Just Take the Average! An Embarrassingly Simple 2^n-Time Algorithm for SVP (and CVP)

Divesh Aggarwal[*1] **and Noah Stephens-Davidowitz**[†2]

1 **CQT and Department of Computer Science, NUS, Singapore**
dcsdiva@nus.edu.sg
2 **New York University, New York, USA**
noahsd@gmail.com

Abstract

We show a $2^{n+o(n)}$-time (and space) algorithm for the Shortest Vector Problem on lattices (SVP) that works by repeatedly running an embarrassingly simple "pair and average" sieving-like procedure on a list of lattice vectors. This matches the running time (and space) of the current fastest known algorithm, due to Aggarwal, Dadush, Regev, and Stephens-Davidowitz (ADRS, in *STOC*, 2015), with a far simpler algorithm. Our algorithm is in fact a modification of the ADRS algorithm, with a certain careful rejection sampling step removed.

The correctness of our algorithm follows from a more general "meta-theorem," showing that such rejection sampling steps are unnecessary for a certain class of algorithms and use cases. In particular, this also applies to the related $2^{n+o(n)}$-time algorithm for the Closest Vector Problem (CVP), due to Aggarwal, Dadush, and Stephens-Davidowitz (ADS, in *FOCS*, 2015), yielding a similar embarrassingly simple algorithm for γ-approximate CVP for any $\gamma = 1 + 2^{-o(n/\log n)}$. (We can also remove the rejection sampling procedure from the $2^{n+o(n)}$-time ADS algorithm for *exact* CVP, but the resulting algorithm is still quite complicated.)

1998 ACM Subject Classification F.2.2 Nonnumerical algorithms and problems – Geometrical problems and computations

Keywords and phrases Lattices, SVP, CVP

Digital Object Identifier 10.4230/OASIcs.SOSA.2018.12

1 Introduction

A lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of some linearly independent basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^n$,

$$\mathcal{L} := \left\{ \sum_{i=1}^{n} z_i \boldsymbol{b}_i \ : \ z_i \in \mathbb{Z} \right\}.$$

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Given a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ for a lattice

* The first author was supported by the Singapore Ministry of Education and the National Research Foundation, also through the Tier 3 Grant "Random numbers from quantum processes" MOE2012-T3-1-009.
† The second author was supported by the National Science Foundation (NSF) under Grant No. CCF-1320188, and the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236.

$\mathcal{L} \subset \mathbb{R}^n$, SVP asks us to find a shortest non-zero vector in $\mathcal{L}$, and CVP asks us to find a closest lattice vector to some target vector $\boldsymbol{t} \in \mathbb{R}^n$. (Throughout this paper, we define distance in terms of the Euclidean, or ℓ_2, norm.) CVP seems to be the harder of the two problems, as there is an efficient reduction from CVP to SVP that preserves the dimension n [17], but both problems are known to be NP-hard [36, 3]. They are even known to be hard to approximate for certain approximation factors [24, 14, 21, 18].

Algorithms for solving these problems, both exactly and over a wide range of approximation factors, have found innumerable applications since the founding work by Lenstra, Lenstra, and Lovász in 1982 [23]. (E.g., [23, 19, 34, 20, 13].) More recently, following the celebrated work of Ajtai [4] and Regev [31], a long series of works has resulted in many cryptographic constructions whose security is based on the assumed *worst-case* hardness of approximating these (or closely related) problems. (See [29] for a survey of such constructions.) And, some of these constructions are now nearing widespread deployment. (See, e.g., [28, 7, 11].)

Nearly all of the fastest known algorithms for lattice problems – either approximate or exact – work via a reduction to either exact SVP or exact CVP (typically in a lower dimension). Even the fastest known polynomial-time algorithms (which solve lattice problems only up to large approximation factors) work by solving exact SVP on low-dimensional sublattices [33, 15, 27]. Therefore, algorithms for exact lattice problems are of particular importance, both theoretically and practically (and both for direct applications and to aid in the selection of parameters for cryptography). Indeed, much work has gone into improving the running time of these algorithms (e.g., [20, 5, 6, 30, 25, 26]), culminating in $2^{n+o(n)}$-time algorithms for both problems based on the technique of *discrete Gaussian sampling*, from our joint work with Dadush and Regev [1] and follow-up work with Dadush [2].

In order to explain our contribution, we first give a high-level description of the SVP algorithm from [1], which we refer to as the ADRS algorithm. (The presentation below does not represent the way that we typically view that algorithm.)

1.1 Sieving by averages

One can think of the ADRS algorithm as a rather strange variant of *randomized sieving*. Recall that the celebrated randomized sieving technique due to Ajtai, Kumar, and Sivakumar [5] starts out with a list of $2^{O(n)}$ not-too-long random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M$ sampled from some efficiently samplable distribution. The sieving algorithm then repeatedly (1) searches for pairs of vectors $(\boldsymbol{X}_i, \boldsymbol{X}_j)$ that happen to be remarkably close together; and then (2) replaces the old list of vectors with the differences of these pairs $\boldsymbol{X}_i - \boldsymbol{X}_j$.

The ADRS algorithm similarly starts with a randomly chosen collection of $2^{n+o(n)}$ not-too-long vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M$ and repeatedly (1) selects pairs of vectors according to some rule; and (2) replaces the old list of vectors with some new vectors generated from these pairs. However, instead of taking the differences $\boldsymbol{X}_i - \boldsymbol{X}_j$ of pairs $(\boldsymbol{X}_i, \boldsymbol{X}_j)$, the ADRS algorithm takes *averages*, $(\boldsymbol{X}_i + \boldsymbol{X}_j)/2$.

Notice that the average $(\boldsymbol{X}_i + \boldsymbol{X}_j)/2$ of two lattice vectors $\boldsymbol{X}_i, \boldsymbol{X}_j \in \mathcal{L}$ will not generally be in the lattice. In fact, this average will be in the lattice if and only if the two vectors are equivalent mod $2\mathcal{L}$, i.e., $\boldsymbol{X}_i \equiv \boldsymbol{X}_j \bmod 2\mathcal{L}$. Therefore, at a minimum, the ADRS algorithm should select pairs that lie in the same *coset* mod $2\mathcal{L}$. (Notice that there are 2^n possible cosets.) I.e., the simplest possible version of "sieving by averages" just repeats Procedure 1 many times (starting with a list of $2^{n+o(n)}$ vectors, which is sufficient to guarantee that we can pair nearly every vector with a different unique vector in the same coset). The ADRS algorithm is more complicated than this, but it still only uses the cosets of the vectors mod $2\mathcal{L}$ when it decides which vectors to pair.

It might seem like a rather big sacrifice to only look at a vector's coset, essentially ignoring all geometric information. For example, the ADRS algorithm (and our variant) is likely to

Procedure 1: The basic "pair and average" procedure, which computes the averages $(\boldsymbol{X}_i + \boldsymbol{X}_j)/2$ of disjoint pairs $(\boldsymbol{X}_i, \boldsymbol{X}_j)$ satisfying $\boldsymbol{X}_i \equiv \boldsymbol{X}_j \bmod 2\mathcal{L}$.

Pair_and_Average $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$
Input : List of vectors $\boldsymbol{X}_i \in \mathcal{L} - \boldsymbol{t}$
Output: List of vectors $\boldsymbol{Y}_i \in \mathcal{L} - \boldsymbol{t}$
for *each unpaired vector* $\boldsymbol{X}_i$ **do**
  if *there exists an unpaired vector* $\boldsymbol{X}_j$ *with* $\boldsymbol{X}_j \equiv \boldsymbol{X}_i \bmod 2\mathcal{L}$ **then**
    add $(\boldsymbol{X}_i + \boldsymbol{X}_j)/2$ to the output
  end
end

Procedure 2: The "reject and average" sieving procedure, which repeatedly applies some rejection sampling procedure f according to the cosets of the $\boldsymbol{X}_i \bmod 2\mathcal{L}$ and then applies Procedure 1 (Pair_and_Average) to the "accepted" vectors. Here, f is some (possibly randomized) function that maps a list of cosets $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M) \bmod 2\mathcal{L}$ to a set of "accepted" indices $\{j_1, \ldots, j_m\} \subseteq \{1, \ldots, M\}$.

Reject-and-Average Sieve $(\ell; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$
Input : Number of steps ℓ, list of vectors $\boldsymbol{X}_i \in \mathcal{L} - \boldsymbol{t}$
Output: List of vectors $\boldsymbol{Y}_i \in \mathcal{L} - \boldsymbol{t}$
for $i = 1, \ldots, \ell$ **do**
  for $j = 1, \ldots, M$ **do**
    set $\boldsymbol{c}_j$ to be the coset of $\boldsymbol{X}_j \bmod 2\mathcal{L}$
  end
  $\{j_1, \ldots, j_m\} \leftarrow f(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M)$
  $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{M'}) \leftarrow$ Pair_and_Average$(\boldsymbol{X}_{j_1}, \ldots, \boldsymbol{X}_{j_m})$
  $M \leftarrow M'$
end
output $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$.

miss many opportunities to pair two vectors whose average is very short. But, in exchange for this sacrifice, we get very strong control over the distribution of the vectors at each step. In particular, before applying Procedure 1, the ADRS algorithm uses a careful *rejection sampling* procedure over the cosets to guarantee that at each step of the algorithm, the vectors are distributed as independent samples from a distribution that we understand very well (the discrete Gaussian, which we describe in Section 1.3). I.e., at each step, the algorithm randomly throws away many of the vectors in each coset according to some rule that depends only on the list of cosets, and it only runs Procedure 1 on the remaining vectors, as shown in Procedure 2. This rejection sampling procedure is so selective that, though the algorithm starts out with $2^{n+o(n)}$ vectors, it typically finishes with only about $2^{n/2}$ vectors.

This does seem quite wasteful (since the algorithm typically throws away the vast majority of its input vectors) and a bit naive (since the algorithm ignores, e.g., the lengths of the vectors). But, because we get such good control over the output distribution, the result is still the fastest known algorithm for SVP.[1] (The algorithm for CVP in [2], which we refer

[1] There are various "heuristic" sieving algorithms for SVP that run significantly faster (e.g., in time $(3/2)^{n/2}$ [10]) but do not have formal proofs of correctness. One of the reasons that these algorithms lack proofs is because we do not understand their output distributions.

to as the ADS algorithm, relies on the same core idea, plus a rather complicated recursive procedure that converts an approximate CVP algorithm with certain special properties into an exact CVP algorithm.)

1.2 Our contribution

Our main contribution is to show that the rejection sampling procedure used in the ADRS algorithm is unnecessary! Indeed, informally, we show that "any collection of vectors that can be found via such a procedure (when the input vectors are sampled independently from an appropriate distribution) can also be found without it." (We make this precise in Theorem 9.) In particular, the SVP algorithm in [1] can be replaced by an extremely simple algorithm, which starts with a list of $2^{n+o(n)}$ vectors sampled from the right distribution and then just runs Procedure 1 repeatedly. (Equivalently, it runs Procedure 2 with f taken to be the trivial function that always outputs all indices, $\{1, \ldots, M\}$.)

▶ **Theorem 1** (SVP, informal). *There is a $2^{n+o(n)}$-time (and space) algorithm for SVP that starts with $2^{n+o(n)}$ vectors sampled from the same distribution as the ADRS algorithm and then simply applies Procedure 1 repeatedly, $\ell = O(\log n)$ times.*

The situation for CVP is, alas, more complicated because Procedure 2 is not the most difficult part of the *exact* CVP algorithm from [2]. Indeed, while this algorithm does run Procedure 2 and we *do* show that we can remove the rejection sampling procedure, the resulting algorithm retains the complicated recursive structure of the original algorithm. However, [2] also shows a much simpler non-recursive version of the algorithm that solves CVP up to an extremely good approximation factor. If we are willing to settle for such an algorithm, then we get the same result for CVP.

▶ **Theorem 2** (CVP, informal). *There is a $2^{n+o(n)}$-time (and space) algorithm that approximates CVP up to an approximation factor γ for any $\gamma = 1 + 2^{-o(n/\log n)}$ that starts with $2^{n+o(n)}$ vectors from the same distribution as the ADS algorithm and then simply applies Procedure 1 repeatedly, $\ell = o(n/\log n)$ times.*

In practice, such a tiny approximation factor is almost always good enough for applications.

1.3 Proof techniques

To describe the technical ideas behind our result, we now define the *discrete Gaussian distribution*, which plays a fundamental role in the algorithms in [1, 2] and a big part in our analysis. For any vector $\boldsymbol{x} \in \mathbb{R}^n$ and parameter $s > 0$, we define its Gaussian mass as

$$\rho_s(\boldsymbol{x}) := \exp(-\pi \|\boldsymbol{x}\|^2 / s^2) \ ,$$

and we extend this definition to a shift of a lattice $\mathcal{L} \subset \mathbb{R}^n$ with shift vector $\boldsymbol{t} \in \mathbb{R}^n$ in the natural way,

$$\rho_s(\mathcal{L} - \boldsymbol{t}) := \sum_{\boldsymbol{y} \in \mathcal{L}} \rho_s(\boldsymbol{y} - \boldsymbol{t}) \ .$$

The *discrete Gaussian distribution* $D_{\mathcal{L}-\boldsymbol{t},s}$ is the probability distribution over $\mathcal{L} - \boldsymbol{t}$ induced by this measure, given by

$$\Pr_{\boldsymbol{X} \sim D_{\mathcal{L}-\boldsymbol{t},s}}[\boldsymbol{X} = \boldsymbol{y} - \boldsymbol{t}] := \frac{\rho_s(\boldsymbol{y} - \boldsymbol{t})}{\rho_s(\mathcal{L} - \boldsymbol{t})}$$

for any $\boldsymbol{y} \in \mathcal{L}$.

For very large parameters $s > 0$, we can sample from the discrete Gaussian $D_{\mathcal{L}-\boldsymbol{t},s}$ efficiently [16, 12]. (Notice that $D_{\mathcal{L}-\boldsymbol{t},s}$ tends to concentrate on shorter vectors as the parameter $s > 0$ gets smaller. In particular, [1] showed that about 1.38^n independent samples from the discrete Gaussian $D_{\mathcal{L},s}$ with an appropriately chosen parameter s will contain a shortest non-zero lattice vector with high probability. See Proposition 16.) So, in [1, 2], we use Procedure 2 with a carefully chosen rejection sampling procedure f in order to convert many independent samples from $D_{\mathcal{L}-\boldsymbol{t},s}$ with a relatively large parameter s to some smaller number of independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/2^{\ell/2}}$.

This rejection sampling is certainly necessary if we wish to use Procedure 1 to sample from the discrete Gaussian distribution. Our new observation is that, even when we do not do this rejection sampling, the output of Procedure 1 still has a nice distribution. In particular, if we fix the coset mod $2\mathcal{L}$ of a pair of discrete Gaussian vectors $(\boldsymbol{X}_i, \boldsymbol{X}_j)$ with parameter $s > 0$, then their average will be distributed as a *mixture* of discrete Gaussians with parameter $s/\sqrt{2}$ over the cosets of $2\mathcal{L}$. I.e., while the probability of their average landing in any particular coset will not in general be proportional to the Gaussian mass of the coset, the distribution *inside each coset* will be exactly Gaussian. (See Lemma 6.)

This observation is sufficient to prove that no matter what rejection sampling procedure f we use in Procedure 2, if the input consists of independent samples from $D_{\mathcal{L}-\boldsymbol{t},s}$, the output will always be distributed as some *mixture* of samples from $D_{2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t},s/2^{\ell/2}}$ over the cosets $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$. I.e., while the output distribution might distribute weight amongst the cosets differently, if we condition on a fixed number of vectors landing in each coset, the output will always be distributed as independent discrete Gaussian vectors with parameter $s/2^{\ell/2}$. It follows immediately that "rejection sampling cannot help us." In particular, the probability that the output of Procedure 2 will contain a particular vector (say a shortest non-zero vector) with any rejection sampling procedure f will never be greater than the probability that we would see that vector without rejection sampling (i.e., when f is the trivial function that outputs $\{1, \ldots, M\}$).[2] See Corollary 8 and Theorem 9 for more detail.

1.4 An open problem – towards a $2^{n/2}$-time algorithm

Our result shows that all known applications of the $2^{n+o(n)}$-time discrete Gaussian sampling algorithms in [1, 2] work just as well if we remove the rejection sampling procedure from these algorithms. This in particular includes the SVP application mentioned in Theorem 1 and the approximate CVP application mentioned in Theorem 2. (More generally, we can remove the rejection sampling procedure from any application that simply relies on finding a set of vectors with a certain property in the output distribution, such as a shortest non-zero vector, all shortest non-zero vectors, a vector that is close to a shortest lattice vector in $\mathcal{L} - \boldsymbol{t}$, etc.)

However, [1] also presents a $2^{n/2+o(n)}$-time algorithm that samples from $D_{\mathcal{L}-\boldsymbol{t},s}$ as long as the parameter $s > 0$ is not too small. (In particular, we need $s \geq \sqrt{2}\eta_{1/2}(\mathcal{L})$, where $\eta_{1/2}(\mathcal{L})$ is the *smoothing parameter* of the lattice. See [1] or [35] for the details.) This algorithm is similar to the $2^{n+o(n)}$-time algorithms in that it starts with independent discrete Gaussian vectors with some high parameter, and it gradually lowers the parameter using a rejection sampling procedure together with a procedure that takes the averages of pairs of vectors that lie in the same coset modulo some sublattice. But, it fails for smaller parameters specifically

[2] Notice that this property is far from obvious without the observation that the output distribution is always a mixture of Gaussians over the cosets. For example, if we modified Procedure 2 so that f acted on the $\boldsymbol{X}_i$ themselves, rather than just their cosets mod $2\mathcal{L}$, then this property would no longer hold.

because the rejection sampling procedure that it uses must throw out too many vectors in this case. (In [35], we use a different rejection sampling procedure that never throws away too many vectors, but we do not know how to implement it in $2^{n/2+o(n)}$ time for small parameters $s < \sqrt{2}\eta_{1/2}(\mathcal{L})$.) If we could find a suitable variant of this algorithm that works for small parameters, we would be able to solve SVP in $2^{n/2+o(n)}$ time.

So, we are naturally very interested in understanding what happens when we simply remove the rejection sampling procedure from this algorithm. And, the fact that this works out so nicely for the $2^{n+o(n)}$-time algorithm works seems quite auspicious! Unfortunately, we are unable to say very much at all about the resulting distribution in the $2^{n/2+o(n)}$-time case.[3] So, we leave the study of this distribution as an open problem.

Organization

In Section 2, we review a few basic facts necessary to prove our main "meta-theorem," Theorem 9, which shows that "rejection sampling is unnecessary." In Section 3, we finish this proof. In particular, this implies Theorem 1 and 2. For completeness, in the appendix, we prove these theorems more directly and show the resulting algorithms in full detail.

2 Preliminaries

We write $\mathbb{N} := \{0, 1, \ldots\}$ for the natural numbers (including zero). We make little to no distinction between a random variable and its distribution. For $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, we write $\|\boldsymbol{x}\| := (x_1^2 + \cdots + x_n^2)^{1/2}$ for the Euclidean norm of $\boldsymbol{x}$. For any set S, we write $S^* := \{(x_1, \ldots, x_M) \ : \ x_i \in S\}$ for the set lists over S of finite length. (The order of elements in a list $\mathcal{M} \in S^*$ will never concern us. We could therefore instead use multisets.)

2.1 Lattices

A lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of integer linear combinations

$$\mathcal{L} := \{a_1\boldsymbol{b}_1 + \cdots + a_n\boldsymbol{b}_n \ : \ a_i \in \mathbb{Z}\}$$

of some linearly independent basis vectors $\mathbf{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. We sometimes write $\mathcal{L}(\mathbf{B})$ for the lattice spanned by $\mathbf{B}$.

We write $\mathcal{L}/(2\mathcal{L})$ for the set of cosets of $\mathcal{L}$ over $2\mathcal{L}$. E.g., if $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is a basis for $\mathcal{L}$, then each coset $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$ corresponds to a unique vector $a_1\boldsymbol{b}_1 + \cdots + a_n\boldsymbol{b}_n$ with $a_i \in \{0, 1\}$, and this correspondence is a bijection. Notice that the cosets in $\mathcal{L}/(2\mathcal{L})$ have a group structure under addition that is isomorphic to $\mathbb{Z}_2^n$.

2.2 The discrete Gaussian

For a parameter $s > 0$ and vector $\boldsymbol{x} \in \mathbb{R}^n$, we write

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

[3] After one step of "pairing and averaging," we know exactly the distribution that we get, and it *is* a weighted combination of Gaussians over the cosets of a certain sublattice! This seems quite auspicious. Unfortunately, the particular sublattice is not the same sublattice that we use to pair the vectors in the next step, and we therefore are unable to say much at all about what happens even after two steps.

for the *Gaussian mass of* $\boldsymbol{x}$ *with parameter* $s > 0$. Up to scaling, the Gaussian mass is the unique function on $\mathbb{R}^n$ that is invariant under rotations and a product function. In particular, it satisfies the following nice rotation identity,

$$\rho_s(\boldsymbol{x})\rho_s(\boldsymbol{y}) = \rho_{\sqrt{2}s}(\boldsymbol{x}+\boldsymbol{y})\rho_{\sqrt{2}s}(\boldsymbol{x}-\boldsymbol{y}) \tag{1}$$

for any parameter $s > 0$ and vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. This identity is fundamental to the results of [1, 2]. (See [32, 35] for a more detailed description of this connection and some additional results.)

We extend the Gaussian mass to a shift $\boldsymbol{t} \in \mathbb{R}^n$ of a lattice $\mathcal{L} \subset \mathbb{R}^n$ in the natural way,

$$\rho_s(\mathcal{L}-\boldsymbol{t}) := \sum_{\boldsymbol{y}\in\mathcal{L}} \rho_s(\boldsymbol{y}-\boldsymbol{t}) \ ,$$

and we call this the *Gaussian mass of* $\mathcal{L}-\boldsymbol{t}$ *with parameter* s.

We will need the following identity from [2]. (See [32, 35] for a much more general identity.)

▶ **Lemma 3.** *For any lattice* $\mathcal{L} \subset \mathbb{R}^n$*, shift* $\boldsymbol{t}$*, and parameter* $s > 0$*, we have*

$$\sum_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})^2 = \rho_{s/\sqrt{2}}(\mathcal{L})\rho_{s/\sqrt{2}}(\mathcal{L}-\boldsymbol{t}) \ .$$

Proof. We have

$$\begin{aligned}
\sum_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})^2 &= \sum_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \sum_{\boldsymbol{y}_1,\boldsymbol{y}_2\in\mathcal{L}} \rho_s(2\boldsymbol{y}_1+\boldsymbol{c}-\boldsymbol{t})\rho_s(2\boldsymbol{y}_2+\boldsymbol{c}-\boldsymbol{t}) \\
&= \sum_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \sum_{\boldsymbol{y}_1,\boldsymbol{y}_2\in\mathcal{L}} \rho_{s/\sqrt{2}}(\boldsymbol{y}_1+\boldsymbol{y}_2+\boldsymbol{c}-\boldsymbol{t})\rho_{s/\sqrt{2}}(\boldsymbol{y}_1-\boldsymbol{y}_2) \\
&= \sum_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \sum_{\boldsymbol{w},\boldsymbol{y}_1\in\mathcal{L}} \rho_{s/\sqrt{2}}(2\boldsymbol{y}_1-\boldsymbol{w}+\boldsymbol{c}-\boldsymbol{t})\rho_{s/\sqrt{2}}(\boldsymbol{w}) \\
&= \rho_{s/\sqrt{2}}(\mathcal{L}-\boldsymbol{t}) \sum_{\boldsymbol{w}\in\mathcal{L}} \rho_{s/\sqrt{2}}(\boldsymbol{w}) \\
&= \rho_{s/\sqrt{2}}(\mathcal{L}-\boldsymbol{t})\rho_{s/\sqrt{2}}(\mathcal{L}) \ ,
\end{aligned}$$

as needed. ◀

2.3 Dominating distributions

Intuitively, we say that some random list $\mathcal{M} \in S^*$ dominates another random list $\mathcal{M}' \in S^*$ if for every fixed list $\mathcal{S} \in S^*$, "$\mathcal{M}$ is at least as likely to contain $\mathcal{S}$ as a subsequence as $\mathcal{M}'$ is."

▶ **Definition 4** (Dominating distribution)**.** For some finite set S (which we identify with $\{1,\dots,N\}$ without loss of generality) and two random lists $\mathcal{M} := (X_1,\dots,X_M) \in S^*$ and $\mathcal{M}' := (X'_1,\dots,X'_{M'}) \in S^*$ (where M and M' might themselves be random variables), we say that $\mathcal{M}$ *dominates* $\mathcal{M}'$ if for any $(k_1,\dots,k_N) \in \mathbb{N}^N$,

$$\Pr[|\{j \ : \ X_j = i\}| \geq k_i, \ \forall i] \geq \Pr[|\{j \ : \ X'_j = i\}| \geq k_i, \ \forall i] \ .$$

We note the following basic facts about dominant distributions, which show that domination yields a partial order over random variables on S^*, and that this partial order behaves nicely under taking sublists.

▶ **Fact 5.** *For any finite set S and random variable $\mathcal{M} \in S^*$ that dominates some other random variable $\mathcal{M}' \in S^*$,*

1. *$\mathcal{M}$ dominates itself;*
2. *if $\mathcal{M}'$ dominates some random variable $\mathcal{M}'' \in S^*$, then $\mathcal{M}$ also dominates $\mathcal{M}''$; and*
3. *for any function $f : S^* \to S^*$ that maps a list of elements to a sublist, $\mathcal{M}$ dominates $f(\mathcal{M}')$.*

3 No need for rejection!

We now show our main observation: if $\boldsymbol{X}_1 \in \mathcal{L} - \boldsymbol{t}$ and $\boldsymbol{X}_2 \in \mathcal{L} - \boldsymbol{t}$ are sampled from the discrete Gaussian over a fixed coset $2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t}$ for some $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$, then their average $(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2$ is distributed as a mixture of Gaussians over the cosets $2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t}$ for $\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})$ with parameter lowered by a factor of $\sqrt{2}$.

▶ **Lemma 6.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, parameter $s > 0$, coset $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$, $s > 0$, and $\boldsymbol{y} \in \mathcal{L}$, we have*

$$\Pr_{\boldsymbol{X}_1, \boldsymbol{X}_2 \sim D_{2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t}, s}}[(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2 = \boldsymbol{y} - \boldsymbol{t}] = \rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t}) \cdot \frac{\rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} + \boldsymbol{y})}{\rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})^2} \ .$$

In particular, for any $\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})$ and $\boldsymbol{y} \in 2\mathcal{L} + \boldsymbol{d}$,

$$\Pr_{\boldsymbol{X}_1, \boldsymbol{X}_2 \sim D_{2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t}, s}}[(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2 = \boldsymbol{y} - \boldsymbol{t} \mid (\boldsymbol{X}_1 + \boldsymbol{X}_2)/2 \in 2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t}] = \frac{\rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t})}{\rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t})} \ .$$

Proof. We have

$$\begin{aligned}
&\rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})^2 \cdot \Pr_{\boldsymbol{X}_1, \boldsymbol{X}_2 \sim D_{2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t}, s}}[(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2 = \boldsymbol{y} - \boldsymbol{t}] \\
&\quad = \sum_{\boldsymbol{x} \in 2\mathcal{L} + \boldsymbol{c}} \rho_s(\boldsymbol{x} - \boldsymbol{t}) \rho_s(2\boldsymbol{y} - \boldsymbol{x} - \boldsymbol{t}) \\
&\quad = \rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t}) \sum_{\boldsymbol{x} \in 2\mathcal{L} + \boldsymbol{c}} \rho_{s/\sqrt{2}}(\boldsymbol{x} - \boldsymbol{y}) \qquad \text{(Eq. (1))} \\
&\quad = \rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t}) \rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} + \boldsymbol{y}) \ ,
\end{aligned}$$

as needed. The "in particular" then follows from the fact that $\rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} + \boldsymbol{y}) = \rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} + \boldsymbol{d})$ is constant for $\boldsymbol{y} \in 2\mathcal{L} + \boldsymbol{d}$ for some fixed $\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})$. ◀

Lemma 6 motivates the following definition, which captures a key property of the distribution described in Lemma 6.

▶ **Definition 7.** For a lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, and parameter $s > 0$ we say that the random list $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M) \in (\mathcal{L} - \boldsymbol{t})^*$ is a *mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s* if the "distributions within the cosets of $2\mathcal{L}$" are independent Gaussians with parameter s. I.e., for any list of cosets $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M) \in ((\mathcal{L} - \boldsymbol{t})/(2\mathcal{L}))^* \bmod 2\mathcal{L}$, if we *condition on* $\boldsymbol{X}_i \in 2\mathcal{L} + \boldsymbol{c}_i$ for all i, then the $\boldsymbol{X}_i$ are independent with $\boldsymbol{X}_i \sim D_{2\mathcal{L}+\boldsymbol{c}_i, s}$.

We call $(2\mathcal{L} + \boldsymbol{X}_1, \ldots, 2\mathcal{L} + \boldsymbol{X}_M)$ the *coset distribution* of the $\boldsymbol{X}_i$. We say that a mixture of independent Gaussians $\mathcal{M}$ over $\mathcal{L} - \boldsymbol{t}$ with parameter $s > 0$ dominates another, $\mathcal{M}'$, if the coset distribution of $\mathcal{M}$ dominates the coset distribution of $\mathcal{M}'$ (as in Definition 4).

In other words, mixtures of independent Gaussians are exactly the distributions obtained by first sampling $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M) \in ((\mathcal{L} - \boldsymbol{t})/(2\mathcal{L}))^*$ from some arbitrary coset distributions and then sampling $\boldsymbol{X}_i \sim D_{\mathcal{L}+\boldsymbol{c}_i, s}$ independently for each i. We now list some basic facts that follow from what we have done so far.

▶ **Corollary 8** (Properties of mixtures of Gaussians and Procedure 1). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, and parameter $s > 0$,*

1. *a mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s is uniquely characterized by its coset distribution;*
2. *if we apply Procedure 1 to a mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s, the result will be a mixture of Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter $s/\sqrt{2}$;*
3. *Procedure 1 preserves domination – i.e., if we apply Procedure 1 to two mixtures $\mathcal{M}, \mathcal{M}'$ of Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s and $\mathcal{M}$ dominates $\mathcal{M}'$, then the output of Procedure 1 on input $\mathcal{M}$ will dominate that of $\mathcal{M}'$; and*
4. *if $\boldsymbol{X}_1, \boldsymbol{X}_2$ are a mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s with coset distribution given by $\boldsymbol{X}_1 \equiv \boldsymbol{X}_2 \bmod 2\mathcal{L}$ and*

$$\Pr[2\mathcal{L} + \boldsymbol{X}_1 + \boldsymbol{t} = \boldsymbol{c}] = \frac{\rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})^2}{\sum_{\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t})^2}$$

for any $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$, then their average $(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2$ is distributed exactly as $D_{\mathcal{L}-\boldsymbol{t}, s/\sqrt{2}}$.

Proof. Item 1 follows immediately from the definition of a mixture of Gaussians. Items 2 and 3 are immediate consequences of Lemma 6.

For Item 4, we apply Lemma 6 to see that for any $\boldsymbol{y} \in \mathcal{L}$,

$$\begin{aligned}\Pr[(\boldsymbol{X}_1 + \boldsymbol{X}_2)/2 = \boldsymbol{y} - \boldsymbol{t}] &= \frac{\rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t})}{\sum_{\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t})^2} \sum_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} + \boldsymbol{y}) \\ &= \frac{\rho_{s/\sqrt{2}}(\boldsymbol{y} - \boldsymbol{t})}{\sum_{\boldsymbol{d} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{d} - \boldsymbol{t})^2} \cdot \rho_{s/\sqrt{2}}(\mathcal{L}) \ .\end{aligned}$$

The result then follows from Lemma 3. (Indeed, summing the left-hand side and the right-hand side over all $\boldsymbol{y} \in \mathcal{L}$ gives a proof of Lemma 3.) ◀

In [1, 2], we performed a careful rejection sampling procedure f in Procedure 2 so that, at each step of the algorithm, the output was distributed exactly as $D_{\mathcal{L}-\boldsymbol{t}, s/2^{i/2}}$ (up to some small statistical distance). In particular, we applied the rejection sampling procedure guaranteed by Theorem 12 to obtain independent vectors distributed as in Item 4, which yield independent Gaussians with a lower parameter when combined as in Procedure 1. But, Corollary 8 makes this unnecessary. Indeed, Corollary 8 shows that "any collection of vectors that can be found with any rejection sampling procedure can be found without it." The following meta-theorem makes this formal.

▶ **Theorem 9.** *For any (possibly randomized) rejection function f mapping lists of cosets modulo $2\mathcal{L}$ to a subset of indices (as in Procedure 2), let $\mathcal{A}$ be the algorithm defined in Procedure 2. Let $\mathcal{A}'$ be the same algorithm with f replaced by the trivial function that just outputs all indices (i.e., $\mathcal{A}'$ just repeatedly runs Procedure 1 with no rejection).*

Then, for any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift vector $\boldsymbol{t} \in \mathbb{R}^n$, parameter $s > 0$, if $\mathcal{A}$ and $\mathcal{A}'$ are each called on input $\ell \geq 1$ and a list of $M \geq 2$ independent samples from $D_{\mathcal{L}-\boldsymbol{t}, s}$, the resulting output distributions will be mixtures of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter $s/2^{\ell/2}$. Furthermore, the distribution corresponding to $\mathcal{A}'$ will dominate the distribution corresponding to $\mathcal{A}$. In particular, for any finite set $S \subset \mathcal{L} - \boldsymbol{t}$,

$$\Pr_{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M \sim D_{\mathcal{L}-\boldsymbol{t}, s}}[S \subseteq \mathcal{A}(\ell, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)] \leq \Pr_{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M \sim D_{\mathcal{L}-\boldsymbol{t}, s}}[S \subseteq \mathcal{A}'(\ell, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)] \ .$$

Proof. Notice that, since f only acts on the cosets of the $\boldsymbol{X}_i$, f "preserves mixtures of independent Gaussians." I.e., if $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{M'})$ is some mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter $s' > 0$ and $(j_1, \ldots, j_m) \leftarrow f(2\mathcal{L} + \boldsymbol{X}_1, \ldots, 2\mathcal{L} + \boldsymbol{X}_{M'})$, then $(\boldsymbol{X}_{j_1}, \ldots, \boldsymbol{X}_{j_m})$ is also a mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s'. (Notice that this would *not* be true if f acted on vectors, rather than cosets.) Similarly, by Item 2, Procedure 1 maps mixtures of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter s' to mixtures with parameter $s'/\sqrt{2}$. It follows that for both $\mathcal{A}$ and $\mathcal{A}'$, after the ith step of the algorithm, the list of vectors is a mixture of Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter $s/2^{i/2}$. And, the same holds after the application of f in algorithm $\mathcal{A}$. Therefore, the only question is the coset distributions.

By Fact 5, we see that $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$ dominates $(\boldsymbol{X}_{j_1}, \ldots, \boldsymbol{X}_{j_m})$. Therefore, by Item 3, the distribution of vectors corresponding to $\mathcal{A}'$ dominates the distribution of $\mathcal{A}$ after the first step. If we assume for induction that, after the $(i-1)$st step, the distribution of vectors corresponding to $\mathcal{A}'$ dominates the distribution corresponding to $\mathcal{A}$, then the exact same argument together with another application of Fact 5 shows that the same holds after step i. The result follows. ◀

Theorem 9, together with the corresponding algorithms in [1, 2], immediately implies Theorems 1 and 2. For completeness, we give more direct proofs of these theorems in the appendix, more-or-less recreating the corresponding proofs in [1, 2].

Acknowledgments. We thank Oded Regev and Daniel Dadush for many helpful discussions.

References

1 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n time via discrete Gaussian sampling. In *STOC*, 2015.

2 Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the Closest Vector Problem in 2^n time— The discrete Gaussian strikes again! In *FOCS*, 2015.

3 Miklós Ajtai. The shortest vector problem in L_2 is *NP*-hard for randomized reductions (extended abstract). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 10–19. ACM, 1998. doi:10.1145/276698.276705.

4 Miklós Ajtai. Generating hard instances of lattice problems. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 1–32. Dept. Math., Seconda Univ. Napoli, Caserta, 2004. Preliminary version in STOC'96.

5 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 601–610. ACM, 2001. doi:10.1145/380752.380857.

6 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *CCC*, pages 41–45, 2002.

7 Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange — A new hope. In *USENIX Security Symposium*, 2016.

8 László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. doi:10.1007/BF02579403.

9 Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993. doi:10.1007/BF01445125.

10 Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, 2016.

11 Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *CCS*, 2016.

12 Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584. ACM, 2013. doi:10.1145/2488608.2488680.

13 R. de Buda. Some optimal codes have structure. *Selected Areas in Communications, IEEE Journal on*, 7(6):893–899, Aug 1989.

14 Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

15 Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC*, 2008.

16 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

17 Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Pierre Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inf. Process. Lett.*, 71(2):55–61, 1999. doi:10.1016/S0020-0190(99)00083-6.

18 Ishay Haviv and Oded Regev. Tensor-based hardness of the Shortest Vector Problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

19 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

20 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. doi:10.1287/moor.12.3.415.

21 Subhash Khot. Hardness of approximating the Shortest Vector Problem in lattices. *Journal of the ACM*, 52(5):789–808, 2005. Preliminary version in FOCS'04.

22 Philip Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, 2000.

23 A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

24 Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001. Preliminary version in FOCS 1998.

25 Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the Shortest Vector Problem. In *SODA*, 2010.

26 Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

27 Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In *Eurocrypt*, 2016.

28 NIST post-quantum standardization call for proposals. http://csrc.nist.gov/groups/ST/post-quantum-crypto/cfp-announce-dec2016.html, 2016. Accessed: 2017-04-02.

29 Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

30 Xavier Pujol and Damien Stehlé. Solving the Shortest Lattice Vector Problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

31 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. doi:10.1145/1568318.1568324.

32 Oded Regev and Noah Stephens-Davidowitz. An inequality for Gaussians on lattices. *SIDMA*, 2017.

33 Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987. doi:10.1016/0304-3975(87)90064-8.

34 Adi Shamir. A polynomial-time algorithm for breaking the basic merkle-hellman cryptosystem. *IEEE Trans. Information Theory*, 30(5):699–704, 1984. doi:10.1109/TIT.1984.1056964.

35 Noah Stephens-Davidowitz. *On the Gaussian Measure Over Lattices*. PhD thesis, New York University, 2017.

36 Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, University of Amsterdam, Department of Mathematics, Netherlands, 1981. Technical Report 8104.

A Additional preliminaries

We will need some additional preliminaries. We write

$$\lambda_1(\mathcal{L}) := \min_{\boldsymbol{y} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\boldsymbol{y}\|$$

for the length of the shortest non-zero vector in the lattice. And, for a target vector $\boldsymbol{t} \in \mathbb{R}^n$, we write

$$\operatorname{dist}(\boldsymbol{t}, \mathcal{L}) := \min_{\boldsymbol{y} \in \mathcal{L}} \|\boldsymbol{y} - \boldsymbol{t}\|$$

for the distance from $\boldsymbol{t}$ to the lattice. Notice that this is the same as the length of the shortest vector in $\mathcal{L} - \boldsymbol{t}$.

A.1 Some known algorithms

We will need the famous result of Lenstra, Lenstra, and Lovász [23].

▶ **Theorem 10** ([23]). *There is an efficient algorithm that take as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and outputs $\tilde{\lambda} > 0$ with*

$$\lambda_1(\mathcal{L}) \leq \tilde{d} \leq 2^{n/2} \lambda_1(\mathcal{L}) \ .$$

We will also need the following celebrated result due to Babai [8].

▶ **Theorem 11** ([8]). *There is an efficient algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target $\boldsymbol{t} \in \mathbb{R}^n$ and outputs $\tilde{d} > 0$ with*

$$\operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \leq \tilde{d} \leq 2^{n/2} \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \ .$$

A.2 The distribution of disjoint pairs

Recall that Procedure 1 takes the T_i elements from the ith coset and converts them into $\lfloor T_i/2 \rfloor$ disjoint pairs. Therefore, for a list $\mathcal{M} := (X_1, \ldots, X_M) \in S^*$ over some finite set S, we write $\lfloor \mathcal{M}/2 \rfloor$ for the random variable obtained as in Procedure 1. I.e., up to ordering (which does not concern us), $\lfloor \mathcal{M}/2 \rfloor := (X'_1, \ldots, X'_{M'}) \in (S \times S)^*$ is defined by

$$|\{j \ : \ X'_j = (s, s)\}| = \lfloor |\{j \ : \ X_j = s\}|/2 \rfloor$$

for each $s \in S$.

▶ **Theorem 12** ([1, Theorem 3.3]). *For any probabilities, $p_1, \ldots, p_N \in [0,1]$ with $\sum p_i = 1$, integer M, and $\kappa \geq \Omega(\log M)$ (the confidence parameter) with $M \geq 10\kappa^2/p_{\mathsf{max}}$, let $\mathcal{M} = (X_1, \ldots, X_M) \in \{1, \ldots, N\}^M$ be the distribution obtained by sampling each X_j independently from the distribution that assigns to element i probability p_i. Then, there exists a rejection sampling procedure that, up to statistical distance $\exp(-\Omega(\kappa))$, maps $\mathcal{M}$ to the distribution $\mathcal{M}' := (X'_1, \ldots, X'_M) \in \{(1,1), \ldots, (N,N)\}^{M'}$ obtained by sampling each pair X_j independently from the distribution that assigns to the pair (i,i) probability p_i^2/p_{col}, where*

$$M' := \left\lceil M \cdot \frac{p_{\mathsf{col}}}{32\kappa p_{\mathsf{max}}} \right\rceil ,$$

$p_{\mathsf{max}} := \max p_i$, and $p_{\mathsf{col}} := \sum p_i^2$.

▶ **Corollary 13.** *For any probabilities, $p_1, \ldots, p_N \in [0,1]$ with $\sum p_i = 1$, integer M, and $\kappa \geq \Omega(\log M)$ (the confidence parameter) with $M \geq 10\kappa^2/p_{\mathsf{max}}$, let $\mathcal{M} := (X_1, \ldots, X_M) \in \{1, \ldots, N\}^M$ be the distribution obtained by sampling each X_j independently from the distribution that assigns to element i probability p_i. Let $\mathcal{M}' := (X'_1, \ldots, X'_{M'}) \in \{(1,1), \ldots, (N,N)\}^{M'}$ be the distribution obtained by sampling each pair X'_j independently from the distribution that assigns to the pair (i,i) probability p_i^2/p_{col}, where*

$$M' := \left\lceil M \cdot \frac{p_{\mathsf{col}}}{32\kappa p_{\mathsf{max}}} \right\rceil ,$$

$p_{\mathsf{max}} := \max p_i$, and $p_{\mathsf{col}} := \sum p_i^2$. Then, $\mathcal{M}$ dominates $\mathcal{M}'$.

A.3 Additional facts about the discrete Gaussian

We will also need some additional facts about the discrete Gaussian.

▶ **Lemma 14** ([9]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, parameter $s \geq 1$, and shift $\boldsymbol{t} \in \mathbb{R}^n$, $\rho_s(\mathcal{L} - \boldsymbol{t}) \leq s^n \rho(\mathcal{L})$.*

The following theorem shows that, if the parameter s is appropriately small, then $D_{\mathcal{L}-\boldsymbol{t},s} + \boldsymbol{t}$ will be an approximate closest vector to $\boldsymbol{t}$, with approximation factor roughly $1 + \sqrt{n}s/\operatorname{dist}(\boldsymbol{t}, \mathcal{L})$. (This is a basic consequence of Banaszczyk's celebrated theorem [9].)

▶ **Proposition 15** ([35, Corollary 1.3.11], see also [2, Corollary 2.8]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, parameter $s > 0$, shift $\boldsymbol{t} \in \mathbb{R}^n$, and radius $r > \sqrt{n/(2\pi)} \cdot s$, with $r > \operatorname{dist}(\boldsymbol{t}, \mathcal{L})$ and*

$$r^2 > \operatorname{dist}(\boldsymbol{t}, \mathcal{L})^2 + \frac{ns^2}{\pi} \cdot \log(2\pi \operatorname{dist}(\boldsymbol{t}, \mathcal{L})^2/(ns^2)) ,$$

we have

$$\Pr_{\boldsymbol{X} \sim D_{\mathcal{L}-\boldsymbol{t},s}}[\|\boldsymbol{X}\| > r] < (2e)^{n/2+1} \exp(-\pi y^2/2) ,$$

where $y := \sqrt{r^2 - \operatorname{dist}(\boldsymbol{t}, \mathcal{L})^2}/s$.

The next theorem shows that exponentially many samples from $D_{\mathcal{L},s}$ with $s \approx \lambda_1(\mathcal{L})/\sqrt{n}$ is sufficient to find a shortest non-zero lattice vector.

▶ **Proposition 16** ([1, Proposition 4.3]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, and parameter*

$$s := \sqrt{2^{0.198} \pi e/n} \cdot \lambda_1(\mathcal{L}) ,$$

we have

$$\Pr_{\boldsymbol{X} \sim D_{\mathcal{L},s}}[\|\boldsymbol{X}\| = \lambda_1(\mathcal{L})] \geq 1.38^{-n-o(n)} .$$

The next corollary follows immediately from Proposition 16 and Lemma 14.

▶ **Corollary 17.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, and parameter*

$$\sqrt{2^{0.198}\pi e/n} \cdot \lambda_1(\mathcal{L}) \le s \le 1.01 \cdot \sqrt{2^{0.198}\pi e/n} \cdot \lambda_1(\mathcal{L}) \ ,$$

we have

$$\Pr_{\boldsymbol{X} \sim D_{\mathcal{L},s}}[\|\boldsymbol{X}\| = \lambda_1(\mathcal{L})] \ge 1.4^{-n-o(n)} \ .$$

We will also need the following result from [2], which is an immediate consequence of the main identity in [32]. (See also [35].)

▶ **Lemma 18** ([2, Corollary 3.3])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, and parameter $s > 0$, we have*

$$\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})^2 \le \rho_{s/\sqrt{2}}(\mathcal{L}) \max_{\boldsymbol{c} \in \mathcal{L}} \rho_{s/\sqrt{2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t}) \ .$$

From this, we derive the following rather technical-looking inequality, which is implicit in [2]. (This inequality comes up naturally in the proof of Corollary 21. We separate it out here to make that proof cleaner.)

▶ **Corollary 19.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, parameter $s > 0$, and integer $\ell \ge 0$, we have*

$$\prod_{i=0}^{\ell-1} \frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L} - \boldsymbol{t})\rho_{s/2^{(i+1)/2}}(\mathcal{L})}{\rho_{s/2^{i/2}}(\mathcal{L} - \boldsymbol{t}) \cdot \max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}$$
$$\ge \frac{\rho_{s/2^{\ell/2}}(\mathcal{L} - \boldsymbol{t})}{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{\ell/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})} \cdot \frac{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}{\rho_s(\mathcal{L} - \boldsymbol{t})} \ .$$

Proof. From Lemma 18, we see that for all i,

$$\frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L})}{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})} \ge \frac{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{(i+1)/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})} \ .$$

Therefore, the product in the statement of the corollary is at least

$$\prod_{i=0}^{\ell-1} \frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L} - \boldsymbol{t}) \cdot \max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}{\rho_{s/2^{i/2}}(\mathcal{L} - \boldsymbol{t}) \cdot \max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{(i+1)/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}$$
$$= \frac{\rho_{s/2^{\ell/2}}(\mathcal{L} - \boldsymbol{t})}{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_{s/2^{\ell/2}}(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})} \cdot \frac{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})}{\rho_s(\mathcal{L} - \boldsymbol{t})} \ ,$$

where we have used the fact that this is a telescoping product. ◀

B Running Procedure 1 on Gaussian input

▶ **Theorem 20.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, parameter $s > 0$, integer M, and confidence parameter $\kappa \ge \Omega(\log M)$, if $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M$ are sampled independently from $D_{\mathcal{L}-\boldsymbol{t},s}$ with*

$$M \ge 10\kappa^2 \cdot \frac{\rho_s(\mathcal{L} - \boldsymbol{t})}{\max_{\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L} + \boldsymbol{c} - \boldsymbol{t})} \ ,$$

then the output of Procedure 1 applied to the $\boldsymbol{X}_i$ will be a mixture of independent Gaussians with parameter $s/\sqrt{2}$ that dominates the distribution of

$$M' := \left\lceil \frac{M}{32\kappa} \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L}-\boldsymbol{t}) \cdot \rho_{s/\sqrt{2}}(\mathcal{L})}{\rho_s(\mathcal{L}-\boldsymbol{t}) \cdot \max_{\boldsymbol{d}\in\mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L}+\boldsymbol{d}-\boldsymbol{t})} \right\rceil$$

independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/\sqrt{2}}$, up to statistical distance $\exp(-\Omega(\kappa))$.

Proof. By Item 2 of Corollary 8, the resulting distribution will in fact be a mixture of independent Gaussians over $\mathcal{L}-\boldsymbol{t}$ with parameter $s/\sqrt{2}$. Notice that, if $\mathcal{M}$ is the coset distribution of $(\boldsymbol{X}_1,\dots,\boldsymbol{X}_M)$, then Procedure 1 first maps the $\boldsymbol{X}_i$ into the mixture of independent Gaussians over $\mathcal{L}-\boldsymbol{t}$ with parameter s and coset distribution $\lfloor \mathcal{M}/2 \rfloor$ and then takes the averages of the corresponding pairs of these vectors.

We wish to apply Corollary 13 over the coset distribution, with the probabilities $p_i := p_{2\mathcal{L}+\boldsymbol{c}}$ taken to be the weights of the cosets in the original distribution discrete Gaussian,

$$p_{2\mathcal{L}+\boldsymbol{c}} := \frac{\rho_s(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})}{\rho_s(\mathcal{L}-\boldsymbol{t})} \ .$$

Notice that, by Lemma 3,

$$M' = \left\lceil M \cdot \frac{p_{\mathsf{col}}}{32\kappa p_{\mathsf{max}}} \right\rceil \ ,$$

which is exactly what is needed to apply Corollary 13. By the corollary, up to statistical distance $\exp(-\Omega(\kappa))$ this distribution dominates the mixture of independent Gaussians over $\mathcal{L}-\boldsymbol{t}$ with parameter s whose coset distribution is given by $\boldsymbol{c}_{2k-1} = \boldsymbol{c}_{2k}$ for $1 \le k \le M'$, with the odd-indexed cosets $\boldsymbol{c}_{2k-1}$ sampled independently from the distribution that assigns to coset $\boldsymbol{c} \in \mathcal{L}/(2\mathcal{L})$ probability

$$\frac{p_i}{p_{\mathsf{col}}} = \frac{\rho_s(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})^2}{\sum_{\boldsymbol{d}\in\mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L}+\boldsymbol{d}-\boldsymbol{t})^2} \ .$$

Notice that this "squared" distribution" (so-called because the cosets are given weight proportional to their square) is simply M' independent copies of the distribution from Item 4 of Corollary 8. So, if we run Procedure 1 on this "squared" distribution, the output will be exactly M' independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/\sqrt{2}}$.

Finally, by Fact 5, we see that, since the actual pairs dominate these "squared" pairs (up to statistical distance $\exp(-\Omega(\kappa))$), the output must dominate M' independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/\sqrt{2}}$. ◀

▶ **Corollary 21.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, parameter $s > 0$, integer $M \ge 2$, and confidence parameter $\kappa \ge \Omega(\log M)$, if $\boldsymbol{X}_1,\dots,\boldsymbol{X}_M$ are sampled independently from $D_{\mathcal{L}-\boldsymbol{t},s}$ with*

$$M \ge (10\kappa)^{2\ell} \cdot \frac{\rho_s(\mathcal{L}-\boldsymbol{t})}{\max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_s(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})} \ ,$$

and we apply Procedure 1 repeatedly to the $\boldsymbol{X}_i$ a total of $\ell \ge 1$ times, the result will be a mixture of independent Gaussians with parameter $s/2^{\ell/2}$ that dominates the distribution of

$$M' := \left\lceil \frac{M}{(32\kappa)^\ell} \cdot \prod_{i=0}^{\ell-1} \frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L}-\boldsymbol{t})\rho_{s/2^{(i+1)/2}}(\mathcal{L})}{\rho_{s/2^{i/2}}(\mathcal{L}-\boldsymbol{t}) \cdot \max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})} \right\rceil$$

independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/2^{\ell/2}}$, up to statistical distance $\ell \exp(-\Omega(\kappa))$.

Proof. By Item 2 of Corollary 8, the output will in fact be a mixture of independent Gaussians over $\mathcal{L} - \boldsymbol{t}$ with parameter $s/2^{\ell/2}$. The only question is what the coset distribution is.

To show that the coset distribution is as claimed, the idea is to simply apply Theorem 20 ℓ times. In particular, we prove the result via induction on ℓ. When $\ell = 1$, this is exactly Theorem 20. For $\ell > 1$, we assume the statement is true for $\ell - 1$. In particular, before applying Procedure 1 the ℓth time, we have a mixture of independent Gaussians with parameter $s/2^{\ell/2}$ that dominates

$$\widehat{M} := \left\lceil \frac{M}{(32\kappa)^\ell} \cdot \prod_{i=0}^{\ell-2} \frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L}-\boldsymbol{t})\rho_{s/2^{(i+1)/2}}(\mathcal{L})}{\rho_{s/2^{i/2}}(\mathcal{L}-\boldsymbol{t}) \cdot \max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})} \right\rceil$$
$$\geq 10\kappa^2 \cdot \frac{\rho_s(\mathcal{L}-\boldsymbol{t})}{\max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_s(\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})} \cdot \prod_{i=0}^{\ell-2} \frac{\rho_{s/2^{(i+1)/2}}(\mathcal{L}-\boldsymbol{t})\rho_{s/2^{(i+1)/2}}(\mathcal{L})}{\rho_{s/2^{i/2}}(\mathcal{L}-\boldsymbol{t}) \cdot \max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_{s/2^{i/2}}(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})}$$

independent Gaussians up to statistical distance $(\ell-1)\exp(-\Omega(\kappa))$.

By Fact 5, it suffices to prove that the output of Procedure 1 on these $\widehat{M}$ samples dominates M' independent samples from $D_{\mathcal{L}-\boldsymbol{t},s/2^{\ell/2}}$ up to statistical distance $\exp(-\Omega(\kappa))$. Indeed, this is exactly what Theorem 20 says, provided that

$$\widehat{M} \geq 10\kappa^2 \cdot \frac{\rho_{s/2^{(\ell-1)/2}}(\mathcal{L}-\boldsymbol{t})}{\max_{\boldsymbol{c}\in\mathcal{L}/(2\mathcal{L})} \rho_{s/2^{(\ell-1)/2}}(2\mathcal{L}+\boldsymbol{c}-\boldsymbol{t})} .$$

And, this inequality follows immediately from Corollary 19 together with the assumed lower bound on $\widehat{M}$. ◀

C The initial distribution

The following theorem was proven by Ajtai, Kumar, and Sivakumar [5], building on work of Schnorr [33].

▶ **Theorem 22** ([33, 5]). *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $u \geq 2$ and outputs an $u^{n/y}$-reduced basis of $\mathcal{L}$ in time $\exp(O(u)) \cdot \mathrm{poly}(n)$, where we say that a basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of a lattice $\mathcal{L}$ is γ-reduced for some $\gamma \geq 1$ if*
1. *$\|\boldsymbol{b}_1\| \leq \gamma \cdot \lambda_1(\mathcal{L})$; and*
2. *$\pi_{\{\boldsymbol{b}_1\}^\perp}(\boldsymbol{b}_2), \ldots, \pi_{\{\boldsymbol{b}_1\}^\perp}(\boldsymbol{b}_n)$ is a γ-reduced basis of $\pi_{\{\boldsymbol{b}_1\}^\perp}(\mathcal{L})$.*

This next theorem is originally due to [16], based on analysis of an algorithm originally studied by Klein [22]. We present a slightly stronger version due to [12] for convenience.

▶ **Theorem 23** ([12, Lemma 2.3]). *There is a probabilistic polynomial-time algorithm that takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $n \geq 2$, a shift $\boldsymbol{t} \in \mathbb{R}^n$, and $\hat{s} > C\sqrt{\log n} \cdot \|\widetilde{\mathbf{B}}\|$ and outputs a vector that is distributed exactly as $D_{\mathcal{L}-\boldsymbol{t},\hat{s}}$, where $\|\widetilde{\mathbf{B}}\| := \max\|\widetilde{\boldsymbol{b}}_i\|$.*

▶ **Proposition 24** ([2, Proposition 4.5]). *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$, shift $\boldsymbol{t} \in \mathbb{R}^n$, $r > 0$, and parameter $u \geq 2$, such that if*

$$r \geq u^{n/u}(1 + \sqrt{n}u^{n/u}) \cdot \mathrm{dist}(\boldsymbol{t}, \mathcal{L}) ,$$

then the output of the algorithm is $\boldsymbol{y} \in \mathcal{L}$ and a basis $\mathbf{B}'$ of a (possibly trivial) sublattice $\mathcal{L}' \subseteq \mathcal{L}$ such that all vectors from $\mathcal{L}-\boldsymbol{t}$ of length at most $r/u^{n/u} - \mathrm{dist}(\boldsymbol{t}, \mathcal{L})$ are also contained in $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$, and $\|\widetilde{\mathbf{B}}'\| \leq r$. The algorithm runs in time $\mathrm{poly}(n) \cdot 2^{O(u)}$.

Proof. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, $\boldsymbol{t} \in \mathbb{R}^n$, and $r > 0$, the algorithm behaves as follows. First, it calls the procedure from Theorem 22 to compute a $u^{n/u}$-HKZ basis $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of $\mathcal{L}$. Let $(\widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_n)$ be the corresponding Gram-Schmidt vectors. Let $k \geq 0$ be maximal such that $\|\widetilde{\boldsymbol{b}}_i\| \leq r$ for $1 \leq i \leq k$, and let $\mathbf{B}' = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k)$. Let $\pi_k = \pi_{\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}^\perp}$ and $\mathcal{M} = \pi_k(\mathcal{L})$. The algorithm then calls the procedure from Theorem 22 again with the same s and input $\pi_k(\boldsymbol{t})$ and $\mathcal{M}$, receiving as output $\boldsymbol{x} = \sum_{i=k+1}^{n} a_i \pi_k(\boldsymbol{b}_i)$ where $a_i \in \mathbb{Z}$, a $\sqrt{n}u^{n/u}$-approximate closest vector to $\pi_k(\boldsymbol{t})$ in $\mathcal{M}$. Finally, the algorithm returns $\boldsymbol{y} = -\sum_{i=k+1}^{n} a_i \boldsymbol{b}_i$ and $\mathbf{B}' = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k)$.

The running time is clear, as is the fact that $\|\widetilde{\mathbf{B}'}\| \leq r$. It remains to prove that $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ contains all sufficiently short vectors in $\mathcal{L} - \boldsymbol{t}$. If $k = n$, then $\mathcal{L}' = \mathcal{L}$ and $\boldsymbol{y}$ is irrelevant, so we may assume that $k < n$. Note that, since $\mathbf{B}$ is a $u^{n/u}$-HKZ basis, $\lambda_1(\mathcal{M}) \geq \|\widetilde{\boldsymbol{b}}_{k+1}\|/u^{n/u} > r/u^{n/u}$. In particular, $\lambda_1(\mathcal{M}) > (1 + \sqrt{n} \cdot u^{n/u}) \cdot \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \geq (1 + \sqrt{n} \cdot u^{n/u}) \cdot \operatorname{dist}(\pi_k(\boldsymbol{t}), \mathcal{M})$. So, there is a unique closest vector to $\pi_k(\boldsymbol{t})$ in $\mathcal{M}$, and by triangle inequality, the next closest vector is at distance greater than $\sqrt{n} \cdot u^{n/u} \operatorname{dist}(\pi_k(\boldsymbol{t}), \mathcal{M})$. Therefore, the call to the subprocedure from Theorem 22 will output the exact closest vector $\boldsymbol{x} \in \mathcal{M}$ to $\pi_k(\boldsymbol{t})$.

Let $\boldsymbol{w} \in \mathcal{L} \setminus (\mathcal{L}' - \boldsymbol{y})$ so that $\pi_k(\boldsymbol{w}) \neq \pi_k(-\boldsymbol{y}) = \boldsymbol{x}$. We need to show that $\boldsymbol{w} - \boldsymbol{t}$ is relatively long. Since $\mathbf{B}$ is a $s^{n/s}$-HKZ basis, it follows that

$$\|\pi_k(\boldsymbol{w}) - \boldsymbol{x}\| \geq \lambda_1(\mathcal{M}) > r/u^{n/u} \ .$$

Applying triangle inequality, we have

$$\|\boldsymbol{w} - \boldsymbol{t}\| \geq \|\pi_k(\boldsymbol{w}) - \pi_k(\boldsymbol{t})\| \geq \|\pi_k(\boldsymbol{w}) - \boldsymbol{x}\| - \|\boldsymbol{x} - \pi_k(\boldsymbol{t})\| > r/u^{n/u} - \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \ ,$$

as needed. ◀

▶ **Corollary 25** ([2, Corollary 4.6]). *There is an algorithm that takes as input a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $n \geq 2$, shift $\boldsymbol{t} \in \mathbb{R}^n$, $M \in \mathbb{N}$ (the desired number of output vectors), and parameters $u \geq 2$ and $\hat{s} > 0$ and outputs $\boldsymbol{y} \in \mathcal{L}$, a (possibly trivial) sublattice $\mathcal{L}' \subseteq \mathcal{L}$, and M vectors from $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ such that if*

$$\hat{s} \geq 10\sqrt{n \log n} \cdot u^{2n/u} \cdot \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \ ,$$

then the output vectors are distributed as M independent samples from $D_{\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}, \hat{s}}$, and $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ contains all vectors in $\mathcal{L} - \boldsymbol{t}$ of length at most $\hat{s}/(10u^{n/u}\sqrt{\log n})$. The algorithm runs in time $\operatorname{poly}(n) \cdot 2^{O(u)} + \operatorname{poly}(n) \cdot M$. (And, if $\boldsymbol{t} = \mathbf{0}$, then $\boldsymbol{y} = \mathbf{0}$.)

Proof. The algorithm first calls the procedure from Proposition 24 with input $\mathcal{L}$, $\boldsymbol{t}$, and

$$r := \frac{10\hat{s}}{\sqrt{\log n}} \geq u^{n/u}(1 + \sqrt{n}u^{n/u}) \cdot \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \ ,$$

receiving as output $\boldsymbol{y} \in \mathcal{L}$ and a basis $\mathbf{B}'$ of a sublattice $\mathcal{L}' \subset \mathcal{L}$. It then runs the algorithm from Theorem 23 M times with input $\mathcal{L}'$, $\boldsymbol{y} + \boldsymbol{t}$, and $\hat{s}$ and outputs the resulting vectors, $\boldsymbol{y}$, and $\mathcal{L}'$.

The running time is clear. By Proposition 24, $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ contains all vectors of length at most $r/u^{n/u} - \operatorname{dist}(\boldsymbol{t}, \mathcal{L}) \geq \hat{s}/(10u^{n/u}\sqrt{\log n})$ in $\mathcal{L} - \boldsymbol{t}$, and $\|\widetilde{\mathbf{B}}'\| \leq r \leq C\hat{s}/\sqrt{\log n}$. So, it follows from Theorem 23 that the output has the correct distribution. ◀

D Finishing the proof

▶ **Theorem 26** (SVP algorithm). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, the output of Procedure 3 on input $\mathcal{L}$ will be a shortest non-zero vector in $\mathcal{L}$ except with probability at most $\exp(-\Omega(n))$.*

Procedure 3: The final $2^{n+o(n)}$-time SVP algorithm. Here $M = 2^{n+\Theta(\log^2 n)}$, $u = \Theta(n)$, and $\ell = \Theta(\log n)$.

SVP $(\mathcal{L})$
Input : A lattice $\mathcal{L} \subset \mathbb{R}^n$
Output: A vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}\| = \lambda_1(\mathcal{L})$
Use the procedure from Thereom 10 to compute $\widehat{\lambda}$ with $\lambda_1(\mathcal{L}) \leq \widehat{\lambda} \leq 2^{n/2}\lambda_1(\mathcal{L})$.
for $i = 1, \ldots, 200n$ **do**
  Set $\mathcal{L}' \subseteq \mathcal{L}$ and $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M \in \mathcal{L}$ to be the output of Corollary 25 on input $\mathcal{L}$, $\boldsymbol{t} := \boldsymbol{0}$, u, and $s_i := 1.01^{-i} \cdot \widehat{\lambda}$.
  for $j = 1, \ldots, \ell$ **do**
    $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{M'}) \leftarrow$ Pair_and_Average$(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$
    $M \leftarrow M'$
  end
  $\boldsymbol{Y}_i \leftarrow \arg\min_{\boldsymbol{X}_j \neq \boldsymbol{0}} \|\boldsymbol{X}_j\|$.
end
Output $\arg\min \|\boldsymbol{Y}_i\|$.

Proof. The running time is clear. Let $\kappa = \Theta(n)$. Let i such that $s_i/2^{\ell/2}$ satisfies the inequality in Corollary 17. By Corollary 25, the $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$ corresponding to this i will be distributed exactly as $D_{\mathcal{L}', s_i}$ where $\mathcal{L}' \subseteq \mathcal{L}$ contains all vectors of length at most $\lambda_1(\mathcal{L})$. So, $\lambda_1(\mathcal{L}') = \lambda_1(\mathcal{L})$, and it suffices to argue that we will find a shortest vector in $\mathcal{L}'$. By Corollary 21, the output distribution $(X_1, \ldots, X_M)$ will be a mixture of independent Gaussians over $\mathcal{L}'$ with parameter $s_i/2^{\ell/2}$ that dominates the distribution of

$$M' = \left\lceil \frac{M}{(32\kappa)^\ell} \cdot \prod_{j=0}^{\ell-1} \frac{\rho_{s_i/2^{(j+1)/2}}(\mathcal{L}')^2}{\rho_{s_i/2^{j/2}}(\mathcal{L}') \cdot \rho_{s_i/2^{(j+2)/2}}(\mathcal{L}')} \right\rceil$$

independent samples from $D_{\mathcal{L}', s_i/2^{\ell/2}}$ up to statistical distance $\exp(-\Omega(\kappa))$, where we have applied Lemma 14 to show that the coset with maximal mass is the central coset. Noting that this product is telescoping, we have

$$M' = \left\lceil \frac{M}{(32\kappa)^\ell} \cdot \frac{\rho_{s_i/\sqrt{2}}(\mathcal{L}')}{\rho_{s_i}(\mathcal{L}')} \cdot \frac{\rho_{s_i/2^{\ell/2}}(\mathcal{L}')}{\rho_{s_i/2^{(\ell+1)/2}}(\mathcal{L}')} \right\rceil \geq 2^{n/2} \,,$$

where we have applied Lemma 14. The result then follows from Corollary 17, together with the fact that $\sqrt{2} > 1.4$. ◀

Procedure 4: The final $2^{n+o(n)}$-time SVP algorithm. Here $M = 2^{n+\Theta(n/\log n)}$, $u = \Theta(n)$, and $\ell = \Theta(n/\log^2 n)$.

CVP $(\mathcal{L}, \boldsymbol{t})$
Input : A lattice $\mathcal{L} \subset \mathbb{R}^n$ and target $\boldsymbol{t} \in \mathbb{R}^n$
Output: A vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y} - \boldsymbol{t}\| \leq (1 + 2^{-n/\log^2 n}) \cdot \text{dist}(\boldsymbol{t}, \mathcal{L})$
Use the procedure from Thereom 11 to compute $\hat{d}$ with
$\text{dist}(\mathcal{L}, \boldsymbol{t}) \leq \hat{d} \leq 2^{n/2} \text{dist}(\boldsymbol{t}, \mathcal{L})$.
for $i = 1, \ldots, n$ **do**
  Set $\mathcal{L}' \subseteq \mathcal{L}$, $\boldsymbol{y} \in \mathcal{L}$, and $X_1, \ldots, X_M \in \mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ to be the output of Corollary 25 on input $\mathcal{L}$, $\boldsymbol{t}$, u, and $s_i := 20n^2 \cdot 2^{-i} \cdot \hat{d}$.
  for $j = 1, \ldots, \ell$ **do**
    $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{M'}) \leftarrow \text{Pair\_and\_Average}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$
    $M \leftarrow M'$
  end
  $\boldsymbol{Y}_i \leftarrow \arg\min_{\boldsymbol{X}_j} \|\boldsymbol{X}_j\|$.
end
Output $\boldsymbol{t} + \arg\min \|\boldsymbol{Y}_i\|$.

▶ **Theorem 27** (CVP algorithm). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\boldsymbol{t} \in \mathbb{R}^n$, the output of Procedure 4 on input $\mathcal{L}$ and $\boldsymbol{t}$ will a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y} - \boldsymbol{t}\| \leq (1 + \exp(-\Omega(n/\log^2 n))) \cdot \text{dist}(\boldsymbol{t}, \mathcal{L})$, except with probability at most $\exp(-\Omega(n))$.*[4]

Proof. The running time is clear. Let $\kappa = \Theta(n)$. Let i such that

$$10\sqrt{n \log n} \cdot u^{2n/u} \cdot \text{dist}(\boldsymbol{t}, \mathcal{L}) \leq s_i \leq 20\sqrt{n \log n} \cdot u^{2n/u} \cdot \text{dist}(\boldsymbol{t}, \mathcal{L}) \ .$$

By Corollary 25, the $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M)$ corresponding to this i will be distributed exactly as $D_{\mathcal{L}'-\boldsymbol{y}-\boldsymbol{t}, s_i}$ where $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t} \subseteq \mathcal{L} - \boldsymbol{t}$ contains all vectors of length at most $\text{dist}(\boldsymbol{t}, \mathcal{L})$. So, it suffices to argue that we will find a $(1 + 2^{-n/\log^2 n})$-approximate shortest vector in $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$. By Corollary 21, the output distribution $(X_1, \ldots, X_M)$ will be a mixture of independent Gaussians over $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$ with parameter $s_i/2^{\ell/2}$ that dominates the distribution of

$$M' = \left\lceil \frac{M}{(32\kappa)^\ell} \cdot \prod_{j=0}^{\ell-1} \frac{\rho_{s_i/2^{(j+1)/2}}(\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}) \rho_{s_i/2^{(j+1)/2}}(\mathcal{L})}{\rho_{s_i/2^{j/2}}(\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}) \cdot \max_{\boldsymbol{c} \in \mathcal{L}'/(2\mathcal{L}')} \rho_{s_i/2^{j/2}}(2\mathcal{L}' + \boldsymbol{c} - \boldsymbol{y} - \boldsymbol{t})} \right\rceil \geq 1$$

independent samples from $D_{\mathcal{L}'-\boldsymbol{y}-\boldsymbol{t}, s_i/2^{\ell/2}}$ up to statistical distance $\exp(-\Omega(\kappa))$, where we have applied Corollary 19.

Notice that $s_i/2^{\ell/2} < \exp(-\Omega(n/\log^2 n)) \, \text{dist}(\boldsymbol{t}, \mathcal{L})$. The result then follows from Proposition 15, which says that, except with probability $\exp(-\Omega(n))$ a sample from $D_{\mathcal{L}'-\boldsymbol{y}-\boldsymbol{t}, s_i/2^{\ell/2}}$ will be a $(1 + \exp(-\Omega(n/\log^2 n)))$-approximate shortest vector in $\mathcal{L}' - \boldsymbol{y} - \boldsymbol{t}$. ◀

[4] It is immediate from the proof that this result can be extended to work for any approximation factor γ with $\gamma > 1 + \exp(-o(n/\log n))$, by taking $\ell = o(n/\log n)$ and $M = 2^{n+o(n)}$ to be slightly larger.