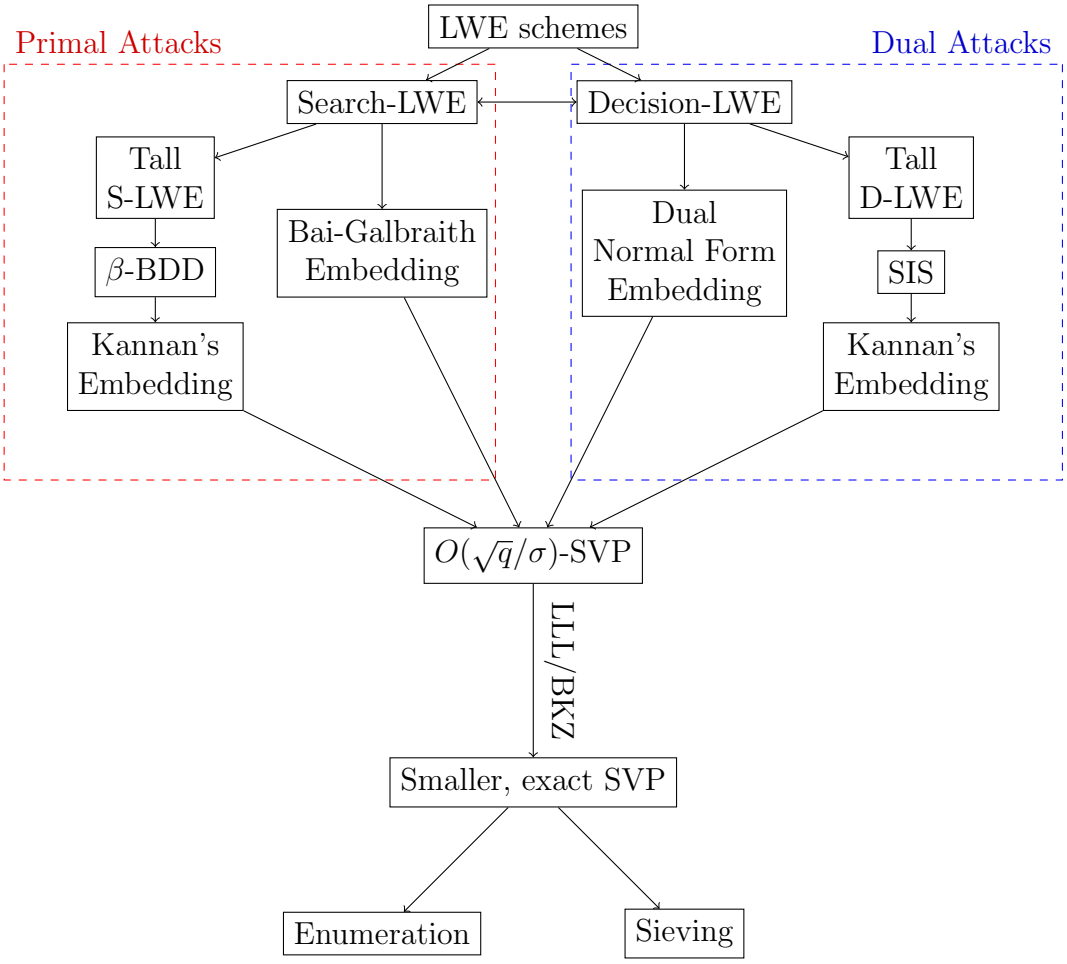


Figure 1: The route of reductions to attack LWE with lattice solvers.

We can see straightforwardly how search and decision LWE break lattice schemes (though we'll see more later), and we saw the reductions between search and decision. We'll continue from here.

2.3.1 Norms of Secret and Error

I will make a brief aside to bound the norms of s and e, as this will be helpful for all the reductions. I'll derive this just for e, though the same logic applies to s.

We can see that

$$\|e\|^2 = |e_1|^2 + \cdots + |e_m|^2 \tag{23}$$

First, we will suppose s and e follow product distributions, meaning every component is i.i.d. (independent and identically distributed). This means we can define a random variable E which follows the distribution of $|e_i|^2$, and we have that

$$\|e\|^2 = E + \cdots + E \tag{24}$$

The sum of m i.i.d. random variables converges to a normal distribution by the central limit theorem! So $\|e\|^2 \sim N(m\mu, mV^2)$, where μ and V^2 are the mean and variance of E. Two key facts:

1. We can assume $\mathbb{E}[e_i] = 0$, since we proved uncentered distributions are equivalent to centered;
2. This gives us $\mathbb{E}[E] = \mathbb{E}[|e_i|^2] = \mathrm{Var}(e_i) = \sigma^2$, where σ^2 is the (componentwise) variance of the original error distribution χ_e.

Thus, we end up with $\mathbb{E}[\|e\|^2] = m\sigma^2$ and it is normally distributed. Now, it is generally false that $\sqrt{\mathbb{E}[X^2]} = \mathbb{E}[X]$, but here we will assume that to be roughly the case. You can find derivations online that show that this is close to true for a normal distribution.

This means the expected value of $\|e\|$ is approximately $\sqrt{m}\sigma$.

Finally we can "prove" a "lemma":

Lemma 2.1. *Suppose that $\chi_s = \chi_e$, and they have variance σ^2. Then for $n \to \infty$, LWE is "unique" iff $\sigma < \frac{q^{\frac{m}{m+n}}}{\sqrt{2\pi e}}$.*

Proof. This will be a pretty sketchy proof. Assume that χ_s is really χ_s^n, i.e., iid components. Then $s_1^2 + \cdots + s_n^2$ converges to a Gaussian by the central limit theorem, where each s_i^2 has mean σ^2, the variance of χ_s. Stealing results from here: `https://stats.stackexchange.com/questions/241504/central-limit-theorem-for-square-roots-of-sums-of-i-i-d-random-variables`, the expected value of $\|s\|$ can be approximated by $\sqrt{n\sigma^2 - \frac{V^2}{4\sigma^2}} \approx \sigma\sqrt{n}$. Another way to see this is that Chebyshev's inequality tells us that

$$\Pr\left(|\|s\|^2 - n\sigma^2| \geq \alpha\right) \leq frac1nV^2\alpha^2 \tag{25}$$

and since $|s|$ is positive we can conclude that

$$\Pr\left(\sqrt{n\sigma^2 - \alpha} \leq \|s\| \leq \sqrt{n\sigma^2 + \alpha}\right) \leq \frac{1}{nV^2\alpha^2} \tag{26}$$

so we're darn close to $\sigma\sqrt{n}$.

We can define a lattice $L(A) = \{(x, y) : Ax \equiv y \mod q\}$, which has a basis

$$\begin{pmatrix} I_n & 0A & qI_m \end{pmatrix} \tag{27}$$

and thus has determinant q^m. We thus expect the shortest vector to have norm approximately $\sqrt{\frac{m+n}{2\pi e}} q^{\frac{m}{m+n}}$.

Now, if we have $As + e = As' + e' \mod q$, then $A(s - s') \equiv e' - e \mod q$. That is $(s - s', e' - e) \in L(A)$; however, this vector is short: it has norm approximately $\sqrt{n + m}\sigma$ by the above. Since we expect the shortest *non-zero* to have norm at least $\sqrt{\frac{m+n}{2\pi e}} q^{\frac{m}{m+n}}$, and thus we need $\sigma \approx \frac{q^{\frac{m}{m+n}}}{\sqrt{2\pi i}}$. □

As an exercise, spot all the logical errors in the above. But it mostly works out: non-unique LWE samples give a short vector in that lattice and very short vectors probably do not exist in it.

2.4 Primal Attacks

2.4.1 Search LWE to BDD

To reduce tall LWE$(m, n, q, \chi_s, \chi_e)$ to $\|e\|$-BDD, we construct the lattice

$$L_{BDD}(A) = \{x \in \mathbb{Z}^m \mid \exists y : Ay \equiv x \mod q\}. \tag{28}$$

Basically, take the image of A and shift it by q. We can write this as something like $Im(A) + q\mathbb{Z}^m$.

It's clear that $As \mod q$ is in this lattice, and this is close to b by distance $\|e\|$, so that's the reduction.

Can we reduce in reverse? Not really. The problem is that the LWE lattice is quite special. First, it's a q-ary lattice. A q-ary lattice is a lattice L such that $q\mathbb{Z}^n \subseteq L$.

Being q-ary is not that special:

Proposition 2.2. *If L is an integer lattice, then it is $vol(L)$-ary.*

Proof. To see this, recall that $B^{-1} = \frac{1}{\det(B)}\text{adj}(B)$, where the adjugate is a matrix of determinants of submatrices of B. This means $adj(B)$ is an integer matrix, so

$$I = BB^{-1} \tag{29}$$

$$= B\frac{1}{\det(B)}\text{adj}(B) \tag{30}$$

$$\det(B)I = B\text{adj}(B) \tag{31}$$

so $\det(B)I$ can be formed by integer combinations of vectors in B. □

Notice that $q\mathbb{Z}^n$ is a q-ary lattice, but $\text{Vol}(q\mathbb{Z}^n) = q^n$. So there can be quite a "gap" between these two bounds. In fact, the LWE lattice we constructed above has determinant q^{m-n}, approximately. We can show this precisely:

For a matrix A, we can write a basis of $L_{BDD}(A)$ as

$$\begin{pmatrix} qI_{m-n} & A_1A_0^{-1} \\ 0 & I_n \end{pmatrix} \tag{32}$$

where $A = [A_0; A_1]$ for A_0 full-rank in $\mathbb{Z}_q$. Most matrices B cannot be transformed into this shape with only integer operations. And indeed, BDD is a "worst-case" problem, whereas LWE is an average-case problem.

2.4.2 BDD to SVP

But, how do we then solve BDD? We can solve BDD with an approximate SVP solver using Kannan's embedding.

Lemma 2.2. *Let B be a basis for a lattice L. BDD($L, \beta < \frac{\lambda_1(L)}{\sqrt{2}}$) reduces to 1-SVP.*

Proof. We simply use Kannan's embedding. Let $\mu = \beta$. Suppose v is a closest lattice point to t, so that $\|t - v\| \leq \beta$, and $v = Bw$ for some w. Then let

$$B' = \begin{pmatrix} B & -t \\ 0 & \mu \end{pmatrix} \tag{33}$$

and we see that $B'(v, 1) = (v - t, \mu)$. The norm of this is $\sqrt{\mu^2 + \|v - t\|^2} \leq \sqrt{2\beta^2} = \sqrt{2}\beta < \lambda_1(L)$.

Suppose $v \in L(B')$ has norm $\|v\| = \lambda_1(L(B')) \leq \sqrt{2}\beta$, since we already constructed a vector of norm $\sqrt{2}\beta$. Then $v = (v' - nt, n\mu)$ for some $v' \in L$ and $n \in \mathbb{Z}$. If $|n| \geq 2$, then $\|v\| \geq 2\beta > \lambda_1(L)$; a contradiction. If $n = 0$, $\|v\| = \|v'\| \geq \lambda_1(L) > \sqrt{2}\beta$, another contradiction. Finally, if $n = 1$, any vector shorter than $\sqrt{2}\beta$ solves the BDD problem anyway. □

To put that into the context of our LWE problem, the vector $(e, 1)$ will be in the lattice. Finding e is of course equivalent to breaking the scheme.

Proposition 2.3. *The shortest vector in the Kannan embedding of a primal LWE attack has heuristic norm* $\sqrt{\frac{m+1}{2\pi e}} q^{1-\frac{n-1}{m+1}}$.

Proof. Using Equation 32 and 33, this is an upper-triangular matrix so the determinant is just the product of the diagonals, which is $q^{m-n}\beta$. Recall that the Gaussian heuristic says that an $m+1$-dimensional lattice satisfies

$$\lambda_1(L) \approx \sqrt{\frac{m+1}{2\pi e}} \mathrm{Vol}(L)^{1/(m+1)} = \sqrt{\frac{m+1}{2\pi e}} q^{\frac{m-n}{m+1}} \beta^{\frac{1}{m+1}} \tag{34}$$

and that's exactly what we need. □

Here's something odd, though: the vector $(e, 1)$ is in the lattice, and recalling our previous analysis, $\|e\| \approx \sqrt{m}\sigma$. Unless $\sigma \approx q^{1-n/m}$, this is much shorter than expected!

There's two approaches here, roughly: One is to argue that this is a "unique-SVP" problem. That is, the *second*-shortest vector in the Kannan embedding should approximately match the Gaussian heuristic. If it does, then if we can solve γ-SVP for any $\gamma \leq \frac{\lambda_2(L)}{\lambda_1(L)}$, then this vector must be the shortest vector (or an integer multiple of it).

In our case, we would have $\lambda_2(L) \approx \sqrt{\frac{m+1}{2\pi e}} q^{1-\frac{n-1}{m+1}}$, and $\lambda_1(L) \approx \sqrt{m}\sigma$. Dividing these gives $\gamma \approx \frac{q^{1-\frac{n}{m}}}{\sigma}$.

The second approach (used in Kyber's security analysis) is to use the shortness of $(e, 1)$ to give a more refined analysis of BKZ, which we will discuss later. This is a more complicated analysis, so I will ignore it.

The analysis of this approximation factor and how it works is still a topic of active research. Like a lot of topics in crypto, we treat this as fact based mostly on experiments, rather than analysis. Why? Because if it works for an attacker, it doesn't matter if they can prove it!

The conclusion in either case is that we get an approximation factor which increases in q and decreases in σ. This makes sense: larger approximation factors in SVP are easier, and we should have that larger q is easier and smaller σ is easier.

2.4.3 Bai-Galbraith Embedding

Kannan's embedding gives us only a vector e such that $b-e$ is in the span of A, modulo q. But that only gives us s if s is uniform, i.e., this only attacks "tall" LWE. We need a different reduction for normal form.

We can define a lattice $L(A,e) = \{(x,y,z) \in \mathbb{Z}^{m+n+1} : Ay + x \equiv bz \mod q\}$. This has a basis:

$$B = \begin{pmatrix} qI_m & A & -b \\ 0 & I_n & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{35}$$

Notice that the vector $(e,s,1)$ is in this lattice. As long as χ_s and χ_e are small, this vector is short.

We can apply the same logic as before. Since this is normal form, we assume $\chi_s = \chi_e$, so

$$\|(e,s,1)\|^2 = \|e\|^2 + \|s\|^2 + 1 = m\sigma^2 + n\sigma^2 + 1 \approx (m+n)\sigma^2. \tag{36}$$

(if it's normal form we also have $m = n$, which I will substitute later).

And we can also compute $\text{Vol}(L(B)) = q^m$, while it's dimension is $m + n + 1$.

We can again apply the Gaussian heuristic here as well:

$$\lambda_1(L(B)) \approx \sqrt{\frac{m+n+1}{2\pi e}}(q^m)^{\frac{1}{m+n+1}} \tag{37}$$

And so the ratio of the expected size of the shortest vector, to the actual size, is approximately

$$\frac{q^{\frac{m}{m+n+1}}}{\sigma} \approx \frac{\sqrt{q}}{\sigma} \text{ for normal form} \tag{38}$$

2.5 Dual Attacks

2.5.1 Tall decisional LWE to SIS

We will attack tall decisional-LWE by finding a short vector v such that $v^T A \equiv 0 \mod q$. To see how this helps, suppose we're given (A,b) as a D-LWE problem. Once we find the short v, we compute $v^T b \mod q$. There are two cases:

- If b is an LWE sample, then

$$v^T b = v^T(As + e) = (v^T A)s + v^T e \equiv v^T e \mod q \tag{39}$$

  Here we know both v and e are short, so by Cauchy-Schwarz, this inner product is also short. Thus, $v^T b \mod q$ will always be short if b is an LWE sample.

- If b is uniformly random, then $v^T b$ is also uniformly random, which is only short with a small probability.

Thus, we have a distinguisher. It's not a great distinguisher because $v^T b$ might just be small anyway. If b is LWE, it's very unlikely to have $v^T b$ large, so the chance of false negatives is extremely low, but false positives are much higher. Most lattice solvers will actually give us multiple linearly independent short v, and so we get a few samples to try.

Finding such short v ends up being the same as the SIS problem:

Problem 2.1 (Short Integer Solutions (SIS))**.** *Given an integer matrix A, a modulus q, and a bound β, find a non-zero integer vector x such that $\|x\| \leq \beta$ and $Ax \equiv 0 \mod q$.*

Solving SIS with A^T gives us exactly the short vector v that we need to attack D-LWE.

2.5.2 SIS to SVP

We can construct a lattice to solve SIS:

$$L_{SIS}(A) := \{v \in \mathbb{Z}^m \mid Av = 0 \mod q\} \tag{40}$$

We can see (why?) that this is a lattice, and so a short vector in this lattice solves SIS directly.

Why is this called a dual attack? First we define the dual lattice:

Definition 2.5. *The dual of a lattice $L \subseteq \mathbb{R}^n$ is*

$$L^\vee := \{v \in \textit{real span of } L \mid \langle v, w\rangle \subseteq \mathbb{Z}, \forall w \in L\} \tag{41}$$

If L has a full-rank basis then the real span of L is just $\mathbb{R}^n$. In other cases (say, if L has a basis that is taller than it is wide), we restrict to the span of these vectors. A key fact is that the dual of an integer lattice will often contain non-integer entries (as an exercise, what's the dual of $2\mathbb{Z}^n$?).

Recall that our lattice to reduce LWE to BDD was generated as

$$L_{BDD}(A) = \{v \in \mathbb{R}^m : \exists x, Ax \equiv v \mod q\} \tag{42}$$

The dual of this is not quite the same as $L_{SIS}(A)$. In fact, we want $qL_{BDD}(A)^\vee$.

Proposition 2.4. $L_{SIS}(A^T) = qL_{BDD}(A)^\vee$.

(note: $qL_{BDD}(A)^\vee \neq (qL_{BDD}(A))^\vee$; we take the dual, *then* multiply everything by q).

Proof. If $v \in qL_{BDD}(A)^\vee$, then $v = qv'$ for $v' \in L_{BDD}(A)^\vee$, and so $\langle v', w\rangle \in \mathbb{Z}$ for all $w \in L_{BDD}(A)$. This means $\langle qv', w\rangle \in q\mathbb{Z}$, or $\langle qv', w\rangle \equiv 0 \mod q$. But $w = Ax \mod q$, for some x, so we have that $\langle qv', Ax\rangle \equiv 0 \mod q$, or $(qv')^T Ax = v^T Ax \equiv 0 \mod q$. Since this holds for all $x \in \mathbb{Z}_q^n$, we can conclude that $v^T A \equiv 0 \mod q$.

This is almost exactly the requirement that $v \in L_{SIS}(A^T)$, except we have not yet shown that $v \in \mathbb{Z}^m$. We know that $q\mathbb{Z}^m \subseteq L_{BDD}(A)$. Thus, suppose there is some component i of v' which is a fraction $\frac{r}{s}$. Then since $v' \in L_{BDD}(A)^\vee$, we must have that $\langle v', qe_i\rangle = \frac{qr}{s} \in \mathbb{Z}$. This means $v_i = qv_i' \in \mathbb{Z}$, for any i; thus, $v = qv' \in \mathbb{Z}^m$.

For the reverse direction, if $v \in L_{SIS}(A^T)$, then we can consider $v' = \frac{1}{q}v \in \mathbb{Q}^m$, and take $\langle v', w\rangle$ for any $w \in L_{BDD}(A)$. We know that $w \equiv Ax \mod q$, or $w = Ax + qw'$ for some w'. We also know that $v^T A \equiv 0 \mod q$, or $v^T A = qv''$ for some v''. Putting all this together:

$$\begin{aligned}
\langle v', w\rangle =& \frac{1}{q}\langle v, w\rangle && (43)\\
=& \frac{1}{q}\langle v, Ax + qw'\rangle && (44)\\
=& \frac{1}{q}v^T Ax + v^T w' && (45)\\
=& \frac{1}{q}(qv'')^T x + v^T w' && (46)\\
=& v''^T x + v^T w' \in \mathbb{Z} && (47)
\end{aligned}$$

Thus, $v' \in L_{BDD}(A)^\vee$. □

We then take a theorem:

Theorem 2.2. *If B is a basis of a lattice L, then $D = B(B^TB)^{-1}$ is a basis of the dual $L^\vee$.*

Proof. Suppose $v \in L(D)$, i.e., $v = Dx$ for some $x \in \mathbb{Z}^n$. Then for any $w \in L$, $v^Tw = xD^Tw = x^T(B^TB)^{-1,T}B^Tw$. Since :

- $w = By$ for some integer vector y,
- the inverse of the transpose is the transpose of the inverse;
- $(B^TB)^T = B^TB$

then we get that

$$v^Tw = x^T(B^TB)^{-1}(B^TBy) = x^Ty \in \mathbb{Z} \tag{48}$$

Thus, $L(D) \subseteq L^\vee$.

Then let $v \in L^\vee$. We can see that $B^Tv \in \mathbb{Z}^n$ by definition. Thus, we can write:

$$\begin{aligned} v =& BB^{-1}(B^T)^{-1}B^Tv && (49)\\ =& B(B^TB)^{-1}B^Tv && (50)\\ =& DB^Tv && (51)\\ \in& D(\mathbb{Z}^n) && (52)\\ \in& L(D) && (53) \end{aligned}$$

□

Now, we want to get the shortest vector in the dual. We could laboriously compute $D = B(B^TB)^{-1}$, but what's a better way?

Given our expression, we can say:

$$\mathrm{Vol}(L^\vee) = |\det(D)| = \left|\frac{\det(B)}{\det(B)^2}\right| = \frac{1}{|\det(B)|} \tag{54}$$

This allows us to easily compute

$$\mathrm{Vol}(L_{SIS}(A^T)) = \mathrm{Vol}(qL_{BDD}(A)^\vee) = q^m\mathrm{Vol}(L_{BDD}(A)^\vee) = q^m\frac{1}{q^{m-n}} = q^n. \tag{55}$$

The dimension of $L_{SIS}(A^T)$ is also m. This means by the Gaussian heuristic, we expect our shortest vector to be

$$\sqrt{\frac{m}{2\pi e}} q^{\frac{n}{m}} \tag{56}$$

Quick sanity check: If m gets larger, then A is much taller. Thinking about $v^T A \equiv 0 \mod q$, we have more choices of values of v to possibly make this work, so we would expect smaller v. Indeed, that's what the Gaussian heuristic predicts.

I'm guessing at where to go from here, but I see no reason to expect a shorter vector. This means that if we solve γ-SVP for $L_{SIS}(A^T)$, we can expect the size of $v^T b \mod q$, for an LWE sample, to be about

$$\gamma \lambda_1(L_{SIS}(A^T))\|e\| \lessapprox \gamma m q^{\frac{n}{m}} \sigma. \tag{57}$$

We want this to be noticeably less than $q/2$, the expected value of $v^T b \mod q$ for uniformly random b. Rearranging we get

$$\gamma \lessapprox \frac{q^{1-\frac{n}{m}}}{m\sigma} \tag{58}$$

Were it not for the factor of m, this is basically the same as the primal attack. Perhaps there is some good reason to remove the factor of m, but I don't know it.

2.5.3 Normal LWE to SVP

The above only works for tall LWE. In fact if we define $L_{SIS}(A^T)$ for full-rank $n \times n$ A, the lattice is just $q\mathbb{Z}^n$, which is useless to us. Thus, we instead define:

$$L^{\vee}_{normal}(A) = \left\{(x, y) \in \mathbb{Z}^{m+n} \mid A^T x \equiv y^T \mod q\right\} \tag{59}$$

The shortest vector in this will solve D-LWE as well, with the same technique: Given a short vector (v, w), we just compute $v^T b$. This is still uniformly random for uniformly random b, but for LWE:

$$\begin{aligned} v^T b \equiv& v^T(As + e) && (60)\\ \equiv& (v^T A)s + v^T e && (61)\\ \equiv& w^T s + v^T e \mod q && (62) \end{aligned}$$

and both of these are small as well, since all of v, w, s, e are small.

We already analyzed a similar lattice to this, and we found it has a basis of

$$\begin{pmatrix} I_m & 0 \\ A^T & q_n \end{pmatrix} \tag{63}$$

which has determinant q^n and dimension $m+n$, and so we could apply the same Gaussian heuristic to find an approximation factor, but I will leave this out.

2.6 Lattice Basis Reduction

Having reduced LWE to approximate SVP problems, we want to approximately solve SVP. To do this we will use the Block-Korkine-Zolotarev (BKZ) algorithm, but we will warm up with 2 problems first. This section follows Galbraith's lattice chapter very closely.

2.6.1 Two-Dimensional Lattices

Suppose we have a two-dimensional lattice L, with a basis $B = \{b_1, b_2\}$. It's easy to find a shorter basis as follows: set $b_1' = b_1 - kb_2$, where k is an integer such that $\|b_1'\|$ is minimized. This is dead easy to find, since we know that $\|b_1 - kb_2\|^2 = \langle b_1 - kb_2, b_1 - kb_2\rangle = \|b_1\|^2 - 2k\langle b_1, b_2\rangle + k^2\|b_2\|^2$. This is easy to optimize (if k were continuous), at

$$k = \frac{\langle b_1, b_2\rangle}{\|b_2\|^2}. \tag{64}$$

The norm of b_1' is symmetric about this minimum, so we can simply round the value above to get the minimum integer value

This gives us the Lagrange-Gauss basis reduction technique: while $k \neq 0$, repeat the above process, swapping b_1 and b_2 after each iteration.

You can argue that this must terminate, and when it does, we have the shortest possible basis.

Proposition 2.5. *When the Lagrange-Gauss basis reduction terminates, we have the shortest possible basis: for any $v \in L$, $\|v\| \geq \max\{\|b_1\|, \|b_2\|\}$.*

Proof. Assume without loss of generality that $\|b_2\| \leq \|b_1\|$. Let $v \in L$, so that $v = a_1b_1 + a_2b_2$ for non-zero integers a_1 and a_2 (otherwise we just have

the basis vectors!). We can assume a_1 and a_2 are co-prime (exercise: why?), and so we can just divide a_2 by a_1 and write $a_2 = qa_1 + r$ for $1 \leq r < q$ ($r \geq 1$ because they are co-prime).

Then we just fiddle a bit with the arithmetic:

$$v = a_1 b_1 + (qa_1 + r)b_2 = a_1(b_1 + qb_1) + rb_2 \tag{65}$$

and use the reverse triangle inequality:

$$\|v\| \geq |a_1|\|b_1 + qb_2\| - r\|b_2\| \tag{66}$$

and then we just pop a factor of r out of the first term:

$$\|v\| \geq (|a_1| - r)\underbrace{\|b_1 + qb_2\|}_{(A)} + r(\underbrace{\|b_1 + qb_2\| - \|b_2\|}_{(B)}) \tag{67}$$

But we know that $\|b_1 + qb_2\| \geq \|b_1\|$, or else we would not have terminated our reduction, and this means that (B) is non-negative. Since we also assumed $\|b_1\| \geq \|b_2\|$, this means that (A) is at least as large as $\|b_1\|$. Finally, we know that $|a_1| - r \geq 1$, by definition of the remainder. This gives

$$\|v\| \geq \|b_1\| \tag{68}$$

□

Two key ideas we will take from this simple case:

- the choice of the value of k to minimize the norm;
- the notion of swapping and reducing to make shorter vectors.

We could attempt to generalize this algorithm to an n-dimensional lattice by simply reducing all pairs of lattice vectors in this way. But there's no guarantee this terminates in polynomial time, and I do not see a proof that even if it terminates that we will have a short basis.

2.6.2 Gram-Schmidt Orthogonalization

We will recall the Gram-Schmidt orthogonalization from intro linear algebra. The procedure works as follows:

1. For i from 1 to n:

(a) Set $b_i^* = b_i$

(b) For j from 1 to $i-1$:

i. Set $b_j^* = b_j^* - \frac{\langle b_j^*, b_i^* \rangle}{\langle b_i^*, b_i^* \rangle} b_i^*$

A few fun facts about this:

1. The Gram-Schmidt (GS) basis B^* is deterministically created from the original basis B. Thus, an *ordered* lattice basis B defines a GS basis, so we will treat the properties of the GS basis as properties of the lattice basis B.

2. Given a basis B, this gives us a basis $B^* = VB$ which is orthogonal. Notice that we did not normalize! We only want an orthogonal basis; the lengths of the basis are important information. It will also be more convenient to write $B = B^*U$, and then U is an upper triangular matrix with 1s on the diagonal.

3. We can also write
$$b_i^* = b_i - \sum_{j=1}^{i-1} \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} b_j^* \tag{69}$$

4. The coefficients in that last equation are quite important, so we give them their own notation:
$$\mu_{ij} := \frac{\langle b_j, b_i^* \rangle}{\langle b_i^*, b_i^* \rangle}. \tag{70}$$
These will be the upper elements in the matrix U such that $B = B^*U$.

5. The very first GS vector b_1^* wasn't modified, so $b_1^* = b_1$ is a lattice vector.

6. We have that
$$\langle b_i^*, b_i \rangle = \|b_i^*\|^2 \tag{71}$$
which one can see readily from the equation in item 3.

7. Since $B = B^*U$, we have $\det(B) = \det(B^*)\det(U)$, but $\det(U) = 1$ (it is upper-triangular with 1s on the diagonal, so $|\det(B^*)| = |\det(B)| =$

$\mathrm{Vol}(L)$. You can use the vectors of B^* to diagonalize it, and the diagonal will be the norms of each vector, and so $|\det(B^*)| = \prod_{i=1}^{n} \|b_i^*\|$. This leads to a very important equation:

$$\mathrm{Vol}(L) = \prod_{i=1}^{n} \|b_i^*\|. \tag{72}$$

8. It will also be convenient to define projection operators π_i which project a vector away from the space spanned by $\{b_1, \ldots, b_{i-1}\}$. We can give an explicit formula:

$$\pi_i(v) = v - \sum_{j=1}^{i-1} \frac{\langle v, b_j^* \rangle}{\|b_j^*\|} b_j^* \tag{73}$$

and this immediately tells us that $b_i^* = \pi_i(b_i)$.

2.6.3 The LLL Algorithm

Size-Reduction Notice that μ_{12} is exactly what we computed to solve SVP in two dimensions. Moreoever, the algorithm terminates when we the minimum of $\|b_1 + kb_2\|$ is at $k = 0$, which would mean $\lfloor \mu_{12} \rceil = 0$, or $|\mu_{12}| \leq \frac{1}{2}$. We can make that more general:

Definition 2.6 (Size-Reduced)**.** *A basis B is size-reduced if $|\mu_{ij}| \leq \frac{1}{2}$ for all $i \neq j$.*

Basically, we're generalizing our two-dimensional SVP solver: for each pair i, j, we set $b_j \leftarrow b_j + kb_i$ to minimize its norm, but only for $j > i$. After this we could swap vectors and try again, but that raises a tricky question about which vectors to swap. We will solve that later, but for now let's satisfy the size-reduced condition.

Lemma 2.3. *Given a basis B, construct B' from B by setting $b'_j = b_j - \lfloor \mu_{ij} \rceil b_i$ for some i, j with $i < j$. Then $|\mu'_{ij}| \leq \frac{1}{2}$ and $\mu'_{k\ell} = \mu_{k\ell}$ for all $(k, \ell) \neq (i, j)$ unless $\ell = j$ and $k < i$.*

Proof. First notice that $b_k^{*'} = b_k^*$ for all $k \neq j$: this is clearly true for $k < j$ because none of the first $j - 1$ vectors change. For $k > j$, we can see that the span of the first $k - 1$ vectors is the same in B', since we only changed

b_j by added a basis vector already in this span, so projecting away from this span (which is what the GS orthogonalization does) will not change.

We also see that $b_j^{*'} = b_j^*$, since we project b'_j by the span of the first $j-1$ vectors, which projects away the new b_i direction added.

Then recall that

$$\mu_{ij} = \frac{\langle b_j, b_i* \rangle}{\|b_i^*\|^2} \tag{74}$$

Because b_k^* and b_ℓ are unchanged for $k, \ell \neq j$, the only $\mu_{k\ell}$ which might be different are with $k = j$ or $\ell = j$.

We can already see that $\mu'_{jk} = \mu_{jk}$ because $b_j^{*'} = b_j^*$.

Then consider μ'_{kj} for $k > i$. We know that b_k^* is orthogonal to b_i, so $\langle b'_j, b_k^* \rangle = \langle b_j, b_k^* \rangle$ and thus $\mu'_{kj} = \mu_{kj}$.

Finally,

$$\mu'_{ij} = \frac{\langle b_j - \lfloor \mu_{ij} \rceil \, b_i, b_i* \rangle}{\|b_i^*\|^2} \tag{75}$$

$$= \frac{\langle b_j, b_i* \rangle}{\|b_i^*\|^2} - \lfloor \mu_{ij} \rceil \underbrace{\frac{\langle b_i, b_i* \rangle}{\|b_i^*\|^2}}_{=1 \text{ (by GS fact 6)}} \tag{76}$$

$$= \mu_{ij} - \lfloor \mu_{ij} \rceil \tag{77}$$

which must be in $[-1/2, 1/2]$. □

This Lemma gives us an immediate algorithm to size-reduce a basis:

1. For $j = n$ down to 1:
   (a) For $i = j - 1$ down to 1:
      i. Set $b_j = b_j - \lfloor \mu_{ij} \rceil \, b_i$

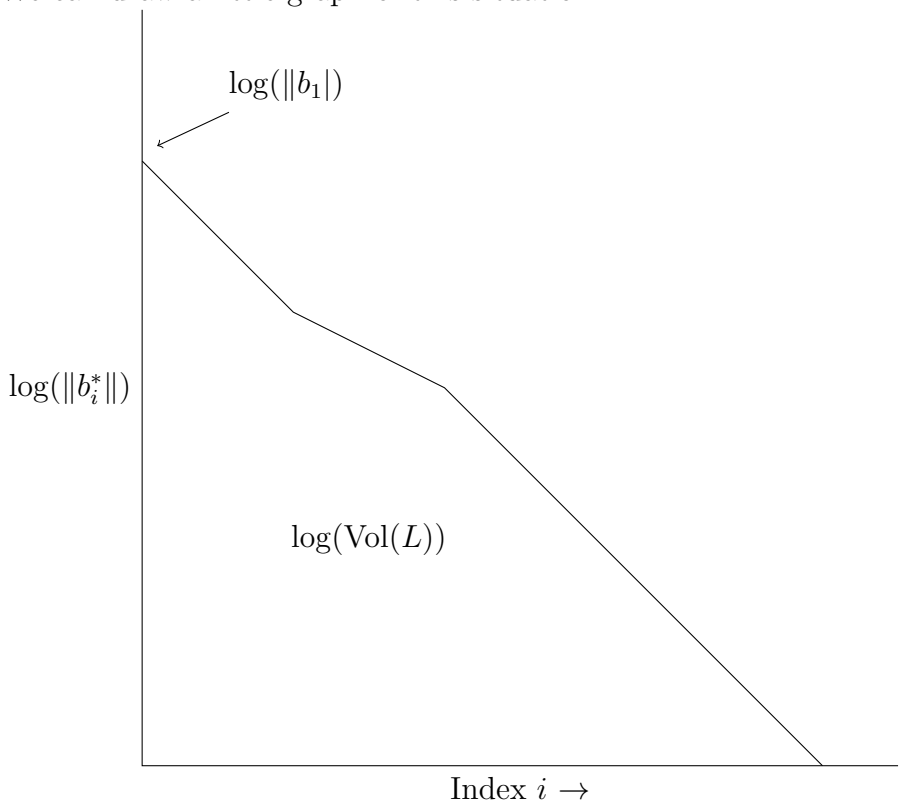
which is only $O(n^2)$ time.

The Lovasz Condition GS Fact 7 is quite interesting because the product of norms of the GS vectors is an invariant of the lattice. But since $b_1^* = b_1$ is in the lattice, that means if we *increase* the norms of all other GS vectors, then the norm of b_1 must *decrease*. This will be the secret of all lattice basis reduction.

If we try orthogonalizing some random vectors, what you'll find is that the later vectors get quite a bit smaller. As intuition, the first $n - 1$ vectors

will span an $n-1$-dimensional subspace of $\mathbb{R}^n$, so choosing a random vector for b_n will have only very little of the vector in the last, unspanned dimension; thus, once we project away from the first $n-1$ vectors, there won't be much left.

We can draw a little graph of this situation:



Since we've taken the log of the norms, then the area under this curve is the sum of the logs, which is the log of the product, which is precisely the log of Vol(L)! That is, if we plot any basis like this, the area under this curve is conserved. This means if the basis slopes down quickly, it means $\log(\|b_1|)$ (the first vector) must be large, and conversely, if it slopes down slowly, we have a good basis and $\|b_1\|$ will be smaller.

We can quantify this a bit more precisely: if we want the slope above some value a, this means that

$$\log(\|b_i^*\|) \leq \log(\|b_{i+1}^*\|) + a \tag{78}$$

or,

$$\|b_i^*\| \leq \alpha \|b_{i+1}^*\| \tag{79}$$

for some α. This is often expressed as:

Definition 2.7 (Lovasz Condition)**.** *A lattice basis satisfies the Lovasz condition for $\delta \in [0, 1)$ if*

$$\delta \|b_i^*\| \leq \|b_{i+1}^*\|^2 + |\mu_{i,i+1}|^2 \|b_i^*\|^2 \tag{80}$$

for all i from 1 to $n-1$.

(other definitions define this only for a specific index i).

This doesn't quite fit the above, but if you recall that for a size-reduced basis we have $|\mu_{i,i+1}|^2 \leq \frac{1}{4}$, we can rearrange the equation to get $\|b_i^*\| \leq \alpha \|b_{i+1}^*\|^2$ for $\alpha = \frac{1}{\delta - \frac{1}{4}}$

If we satisfy this property, then we can chain these inequalities together to get

$$\|b_1\| = \|b_1^*\| \leq \alpha \|b_2^*\| \leq \cdots \leq \alpha^{n-1} \|b_n^*\| \tag{81}$$

Here we use the fact that $\|b_n^*\| \leq \lambda_1(L)$. Thus,

$$\|b_1\| \leq \alpha^{n-1} \lambda_1(L). \tag{82}$$

Great! We've solved α^{n-1}-SVP. This looks exponentially bad, and it is, but that's still better than nothing.

How do we ensure we satisfy this condition? Recall the projection operations (GS Fact 8):

Lemma 2.4. *If a size-reduced basis B satisfies $\|\pi_i(b_i)\| \leq \|\pi_i(b_{i+1})\|$, then it satisfies the Lovasz condition at index i for any $\alpha \geq \sqrt{\frac{4}{3}}$.*

Proof. Recall that $\pi_i(b_i) = b_i^*$, though $\pi_i(b_{i+1}) \neq b_{i+1}^*$. Instead,

$$b_{i+1}^* = \pi_i(b_{i+1}) - \mu_{i,i+1} b_i^* \tag{83}$$

(that is, project away from the first $i-1$ basis vectors, then the ith basis vector).

We then take the inner product with $\pi_i(b_{i+1})$. The same logic as GS Fact 6 shows that

$$\langle b_{i+1}^*, \pi_i(b_{i+1}) \rangle = \|b_{i+1}^*\|^2 \tag{84}$$

(or by rearranging Equation 83 and taking the inner product with b_{i+1}^*). And we also have that

$$\langle b_i^*, \pi_i(b_{i+1}) \rangle = \langle b_i^*, b_{i+1} \rangle = \mu_{i,i+1} \|b_i^*\|^2 \tag{85}$$

by similar reasoning. This means

$$\begin{aligned}
\|b_{i+1}^*|^2 =& \|\pi_i(b_{i+1})\|^2 - \mu_{i,i+1}^2 \|b_i^*\|^2 && (86)\\
\geq& \|b_i^*\|^2 - \mu_{i,i+1}^2 \|b_i^*\|^2 && (87)\\
=& \|b_i^*\|^2 (1 - \mu_{i,i+1}^2) && (88)\\
\geq& \|b_i^*\|^2 \frac{3}{4} && (89)
\end{aligned}$$

with the last step because the basis is size-reduced. □

This last lemma suggests that once we size-reduce, if we find an index that does not satisfy the Lovasz condition, we should just swap it with its successor, and then necessarily it *will* satisfy the Lovasz condition: since this will not change π_i, then $\|\pi_i(b_i)\| \leq \|\pi_i(b_{i+1})\|$ in the new basis.

This works, but not quite for that reason. Unfortunately, we might have spoiled the Lovasz condition between i and $i-1$, or $i+1$ and $i+2$. The full proof that LLL works is more complicated, but the above just gave some intuition.

Here is LLL, more or less:

1. While the Lovasz condition does not hold:
   (a) Size-Reduce B
   (b) Swap b_i and b_{i+1} at the first index i that does not satisfy the Lovasz condition.

Of course we can be much more efficient and do the same things (if we just swap b_i and b_{i+1}, there's actually not much we need to change to keep it size-reduced), but this is sufficient.

Theorem 2.3. *LLL runs in time polynomial in n for $\delta \leq \frac{3}{4}$ (equivalently, $\alpha \geq \sqrt{2}$).*

Again, the proof is actually quite complicated.

Theorem 2.4. *LLL solves $2^{\frac{n-1}{2}}$-SVP in polynomial time.*

Proof. As we saw before, if the Lovasz condition holds, then $\|b_1\| \leq \alpha^{n-1}\lambda_1(L)$, and since we can do $\alpha = \sqrt{2}$, the result follows. □

Again, LLL's approximation factor seems kind of trivial, but it is suprisingly insightful and powerful. It is enough for Coppersmith's lattice-based attacks on RSA, for example.

2.6.4 The BKZ Algorithm

If you consider LLL, between the size-reduction and swapping, we're basically solving a 2-dimensional shortest vector problem for b_i and b_{i+1}. Not exactly: we're solving it for $\pi_i(b_i)$ and $\pi_i(b_{i+1})$. BKZ asks: what if we did more than a 2-dimensional shortest vector problem? That is, what if we had an oracle that could solve β-dimensional SVP *exactly*, could we use it solve approximate SVP for a better approximation factor?

That's what it does, more or less. We will define two things to make it work:

$$L_i := L(\{\pi_i(b_i), \pi_i(b_{i+1}), \ldots, \pi_i(b_{i+\beta-1})\}) \tag{90}$$

That is, L_i is the lattice generated by the projections of the next β vectors after $i-1$.

Given any vector $v \in L_i$, we can lift it to a vector in L as follows: We know that

$$v = \sum_{j=i}^{i+\beta-1} a_i \pi_i(b_j) \tag{91}$$

for $a_i \in \mathbb{Z}$, so we can just make

$$\text{lift}(v) = \sum_{j=i}^{i+\beta-1} a_i b_j. \tag{92}$$

By linearity of the projection operator, we have that $\pi_i(\text{lift}(v) = v$.

This gives us BKZ. In short, we solve SVP exactly in blocks of size β, and lift the results into the original basis.

1. Repeat:
   (a) For $i = 1$ to n:
      i. Find b_i', the lift of a shortest vector in L_i
      ii. If $b_i' \neq b_i$, then:
         A. Add the lift of the shortest vector in L_i to B
         B. Use LLL to reduce B from $n+1$ to n vectors again
         C. Restart the main loop

A nice fact about LLL is that if you give it extra vectors (so, more than a basis of the lattice) it will output a basis. I have no idea how to prove this fact.

I also do not know how to prove that BKZ terminates, though I think this is more of an area of active research. Heuristically, it takes $O(\frac{n^2}{\beta^2})$ loops.

What we can prove is that it produces a reasonably good approximation. Recall that $\pi_i(b_i) = b_i^*$. When BKZ terminates, we know that b_i is the lift of a shortest vector in L_i, which means that $b_i^* = \pi_i(b_i)$ *is* a shortest vector in L_i. This means that

$$\|b_i^*\| \leq \gamma_\beta \text{Vol}(L_i)^{1/\beta} = \gamma_\beta \left(\prod_{j=i}^{i+\beta-1} \|b_j^*\| \right)^{1/\beta} \tag{93}$$

where the last equation uses the fact that orthogonalizing $\{\pi_i(b_i), \ldots, \pi_i(b_{i+\beta-1})\}$ will just give you the original GS vectors b_i^* to $b_{i+\beta-1}^*$.

This tells us that we have a bound for $\|b_i^*\|$ which almost the geometric average of the next β vectors. As β gets larger, this will make it a tighter bound.

We need a quick lemma for this theorem. It's not that important but it can give some flavour of lattice geometry.

Lemma 2.5. *The Hermite constants γ_n satisfy $\gamma_n^n \leq \gamma_m^m$ for $n \leq m$.*

Proof. The Hermite constants are defined to be the minimum values γ_n such that for any lattice L in n dimensions,

$$\lambda_1(L) \leq \gamma_n \text{Vol}(L)^{1/n}. \tag{94}$$

Thus, there should be some lattice L such that there the shortest vector $v \in L$ satisfies

$$v = \gamma_n \text{Vol}(L)^{1/n}. \tag{95}$$

Let B be a basis of L. We construct a new lattice L' with basis

$$B' = \begin{pmatrix} B & 0 \\ 0 & \gamma_n \text{Vol}(L)^{1/n} \end{pmatrix} \tag{96}$$

L' has v as its shortest vector, or $(0, \gamma_n \text{Vol}(L)^{1/n})$ (any other vector would imply a shorter vector in L). However,

$$\text{Vol}(L') = |\det(B)| \gamma_n \text{Vol}(L)^{1/n} = \text{Vol}(L)^{\frac{n+1}{n}} \gamma_n \tag{97}$$

By definition of γ_{n+1}, we have that

$$\|v\| = \lambda_1(L') \leq \gamma_{n+1} \text{Vol}(L')^{\frac{1}{n+1}} \tag{98}$$

which we can rearrange to give

$$\gamma_n \mathrm{Vol}(L)^{\frac{1}{n}} \leq \gamma_{n+1} \mathrm{Vol}(L)^{\frac{1}{n}} \gamma_n^{\frac{1}{n+1}} \tag{99}$$

and this gives the result for $m = n + 1$. Induction finishes the result. □

Theorem 2.5. *When BKZ terminates (i.e., b_i is the lift of a shortest vector in L_i for all $i \in [1, n - \beta + 1]$), the first vector b_1 satisfies*

$$\|b_1\| \leq \gamma_\beta^{\frac{n-1}{\beta-1}} \lambda_1(L). \tag{100}$$

Proof. The bound we ust saw is easier to work with in this form:

$$\|b_i^*\|^\beta \leq \gamma_\beta^\beta \prod_{j=i}^{i+\beta-1} \|b_j^*\|; \tag{101}$$

Suppose we compute

$$\prod_{i=1}^{n} \|b_i^*\|^\beta \tag{102}$$

then our bound above gives us

$$\prod_{i=1}^{n} \|b_i^*\|^\beta \leq \gamma_\beta^{(n-\beta+1)\beta} \left(\prod_{i=1}^{\beta-1} \|b_i^*\|^i \right) \left(\prod_{i=\beta}^{n-\beta} \|b_i^*\|^\beta \right) \left(\prod_{i=n-\beta}^{n} \|b_i^*\|^{n+1-i} \right) \tag{103}$$

This is sort of like a convolution: it's like we're multiplying together all these geometric averages, so they flatten out in the middle but trail off at the edges.

If we cancel out a bunch of terms in the equation above, we're left with

$$\prod_{i=1}^{k} \|b_i^*\|^{\beta-i} \leq \gamma_\beta^{\frac{(n-\beta-1)\beta}{2}} \prod_{i=n-\beta}^{n} \|b_i^*\|^{n+1-i} \tag{104}$$

We want to bound b_1^*.

Notice that if we BKZ-reduced the first block for β, we also BKZ-reduced it for $\beta - 1$, $\beta - 2$, down to 2. Normally removing basis vectors can make the shortest vector larger, but in this case the shortest vector is already in the basis ($\pi_i(b_i)$ is the shortest vector), so it will still be shortest if we have fewer other basis vectors.

Thus, we also have that

$$\|b_1^*\|^i \leq \sqrt{\gamma_i}^i \prod_{j=1}^{i} \|b_j^*\| \tag{105}$$

From Lemma 2.5, $\gamma_i^i \leq \gamma_\beta^\beta$ if $i \leq \beta$, and then we can add in more bounds for the first β terms, i.e.,

$$\prod_{i=1}^{k} \|b_i\|^i \leq \prod_{i=1}^{\beta} \sqrt{\gamma_i^i} \prod_{j=1}^{i} \|b_j^*\| \tag{106}$$

$$\|b_i\|^{\frac{\beta(\beta-1)}{2}} \leq \gamma_\beta^{\frac{\beta(\beta-1)}{2}} \prod_{j=1}^{\beta} \|b_j^*\|^{\beta-i} \tag{107}$$

The right-hand side is basically just the left-hand side of Equation 104, so we can substitute and combine these to give

$$\|b_1\|^{\frac{\beta(\beta-1)}{2}} \gamma_\beta^{-\frac{\beta(\beta-1)}{2}} \leq \gamma_k^{\frac{(n-\beta+1)\beta}{2}} \prod_{i=n-\beta}^{n} \|b_i^*\|^{n+1-i} \tag{108}$$

and we can move all γ_β terms to the right:

$$\|b_1\|^{\frac{\beta(\beta-1)}{2}} \leq \gamma_\beta^{\frac{(n-1)\beta}{2}} \prod_{i=n-\beta}^{n} \|b_i^*\|^{n+1-i} \tag{109}$$

That last piece is annoying. Let's upper bound it by $\|b_{max}^*\| := \max_{i=n-\beta}^{n} \{\|b_i^*\|\}$. This gives us

$$\|b_1\|^{\frac{\beta(\beta-1)}{2}} \leq \gamma_\beta^{\frac{(n-1)\beta}{2}} \|b_{max}^*\|^{\frac{\beta(\beta-1)}{2}} \tag{110}$$

or, taking $\beta(\beta-1)/2$th roots,

$$\|b_1\| \leq \gamma_\beta^{\frac{n-1}{\beta-1}} \|b_{max}\| \tag{111}$$

Our final step is to compare to λ_1. We argue that $\|b_{max}\| \geq \lambda_1$. Let v be a shortest vector in the lattice. Generally, $\pi_i(v)$ won't be the shortest vector in L_i, because $\pi_i(v)$ will not actually be in the lattice L_i. The reason is because v likely has some non-zero components in vectors outside of L_i (i.e., basis vectors past $i + \beta - 1$). However, for $i \geq n - \beta + 1$, we actually do have

that $\pi_i(v) \in L_i$, because all the remaining GS basis vectors are in the lattice! Thus, since we ran BKZ, we are guaranteed that $b^*_{n-\beta}$ is the shortest vector in $L_{n-\beta}$, so $\|\pi_{n-\beta}(v)\| \geq \|b^*_{n-\beta}\|$, and we can assume $\|b^*_i\|$ decreases in i.

We also have, by definition of projections, that $\|\pi_{b-\beta}(v)\| \leq \|v\| = \lambda_1(L)$, which tells us

$$\|b_1\| \leq \gamma_\beta^{\frac{n-1}{\beta-1}} \|b_{max}\| \leq\leq \gamma_\beta^{\frac{n-1}{\beta-1}} \lambda_1(L). \tag{112}$$

□

This is a nice result because if we take $\beta = 2$, we recover (more or less) LLL. If we take $\beta = cn$ for $c < 1$, we get an approximation of

$$\gamma_{cn}^{\frac{n-1}{cn-1}} \leq \sqrt{cn}^{\frac{n-1}{cn-1}} \leq (cn)^{\frac{1}{c}} \tag{113}$$

which is polynomial in n. In fact we can get $O(n)$ approximation with $c = \frac{1}{2}$.

There are better analyses that give a tighter approximation factor, but many rely on heuristics and experimental findings.

Putting this together gives us a nice trade-off between the approximation factor we want and the difficulty, because if we have larger β, then the problem is definitely harder. We'll shortly see that the best known run-time for exact SVP is exponential in the dimension, so the runtime of BKZ is exponential in β. This gives us:

- $\beta = O(1)$, polynomial runtime but exponential approximation factor;
- β subexponential: subexponential runtime and subexponential approximation factor;
- $\beta = cn$: exponential runtime but polynomial approximation factor.

The proof doesn't give us a constant approximation factor at any parameter, but of course if we take $\beta = n$ then, by definition, we solve SVP exactly.