Traditionally, lattices were used as tools in cryptanalysis, that is, as tools in *breaking* cryptographic schemes. We have seen an example of such an application in a previous lecture. In 1996, Ajtai made a surprising discovery: lattices can be used to *construct* cryptographic schemes [1]. His seminal work sparked great interest in understanding the complexity of lattice problems and their relation to cryptography.

Ajtai's discovery is interesting for another reason: the security of his cryptographic scheme is based on the *worst-case hardness* of lattice problems. What this means is that if one succeeds in breaking the cryptographic scheme, even with some small probability, then one can also solve *any* instance of a certain lattice problem. This remarkable property is what makes lattice-based cryptographic construction so attractive. In contrast, virtually all other cryptographic constructions are based on some *average-case* assumptions. For example, in cryptographic constructions based on factoring, the assumption is that it is hard to factor numbers chosen from a certain distribution. But how should we choose this distribution? Obviously, we should not use numbers with small factors (such as even number), but perhaps there are other numbers that we should avoid? In cryptographic constructions based on worst-case hardness, such questions do not even arise.

Let us describe Ajtai's result more precisely. The cryptographic construction given in [1] is known as a *family of one-way functions*. Ajtai proved that the security of this family can be based on the worst-case hardness of the n^c-approximate SVP for some constant c. In other words, the ability to invert a function chosen from this family with non-negligible probability implies an ability to solve *any* instance of n^c-approximate SVP. Shortly after, Goldreich et al. [4] improved on Ajtai's result by constructing a stronger cryptographic primitive known as a family of *collision resistant hash functions (CRHF)*. Much of the subsequent work concentrated on decreasing the constant c (thereby improving the security assumption) [3, 5, 6]. In the most recent work, the constant is essentially $c = 1$.

Shortly after [1], on a different but related direction of research, Ajtai and Dwork [2] constructed a *public-key cryptosystem* whose security is based on the worst-case hardness of lattice problems. Several improvements were given in subsequent works [4, 7]. We should mention that unlike the case of one-way functions and CRHF, the security of all known lattice-based public-key cryptosystems is based on a special case of SVP known as unique-SVP. The hardness of this problem is not understood so well, and it is an open question whether one can base public-key cryptosystems on the (worst-case) hardness of SVP.

1 Our CRHF

In this lecture, we present a CRHF based on the worst-case hardness of $O(n^3)$-approximate SIVP. This construction is a somewhat simplified version of the one in [6]. We remark that it is possible to improve the security assumption to $\tilde{O}(n)$-approximate SIVP, as was done in [6]. We will indicate how this can be done in Section 4. Let us first recall the definition of SIVP.

DEFINITION 1 (SIVP_γ) *Given a basis $B \in \mathbb{Z}^{n \times n}$, find a set of n linearly independent vectors in $\mathcal{L}(B)$ each of length at most $\gamma \lambda_n(\mathcal{L}(B))$.*

The transference theorem of Banaszczyk, which we saw in the last lecture, shows that a solution to SIVP_γ implies a solution to (the optimization variant of) $\mathsf{SVP}_{\gamma \cdot n}$. This is achieved by simply solving SIVP_γ on the dual lattice. Therefore our CRHF construction is also based on the worst-case hardness of $O(n^4)$-approximate SVP. We now give the formal definition of a CRHF.

DEFINITION 2 *A family of collision resistant hash functions (CRHF) is a sequence $\{\mathcal{F}_n\}_{n=1}^{\infty}$, where each $\mathcal{F}_n$ is a family of functions $f : \{0,1\}^{m(n)} \to \{0,1\}^{k(n)}$, with the following properties.*

1. *There exists an algorithm that given any $n \geq 1$ outputs a random element of $\mathcal{F}_n$ in time polynomial in n.*

2. *Every function $f \in \mathcal{F}_n$ is efficiently computable.*

3. *For any $c > 0$, there is no polynomial-time algorithm that with probability at least $\frac{1}{n^c}$, given a random $f \in \mathcal{F}_n$ outputs x, y such that $x \neq y$ and $f(x) = f(y)$ (i.e., there is no polynomial-time algorithm that with non-negligible probability finds a collision).*

REMARK 1 We usually consider functions from $\{0,1\}^m$ to $\{0,1\}^k$ for $m > k$ so that collisions are guaranteed to exist. If no collisions exist, the last requirement is trivially void.

REMARK 2 The more standard notion of a *family of one-way functions* (OWF) is defined similarly, where instead of the last requirement we have the following:

3. *For any $c > 0$, there is no polynomial-time algorithm that with probability at least $\frac{1}{n^c}$, given a random $f \in \mathcal{F}_n$ and the value $f(x)$ for a random $x \in \{0,1\}^m$, outputs y such $f(x) = f(y)$ (i.e., there is no polynomial-time algorithm that with non-negligible probability inverts the function on a random input).*

It is easy to see that any CRHF is in particular a OWF. We remark that both are important primitives in cryptography, but we will not expand on this topic.

Our CRHF is essentially the modular subset-sum function over $\mathbb{Z}_q^n$, as defined next. It is parameterized by two functions $m = m(n), q = q(n)$.

DEFINITION 3 *For any $a_1, \ldots, a_m \in \mathbb{Z}_q^n$, define $f_{a_1,\ldots,a_m}$ as the function from $\{0,1\}^m$ to $\{0,1\}^{n \log q}$ given by*

$$f_{a_1,\ldots,a_m}(b_1, \ldots, b_m) = \sum_{i=1}^{m} b_i a_i \bmod q.$$

Then, we define the family $\mathcal{F}_n$ as the set of functions $f_{a_1,\ldots,a_m}$ for all $a_1, \ldots, a_m \in \mathbb{Z}_q^n$.

This family clearly satisfies the first two properties of a CRHF. Our main theorem below shows that for a certain choice of parameters, the existence of a "collision finder" (i.e., an algorithm that violates the third property of a CRHF) implies a solution to $\mathsf{SIVP}_{O(n^3)}$.

THEOREM 4 *Let $q = 2^{2n}$ and $m = 4n^2$. Assume that there exists a polynomial-time algorithm* COLLISIONFIND *that given random elements $a_1, \ldots, a_m \in \mathbb{Z}_q^n$ finds $b_1, \ldots, b_m \in \{-1, 0, 1\}$, not all zero, such that $\sum_{i=1}^m b_i a_i = 0 \pmod q$ with probability at least n^{-c_0} for some constant $c_0 > 0$. Then there is a polynomial-time algorithm that solves $\mathsf{SIVP}_{O(n^3)}$ on any lattice.*

Notice that for this choice of parameters, $m > n \log q$ so collisions are guaranteed to exist. The proof is based on the idea of smoothing a lattice by Gaussian noise, which is described in the next section.

2 The Smoothing Parameter

For $s > 0$ and $x \in \mathbb{R}^n$ define $\nu_s(x) = \rho_s(x)/s^n$. This is the Gaussian probability density function with parameter s. As we have seen in the last lecture, a vector chosen randomly according to ν_s has length at most $\sqrt{n}s$ with probability $1 - 2^{-\Omega(n)}$. In this section we are interested in understanding what happens when we take the 'uniform' distribution on a lattice and add Gaussian noise to it. An illustration of this is shown in Figure 1. The four plots show the distribution obtained with four different values of s. Notice that as we add more Gaussian noise, the distribution becomes closer to uniform. Our goal in this section is to

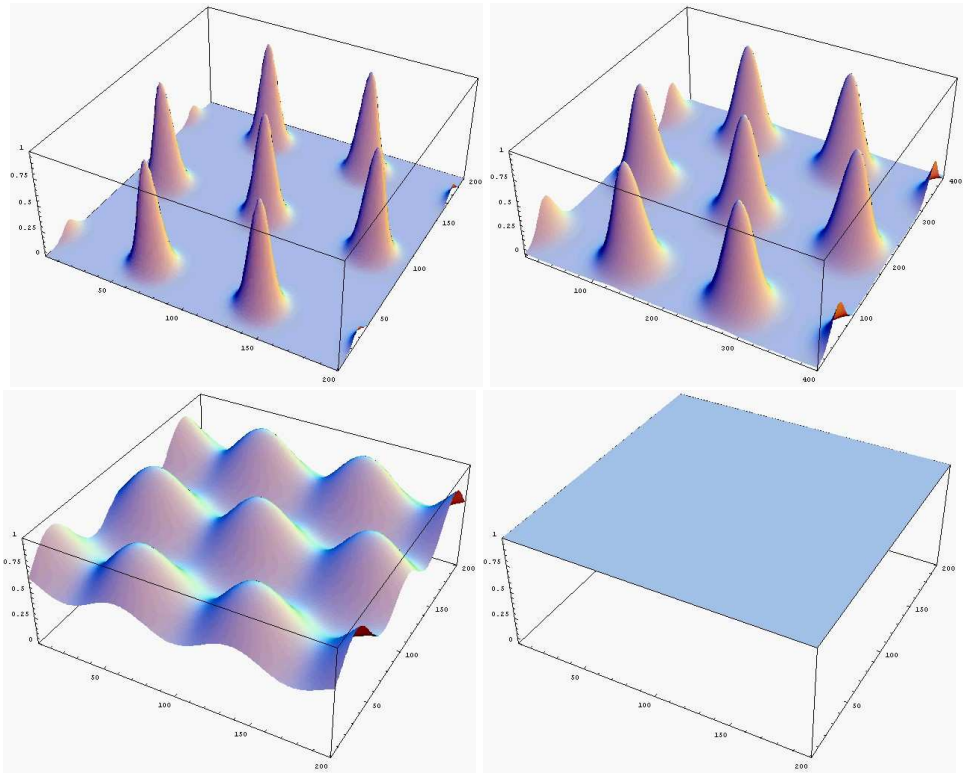

Figure 1: A lattice distribution with different amounts of Gaussian noise

analyze this formally and understand how large s has to be for this to happen. This will play a crucial role in the proof of the main theorem.

To make the above formal, we 'work modulo the parallelepiped', as was described in Lecture 7. Namely, the statement we wish to prove is that for large enough s, if we reduce the distribution ν_s modulo $\mathcal{P}(B)$, we obtain a distribution that is very close to uniform over $\mathcal{P}(B)$. This is done in the following lemma.

LEMMA 5 *Let Λ be a lattice with basis B. Then, the statistical distance between the uniform distribution on $\mathcal{P}(B)$ and the distribution obtained by sampling from ν_s and reducing the result modulo $\mathcal{P}(B)$ is at most $\frac{1}{2}\rho_{1/s}(\Lambda^* \setminus \{0\})$.*

PROOF: We need to calculate the statistical distance between the following two density functions on $\mathcal{P}(B)$:

$$U(x) = \frac{1}{\det(\Lambda)} = \det(\Lambda^*)$$

and

$$Y(x) = \sum_{x' \text{ s.t. } x' \bmod \mathcal{P}(B) = x} \nu_s(x') = \frac{1}{s^n}\rho_s(x + \Lambda).$$

Using the Poisson summation formula and properties of the Fourier transform, we obtain

$$\begin{aligned} Y(x) &= \frac{1}{s^n}\det(\Lambda^*) \cdot s^n \cdot \sum_{w \in \Lambda^*} \rho_{1/s}(w) \cdot e^{2\pi i \langle w, x \rangle} \\ &= \det(\Lambda^*)\left(1 + \sum_{w \in \Lambda^* \setminus \{0\}} \rho_{1/s}(w) \cdot e^{2\pi i \langle w, x \rangle}\right). \end{aligned}$$

So,

$$\begin{aligned}
\Delta(Y,U) &= \frac{1}{2}\int_{\mathcal{P}(B)} |Y(x)-U(x)|dx \\
&\le \frac{1}{2}\mathrm{vol}(\mathcal{P}(B)) \cdot \max_{x\in\mathcal{P}(B)} |Y(x)-\det(\Lambda^*)| \\
&= \frac{1}{2}\det(\Lambda)\cdot\det(\Lambda^*) \max_{x\in\mathcal{P}(B)} \left| \sum_{w\in\Lambda^*\setminus\{0\}} \rho_{1/s}(w)\cdot e^{2\pi i\langle w,x\rangle} \right| \\
&\le \frac{1}{2}\det(\Lambda)\cdot\det(\Lambda^*) \sum_{w\in\Lambda^*\setminus\{0\}} \left|\rho_{1/s}(w)\right| \\
&= \frac{1}{2}\rho_{1/s}(\Lambda^*\setminus\{0\})
\end{aligned}$$

where the last inequality uses the triangle inequality. □

The above lemma motivates the following definition.

DEFINITION 6 *For any $\varepsilon > 0$, we define the* smoothing parameter *of Λ with parameter ε as the smallest s such that $\rho_{1/s}(\Lambda^*\setminus\{0\}) \le \varepsilon$ and denote it by $\eta_\varepsilon(\Lambda)$.*

To see why this is well-defined, notice that $\rho_{1/s}(\Lambda^*\setminus\{0\})$ is a continuous and strictly decreasing function of s with $\lim_{s\to 0}\rho_{1/s}(\Lambda^*\setminus\{0\}) = \infty$ and $\lim_{s\to\infty}\rho_{1/s}(\Lambda^*\setminus\{0\}) = 0$. Using this definition, the lemma can be restated as follows: for any $s \ge \eta_\varepsilon(\Lambda)$, the statistical distance between the uniform distribution on $\mathcal{P}(B)$ and the distribution obtained by sampling from ν_s and reducing the result modulo $\mathcal{P}(B)$ is at most $\frac{1}{2}\varepsilon$. In the rest of this section, we relate $\eta_\varepsilon(\Lambda)$ to other lattice parameters.

CLAIM 7 *For any $\varepsilon < \frac{1}{100}$ we have $\eta_\varepsilon(\Lambda) \ge \frac{1}{\lambda_1(\Lambda^*)}$.*

PROOF: Let $s = \frac{1}{\lambda_1(\Lambda^*)}$, and let $y \in \Lambda^*$ be of norm $\lambda_1(\Lambda^*)$. Then

$$\rho_{1/s}(\Lambda^*\setminus\{0\}) \ge \rho_{1/s}(y) = e^{-\pi\|y/\lambda_1(\Lambda^*)\|^2} = e^{-\pi} > \frac{1}{100}.$$

□

Using Banaszczyk's transference theorem, we immediately obtain the following corollary.

COROLLARY 8 *For any $\varepsilon < \frac{1}{100}$ we have $\eta_\varepsilon(\Lambda) \ge \frac{1}{n}\lambda_n(\Lambda)$.*

CLAIM 9 *For any $\varepsilon \ge 2^{-n+1}$, $\eta_\varepsilon(\Lambda) \le \frac{\sqrt{n}}{\lambda_1(\Lambda^*)}$.*

PROOF: Let $s = \sqrt{n}/\lambda_1(\Lambda^*)$. Our goal is to prove that $\rho_{1/s}(\Lambda^*\setminus\{0\}) \le 2^{-n+1}$. Then,

$$\rho_{1/s}(\Lambda^*\setminus\{0\}) = \rho(s\Lambda^*\setminus\{0\}) \le 2^{-n+1}$$

where the inequality follows from a corollary we saw in the previous lecture together with $\lambda_1(s\Lambda^*) \ge \sqrt{n}$. □

Using the easy direction of the transference theorem, we obtain the following corollary.

COROLLARY 10 *For any $\varepsilon \ge 2^{-n+1}$, $\eta_\varepsilon(\Lambda) \le \sqrt{n}\lambda_n(\Lambda)$.*

We remark that it can be shown that $\eta_\varepsilon(\Lambda) \le \log n \cdot \lambda_n(\Lambda)$ for $\varepsilon \ge n^{-\log n}$ (see Lemma 15).

3 Proof of Theorem 4

Our goal is to describe an algorithm that solves $\mathsf{SIVP}_{O(n^3)}$ on any given lattice Λ using calls to COLLISION-FIND (as defined in Theorem 4). The core of the algorithm is the procedure FINDVECTOR presented below. In this procedure and elsewhere in this section, we fix ε to be $n^{-\log n}$ and recall that we choose $q = 2^{2n}$ and $m = 4n^2$. The output of FINDVECTOR is some random short lattice vector. As we shall see later, by calling FINDVECTOR enough times, we can obtain a set of n short linearly independent vectors, as required.

Roughly speaking, FINDVECTOR works as follows. It first chooses m vectors $x_1, \ldots, x_m$ independently from the Gaussian distribution $\nu_{\tilde{\eta}}$ where $\tilde{\eta}$ is close to the smoothing parameter of the lattice. Since the smoothing parameter is not much bigger than λ_n, these vectors are short. Then, these vectors are reduced modulo $\mathcal{P}(B)$ to obtain $y_1, \ldots, y_m$. By Lemma 5, each of $y_1, \ldots, y_m$ is distributed almost uniformly in $\mathcal{P}(B)$. We now partition $\mathcal{P}(B)$ into a very fine grid containing q^n cells (see Figure 2). Each cell naturally corresponds to an element of $\mathbb{Z}_q^n$ and we define $a_i \in \mathbb{Z}_q^n$ as the element corresponding to the cell containing y_i. Notice that each a_i is distributed almost uniformly in $\mathbb{Z}_q^n$. We can therefore apply COLLISIONFIND to $a_1, \ldots, a_m$ and obtain a $\{-1, 0, 1\}$-combination of them that sums to zero in $\mathbb{Z}_q^n$. We then notice that the same combination applied to $x_1, \ldots, x_m$ is: (i) a short vector (since each x_i is short and the coefficients are at most 1 is absolute value) (ii) extremely close to a lattice vector (which must therefore be short as well). The procedure outputs this close-by lattice vector.

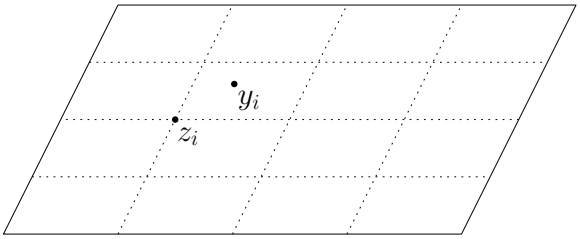

Figure 2: Partitioning a basic parallelepiped into 4^2 parts

Procedure 1 FINDVECTOR

Input: A lattice Λ given by an LLL-reduced basis B, and a parameter $\tilde{\eta}$ satisfying $2\eta_\varepsilon(\Lambda) \leq \tilde{\eta} \leq 4\eta_\varepsilon(\Lambda)$.
Output: A (short) element of Λ, or a message FAIL.

1: For each $i \in \{1, \ldots, m\}$ do the following:
2: Choose a random vector x_i from distribution $\nu_{\tilde{\eta}}$
3: Let $y_i = x_i \bmod \mathcal{P}(B)$
4: Consider the sub-parallelepiped containing y_i. Let a_i be the element of $\mathbb{Z}_q^n$ corresponding to it, and let z_i be its "lower-left" corner. In symbols, $a_i = \lfloor qB^{-1}y_i \rfloor$ and $z_i = Ba_i/q = B\lfloor qB^{-1}y_i \rfloor/q$.
5: Run COLLISIONFIND on $(a_1, \ldots, a_m)$. If it fails then output FAIL. Otherwise, we obtain $b_1, \ldots, b_m \in \{-1, 0, 1\}$, not all zero, such that $\sum_{i=1}^m b_i a_i = 0 \pmod q$
6: Return $\sum_{i=1}^m b_i(x_i - y_i + z_i)$

Later in this section, we will prove that FINDVECTOR satisfies the following properties:

- When it is successful, its output is a lattice vector, and with probability exponentially close to 1, its length is at most $O(n^3 \cdot \lambda_n(\Lambda))$
- It is successful with probability at least $n^{-c_0}/2$
- The distribution of its output is 'full-dimensional', in the sense that the probability that the output vector lies in any fixed $n-1$-dimensional hyperplane is at most 0.9.

Based on these properties, we can now describe the $\mathsf{SIVP}_{O(n^3)}$ algorithm. Given some basis of a lattice Λ, the algorithm starts by applying the LLL algorithm to obtain an LLL-reduced basis B. Assume for simplicity that we know a value $\tilde{\eta}$ as required by FINDVECTOR. We can then apply FINDVECTOR n^{c_0+2} times (where n^{-c_0} is the success probability of COLLISIONFIND). Among all vectors returned, we look for n linearly independent vectors. If such vectors are found, we output them; otherwise, we fail.

By the properties mentioned above, we see that among the n^{c_0+2} applications of FINDVECTOR made by our algorithm, the expected number of successful calls is at least $n^2/2$. Using standard arguments, we obtain that with very high probability, the number of successful calls is at least, say, $n^2/4$. Moreover, we see that with high probability all these vectors are lattice vectors of length at most $O(n^3 \cdot \lambda_n(\Lambda))$. Finally, we claim that these vectors contain n linearly independent vectors with very high probability. Indeed, as long as the dimension of the space spanned by the current vectors is less than n, each new vector increases it by one with probability at least 0.1. Hence, with very high probability, we find n linearly independent vectors.

It remains to explain how to find a parameter $\tilde{\eta}$ in the required range. Recall that the length of the longest vector in an LLL-reduced basis gives a 2^n approximation to λ_n. Together with Corollaries 8 and 10, we obtain a $n^{3/2}2^n$ approximation to $\eta_\varepsilon(\Lambda)$. We can therefore apply the algorithm described above with $n + \frac{3}{2}\log n$ guesses of $\tilde{\eta}$. One of them is guaranteed to be in the required range.

In the rest of this section, we show that FINDVECTOR satisfies the properties mentioned above.

CLAIM 11 *If* FINDVECTOR *does not fail, its output is a lattice vector.*

PROOF: Assuming FINDVECTOR is successful, its output is the vector $\sum_{i=1}^m b_i(x_i - y_i + z_i)$. Each $x_i - y_i$ is a lattice vector by the definition of y_i. Moreover,

$$\sum_{i=1}^m b_i z_i = B \sum_{i=1}^m b_i a_i/q$$

is a lattice vector because $\sum_{i=1}^m b_i a_i/q$ is an integer vector. □

The following claim shows that when FINDVECTOR is successful, its output is a short vector. By combining the bound below with Corollary 10 and our choice of m, we obtain a bound of $O(n^3 \cdot \lambda_n(\Lambda))$ on the length of the output.

CLAIM 12 *If $\tilde{\eta} \geq \eta_\varepsilon(\Lambda)$, the probability that* FINDVECTOR *outputs a vector v of length $\|v\| > 2m \cdot \sqrt{n} \cdot \tilde{\eta}$ is at most $2^{-\Omega(n)}$.*

PROOF: Using the triangle inequality and the fact that $b_i \in \{-1, 0, 1\}$ we get that

$$\left\| \sum_{i=1}^m b_i(x_i - y_i + z_i) \right\| \leq \sum_{i=1}^m |b_i| \cdot \|x_i - y_i + z_i\| \leq \sum_{i=1}^m \|x_i\| + \sum_{i=1}^m \|z_i - y_i\|.$$

We bound the two terms separately. First, each x_i is chosen independently from the distribution $\nu_{\tilde{\eta}}$. As we saw in the previous lecture, the probability that $\|x_i\| > \tilde{\eta} \cdot \sqrt{n}$ is at most $2^{-\Omega(n)}$. So the contribution of the first term is at most $m\sqrt{n}\tilde{\eta}$ except with probability $m \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$.

We now consider the second term. By the definition of z_i, both y_i and z_i are in the same sub-parallelepiped, so $\|z_i - y_i\| \leq \frac{1}{q} \cdot \mathrm{diam}(\mathcal{P}(B))$. This quantity is extremely small: indeed, by our choice of q and Corollary 8 we obtain

$$\|z_i - y_i\| \leq 2^{-2n} \cdot n \cdot 2^n \cdot \lambda_n(\Lambda) \leq 2^{-2n} \cdot n \cdot 2^n \cdot n \cdot \eta_\varepsilon(\Lambda) \ll \tilde{\eta}$$

where we used that B is LLL-reduced and therefore

$$\mathrm{diam}(\mathcal{P}(B)) \leq n \cdot 2^n \cdot \lambda_n(\Lambda).$$

□

CLAIM 13 *If $\tilde{\eta} \geq \eta_\varepsilon(\Lambda)$, algorithm* FINDVECTOR *succeeds with probability at least $\frac{1}{2} \cdot n^{-c_0}$.*

PROOF: By definition, COLLISIONFIND succeeds on a uniformly random input with probability at least n^{-c_0}. So it would suffice to show that the input we provide to COLLISIONFIND is "almost uniform", i.e., that the statistical distance between the m-tuple $(a_1, \ldots, a_m)$ and the uniform distribution on m-tuples of elements in $\mathbb{Z}_q^n$ is negligible.

To show this, notice that by Lemma 5, the statistical distance between the distribution of each y_i and the uniform distribution on $\mathcal{P}(B)$ is at most $\frac{1}{2}\rho_{1/\tilde{\eta}}(\Lambda^* \setminus \{0\})$. By our assumption on $\tilde{\eta}$, this quantity is at most $\frac{1}{2}\varepsilon$, which is negligible.

Now consider the function $f : \mathcal{P}(B) \to \mathbb{Z}_q^n$ given by $f(y) = \lfloor qB^{-1}y \rfloor \in \mathbb{Z}_q^n$. Then we can write $a_i = f(y_i)$. Moreover, it is easy to see that on input a uniform point y in $\mathcal{P}(B)$, $f(y)$ is a uniform element of $\mathbb{Z}_q^n$. These two observations, combined with the fact that statistical distance cannot increase by applying a function, imply that the statistical distance between a_i and the uniform distribution on $\mathbb{Z}_q^n$ is negligible. Since the a_i are chosen independently, the distance between the m-tuple $(a_1, \ldots, a_m)$ and the uniform distribution on $(\mathbb{Z}_q^n)^m$ is at most m times larger, which is still negligible. To summarize, we have the following sequence of inequalities:

$$\begin{aligned}\Delta((a_1, \ldots, a_m), (U(\mathbb{Z}_q^n))^m) &\leq \sum_{i=1}^{m} \Delta(a_i, U(\mathbb{Z}_q^n)) = \\ &= m \cdot \Delta\big(f(\nu_{\tilde{\eta}} \bmod \mathcal{P}(B)), f(U(\mathcal{P}(B)))\big) \leq \\ &\leq m \cdot \Delta(\nu_{\tilde{\eta}} \bmod \mathcal{P}(B), U(\mathcal{P}(B))) \leq \\ &\leq m \cdot \varepsilon.\end{aligned}$$

Since $m \cdot \varepsilon = 4n^2 \cdot n^{-\log n}$ is a negligible function, we are done. □

It remains to prove that the output of FINDVECTOR is full-dimensional. (Notice that so far we haven't even excluded the possibility that its output is constantly the zero vector!) We cannot make any assumptions on the behavior of COLLISIONFIND, and we need to argue that even if it 'acts maliciously', the vectors given by FINDVECTOR are full-dimensional. Essentially, the idea is the following. We note that COLLISIONFIND is only given the a_i. From this, it can deduce the z_i and also the y_i to within a good approximation. But, as we show later, it still has lots of uncertainty about the vectors x_i: conditioned on any fixed value for y_i, the distribution of x_i is full-dimensional. So no matter what COLLISIONFIND does, the distribution of the output vector is full-dimensional.

To argue this formally, it is helpful to imagine that the vectors x_i are chosen *after* we call COLLISIONFIND. This is done by introducing the following 'virtual' procedure FINDVECTOR$'$. We use the notation $D_{s,y}$ to denote the probability obtained by conditioning ν_s on the outcome x satisfying $x \bmod \mathcal{P}(B) = y$. More precisely, for any $x \in \Lambda + y$,

$$\Pr[D_{s,y} = x] = \frac{\nu_s(x)}{\nu_s(\Lambda + y)} = \frac{\rho_s(x)}{\rho_s(\Lambda + y)}.$$

We only use FINDVECTOR$'$ in our analysis and therefore it doesn't matter that we don't have an efficient way to sample from $D_{s,y}$. The important thing is that its output distribution is identical to that of FINDVECTOR.

We complete the analysis with the following lemma. It shows that for $s \geq \sqrt{2}\eta_\varepsilon(\Lambda)$ and any $n-1$-dimensional hyperplane H, the probability that a vector x chosen from $D_{s,y}$ is in H is at most 0.9. This implies that the same holds for the output distribution of FINDVECTOR$'$ (and hence also for that of FINDVECTOR). Indeed, consider Step 5. Not all b_i are zero, so assume for simplicity that $b_1 = 1$. Then for the output of the procedure to be in some $n-1$-dimensional hyperplane H, the vector x_1 must also be in some hyperplane (namely, $H + y_1 - z_1 - \sum_{i=2}^{m} b_i(x_i - y_i + z_i)$), which happens with probability at most 0.9.

Procedure 2 FINDVECTOR′

Input: A lattice Λ given by an LLL-reduced basis B, and a parameter $\tilde{\eta}$ satisfying $2\eta_\varepsilon(\Lambda) \leq \tilde{\eta} \leq 4\eta_\varepsilon(\Lambda)$.

Output: A (short) element of Λ, or a message FAIL.

1: For each $i \in \{1, \ldots, m\}$ do the following:
2: Choose y_i according to the distribution $\nu_{\tilde{\eta}} \bmod \mathcal{P}(B)$
3: Define $a_i = \lfloor qB^{-1}y_i \rfloor$ and $z_i = Ba_i/q = B\lfloor qB^{-1}y_i \rfloor/q$.
4: Run COLLISIONFIND on $(a_1, \ldots, a_m)$. If it fails then output FAIL. Otherwise, we obtain $b_1, \ldots, b_m \in \{-1, 0, 1\}$, not all zero, such that $\sum_{i=1}^m b_i a_i = 0$
5: For each $i \in \{1, \ldots, m\}$, choose x_i from the distribution $D_{\tilde{\eta}, y_i}$
6: Return $\sum_{i=1}^m b_i(x_i - y_i + z_i)$

LEMMA 14 *For $s \geq \sqrt{2}\eta_\varepsilon(\Lambda)$, any y and any $n-1$-dimensional hyperplane H, $\Pr_{x \sim D_{s,y}}[x \in H] < 0.9$.*

PROOF: Let $u \in \mathbb{R}^n$ be a unit vector and $c \in \mathbb{R}$ be such that $H = \{x \in \mathbb{R}^n \mid \langle x, u \rangle = c\}$. Without loss of generality, we can assume that $u = (1, 0, \ldots, 0)$ so $\langle x, u \rangle = x_1$. Clearly, it is enough to show that

$$\mathbf{E}_{x \sim D_{s,y}}[e^{-\pi(\frac{x_1 - c}{s})^2}] < 0.9.$$

The left hand side can be written as

$$\sum_{x \in \Lambda + y} \frac{\rho_s(x)}{\rho_s(\Lambda + y)} \cdot e^{-\pi(\frac{x_1 - c}{s})^2} = \frac{1}{\rho_s(\Lambda + y)} \cdot \sum_{x \in \Lambda + y} e^{-\pi\|\frac{x}{s}\|^2} e^{-\pi(\frac{x_1 - c}{s})^2}.$$

We now analyze this expression. Using the Poisson summation formula and the fact that $s \geq \eta_\varepsilon(\Lambda)$,

$$\rho_s(\Lambda + y) = \det \Lambda^* \cdot s^n \sum_{w \in \Lambda^*} \rho_{1/s}(w) e^{2\pi i \langle w, y \rangle} \geq \det \Lambda^* \cdot s^n \cdot (1 - \varepsilon).$$

To analyze the sum, we define

$$\begin{aligned} g(x) &:= e^{-\pi\|\frac{x}{s}\|^2} e^{-\pi(\frac{x_1 - c}{s})^2} \\ &= e^{-\frac{\pi}{s^2}(x_1^2 + (x_1 - c)^2 + x_2^2 + \cdots + x_n^2)} \\ &= e^{-\frac{\pi}{s^2}\frac{c^2}{2}} \cdot e^{-\frac{\pi}{s^2}((\sqrt{2}(x_1 - \frac{1}{2}c))^2 + x_2^2 + \cdots + x_n^2)}. \end{aligned}$$

From this we can see that the Fourier transform of g is given by

$$\hat{g}(w) = e^{-\frac{\pi}{s^2}\frac{c^2}{2}} \cdot e^{2\pi i w_1(-\frac{1}{2}c)} \cdot s^{n-1} \cdot \frac{s}{\sqrt{2}} \cdot e^{-\pi s^2((\frac{w_1}{\sqrt{2}})^2 + w_2^2 + \cdots + w_n^2)}$$

and in particular,

$$|\hat{g}(w)| \leq \frac{s^n}{\sqrt{2}} \cdot e^{-\pi s^2((\frac{w_1}{\sqrt{2}})^2 + w_2^2 + \cdots + w_n^2)} \leq \frac{s^n}{\sqrt{2}} \cdot \rho_{\sqrt{2}/s}(w).$$

We can now apply the Poisson summation formula and obtain

$$g(\Lambda + y) = \det \Lambda^* \sum_{w \in \Lambda^*} \hat{g}(w) \cdot e^{2\pi i \langle w, y \rangle} \leq \det \Lambda^* \sum_{w \in \Lambda^*} |\hat{g}(w)| \leq \det \Lambda^* \frac{s^n}{\sqrt{2}}(1 + \varepsilon)$$

where the last inequality follows from $s \geq \sqrt{2}\eta_\varepsilon(\Lambda)$. Combining the two bounds, we obtain

$$\mathbf{E}_{x \sim D_{s,y}}[e^{-\pi(\frac{x_1 - c}{2})^2}] \leq \frac{\det \Lambda^* s^n (1 + \varepsilon)/\sqrt{2}}{\det \Lambda^* s^n (1 - \varepsilon)} < 0.9.$$

□

4 Possible Improvements and Some Remarks

The reduction we presented here shows how to solve $\mathsf{SIVP}_{O(n^3)}$ using a collision finder. The best known reduction [6] achieves a solution to $\mathsf{SIVP}_{\tilde{O}(n)}$, where the $\tilde{O}$ hides polylogarithmic factors. This improvement is obtained by adding three more ideas to our reduction:

1. Use the bound $\eta_\varepsilon(\Lambda) \leq \log n \cdot \lambda_n(\Lambda)$ for $\varepsilon = n^{-\log n}$ described below in Lemma 15. This improves the approximation factor to $\tilde{O}(n^{2.5})$.

2. It can be shown that the summands of $\sum_{i=1}^m b_i(x_i - y_i + z_i)$ add up like random vectors, i.e., with cancellations. Therefore, the total norm is proportional to $\sqrt{m}$ and not m. This means that one can improve the bound in Claim 12 to $\tilde{O}(\sqrt{m} \cdot \sqrt{n} \cdot \tilde{\eta})$. Together with the previous improvement, this gives an approximation factor of $\tilde{O}(n^{1.5})$.

3. The last idea is to use an *iterative algorithm*. In other words, instead of obtaining an approximate solution to SIVP in one go, we obtain it in steps: starting with a set of long vectors, we repeatedly make it shorter by replacing long vectors with shorter ones. This allows us to choose a smaller value of q, say, $q = n^{10}$, which in turn allows us to choose $m = \tilde{\Theta}(n)$. This smaller value of m makes the length of the resulting basis only $\tilde{O}(n) \cdot \lambda_n(\Lambda)$. See [6] for more details.

Let us also mention two modifications to the basic reduction. First, notice that it is enough if COLLISIONFIND returns coefficients $b_1, \ldots, b_m$ that are "small", and not necessarily in $\{-1, 0, 1\}$. So finding small solutions to random modular equations is as hard as worst-case lattice problems. Another possible modification is to partition the basic parallelepiped into $p_1 p_2 \cdots p_n$ parts for some primes $p_1, \ldots, p_n$ (instead of q^n parts). This naturally gives rise to the group $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n} = \mathbb{Z}_{p_1 \cdots p_n}$. Hence, we see that finding small solutions to a random equation in $\mathbb{Z}_N$ (for an appropriate N) is also as hard as worst-case lattice problems.

Finally, we note that the basic reduction presented in the previous section is *non-adaptive* in the sense that all oracle queries can be made simultaneously. In contrast, in an adaptive reduction, oracle queries depend on answers from previous oracle queries and therefore cannot be made simultaneously. If we apply the iterative technique outlined above in order to gain an extra $\sqrt{n}$ in the approximation factor, then the reduction becomes adaptive.

4.1 A Tighter Bound on the Smoothing Parameter

LEMMA 15 *Let $\varepsilon = n^{-\log n}$. Then for any lattice Λ, $\eta_\varepsilon(\Lambda) \leq \log n \cdot \lambda_n(\Lambda)$.*

This lemma is essentially tight: consider, for instance, the lattice $\Lambda = \mathbb{Z}^n$. Then clearly $\lambda_n(\Lambda) = 1$. On the other hand, the dual lattice is also $\mathbb{Z}^n$ and we can therefore lower bound $\rho_{1/s}(\Lambda^* \setminus \{0\})$ by (say) $e^{-\pi s^2}$. To make this quantity at most ε, s should be at least $\Omega(\log n)$ and hence $\eta_\varepsilon(\Lambda) \geq \Omega(\log n)$.

PROOF: Let $v_1, \ldots, v_n$ be a set of n linearly independent vectors in Λ of length at most $\lambda_n(\Lambda)$ (such a set exists by the definition of λ_n). Take $s = \log n \cdot \lambda_n(\Lambda)$. Our goal is to show that $\rho_{1/s}(\Lambda^* \setminus \{0\})$ is smaller than ε. The idea is to show that for each i, almost all the contribution to $\rho_{1/s}(\Lambda^*)$ comes from vectors in Λ^* that are orthogonal to v_i. Since this holds for all i, we will conclude that almost all contribution must come from the origin. The origin's contribution is 1, hence $\rho_{1/s}(\Lambda^*)$ is essentially 1 and $\rho_{1/s}(\Lambda^* \setminus \{0\})$ is very small.

For $i = 1, \ldots, n$ and $j \in \mathbb{Z}$ we define

$$S_{i,j} = \{y \in \Lambda^* \mid \langle v_i, y \rangle = j\}.$$

If we recall the definition of the dual lattice, we see that for any i, the union of $S_{i,j}$ over all $j \in \mathbb{Z}$ is Λ^*. Moreover, if $S_{i,j}$ is not empty, then it is a translation of $S_{i,0}$ and we can write

$$S_{i,j} = S_{i,0} + w + ju_i$$

where $u_i = v_i/\|v_i\|^2$ is a vector of length $1/\|v_i\| \geq 1/\lambda_n(\Lambda)$ in the direction of v_i and w is some vector orthogonal to v_i. Using these properties, we see that if $S_{i,j}$ is not empty, then

$$\begin{aligned}\rho_{1/s}(S_{i,j}) &= e^{-\pi\|jsu_i\|^2}\rho_{1/s}(S_{i,0}+w)\\ &\leq e^{-\pi\|jsu_i\|^2}\rho_{1/s}(S_{i,0})\\ &\leq e^{-\pi j^2\log^2 n}\rho_{1/s}(S_{i,0}).\end{aligned}$$

where the first inequality follows from a lemma in the previous lecture. Hence,

$$\begin{aligned}\rho_{1/s}(\Lambda^* \setminus S_{i,0}) &= \sum_{j\neq 0}\rho_{1/s}(S_{i,j})\\ &\leq \rho_{1/s}(S_{i,0})\sum_{j\neq 0} e^{-\pi j^2\log^2 n}\\ &\leq \rho_{1/s}(S_{i,0})\cdot n^{-2\log n} \leq n^{-2\log n}\rho_{1/s}(\Lambda^*).\end{aligned}$$

Since $v_1, \ldots, v_n$ are linearly independent,

$$\Lambda^* \setminus \{0\} = \bigcup_{i=1}^{n}(\Lambda^* \setminus S_{i,0})$$

and therefore

$$\begin{aligned}\rho_{1/s}(\Lambda^* \setminus \{0\}) &\leq \sum_{i=1}^{n}\rho_{1/s}(\Lambda^* \setminus S_{i,0})\\ &\leq n\cdot n^{-2\log n}\cdot\rho_{1/s}(\Lambda^*)\\ &= n^{-2\log n+1}(1+\rho_{1/s}(\Lambda^* \setminus \{0\})).\end{aligned}$$

We obtain the result by rearranging. □

References

[1] M. Ajtai. Generating hard instances of lattice problems. In *Proc. of 28th STOC*, pages 99–108, 1996. Available from ECCC.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM Symp. on Theory of Computing (STOC)*, pages 284–293, 1997.

[3] J. Cai and A. Nerurkar. An improved worst-case to ity average-case connection for lattice problems. In *Proc. of 38th FOCS*, pages 468–477, 1997.

[4] O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Technical report, TR96-056, Electronic Colloquium on Computational Complexity (ECCC), 1996.

[5] D. Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *Proc. of 34th STOC*, pages 609–618, 2002.

[6] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 372–381, 2004.

[7] O. Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in STOC'03.