Lattice cryptography: from cryptanalysis to New Foundations

Woerden, W.P.J. van

CHAPTER 2

Preliminaries

2.1 Notation

The notation $x := y$ means that x is defined to be equal to y. The set of all integer, rational and real numbers is denoted by $\mathbb{Z}, \mathbb{Q}$ and $\mathbb{R}$ respectively. For a real number $x \in \mathbb{R}$ we write $\lfloor x \rceil$ for the unique integer such that $x - \lfloor x \rceil \in [-\frac{1}{2}, \frac{1}{2})$. All vectors $\mathbf{x}, \mathbf{y}$ and matrices $\mathbf{A}, \mathbf{B}$ are denoted by bold lower and upper case letters respectively. All vectors are column-vectors and we write $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ for a matrix where the i-th column vector is $\mathbf{b}_i$. For matrices $\mathbf{A}, \mathbf{B}$, we denote the horizontally stacked matrix $\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix}$, by $[\mathbf{A}; \mathbf{B}]$.

We denote the standard Euclidean inner product of two vectors $\mathbf{v} = (v_1, \ldots, v_d), \mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$ by $\langle \mathbf{v}, \mathbf{w} \rangle := \sum_{i=1}^{d} v_i w_i$, and the norm of a vector $\mathbf{v}$ by $\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$. For a non-empty discrete set $S \subset \mathbb{R}^d$ and a target $\mathbf{t} \in \mathbb{R}^d$ we denote the distance of $\mathbf{t}$ to the set by $\mathrm{dist}(S, \mathbf{t}) := \min_{\mathbf{x} \in S} \|\mathbf{t} - \mathbf{x}\|$. We write $\mathcal{S}^d \subset \mathbb{R}^{d+1}$ and $\mathcal{B}^d \subset \mathbb{R}^d$ for the Euclidean d-sphere and d-ball of radius 1 respectively. For a and b integers with $a \leq b$, we denote by $[\![a, b]\!]$ the set of integers $\{a, a+1, \ldots, b\}$. For a finite set $\mathcal{E}$, we will denote by $|\mathcal{E}|$ its cardinality. We denote the n-dimensional volume of a measurable n-dimensional

body $S \subset \mathbb{R}^d$ by $\mathrm{vol}_n(S)$. If the supposed dimension is clear from the context we just denote $\mathrm{vol}(S)$. For a function $f : A \to B$, and a subset $S \subset A$ we denote $f(S) := \sum_{x \in S} f(x)$.

2.2 Lattices and their properties

2.2.1 Lattice

Definition 1 (Lattice). *A lattice $\mathcal{L} \subset \mathbb{R}^d$ is a discrete subgroup of the vector space $\mathbb{R}^d$; i.e., $\mathcal{L} \subset \mathbb{R}^d$ is non-empty and*

- *is an additive group: for all $\mathbf{v}, \mathbf{w} \in \mathcal{L}$ we have $\mathbf{v} \pm \mathbf{w} \in \mathcal{L}$; and*
- *is discrete: the exists some positive radius $r > 0$ such that all pairs $\mathbf{v}, \mathbf{w} \in \mathcal{L}$ of distinct lattice vectors we have $\|\mathbf{v} - \mathbf{w}\| \geq r$.*

A simple example of a lattice is the orthogonal lattice $\mathbb{Z}^d$, or any subgroup $\mathcal{L} \subset \mathbb{Z}^d$ of it.

Definition 2 (Lattice Basis). *Let $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1} \in \mathbb{R}^d$ be $\mathbb{R}$-linearly independent vectors, we call the matrix $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ a (lattice) basis[1], and we define the lattice $\mathcal{L}(\mathbf{B})$ generated by the $\mathbb{Z}$-span of $\mathbf{B}$ as*

$$\mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^n = \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i : x_i \in \mathbb{Z}^n \right\}.$$

Lemma 3. *For every $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$, the $\mathbb{Z}$-span $\mathcal{L}(\mathbf{B})$ of $\mathbf{B}$ is a lattice. Conversely, every lattice has a basis.*

Definition 4 (Rank). *The rank $\mathrm{rk}(\mathcal{L})$ of a lattice $\mathcal{L}$ is defined as the $\mathbb{R}$-rank of its $\mathbb{R}$-span $\mathrm{rk}(\mathcal{L}) := \mathrm{rk}(\mathrm{span}\, \mathcal{L})$.*

The rank of a lattice coincides exactly with the number of vectors in any basis of it. Unless stated otherwise any lattice mentioned in the remainder of this thesis is assumed to be of rank at least 1. We call a lattice $\mathcal{L} \subset \mathbb{R}^d$ full rank if its rank equals the dimension d of the vector space $\mathbb{R}^d$. Given a basis $\mathbf{B}$ of a lattice of rank n, all the other bases are of the form $\mathbf{B} \cdot \mathbf{U}$ for a *unimodular* $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})$, i.e.,

[1] We start counting at 0 for notational purposes later.

an integer matrix $\mathbf{U} \in \mathbb{Z}^{n\times n}$ with $\det(\mathbf{U}) = \pm 1$. For rank $n \geq 2$ each lattice has an infinite number of distinct bases.

Definition 5 (Primitive). *For a rank n lattice $\mathcal{L} \subset \mathbb{R}^d$, we call a set of vectors $\mathbf{v}_0, \ldots, \mathbf{v}_k \in \mathcal{L}$ primitive if they can be extended to a full basis $[\mathbf{v}_0, \ldots, \mathbf{v}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{n-1}]$ of $\mathcal{L}$.*

A single vector $\mathbf{v} \in \mathcal{L}$ is primitive if it cannot be scaled down to another lattice vector, i.e. if there exists no $\lambda > 1$ such that $\mathbf{v}/\lambda \in \mathcal{L}$.

Another way to capture the geometry of a lattice is by a Gram matrix.

Definition 6 (Gram). *For a lattice basis $\mathbf{B}$ of rank n, we define its Gram matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ as the (positive definite) matrix consisting of all pairwise inner products*

$$\mathbf{G} := \mathbf{B}^\top \mathbf{B} = (\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{0 \leq i,j < n}.$$

We denote $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{G}} := \mathbf{x}^\top \mathbf{G} \mathbf{y}$, and $\|\mathbf{x}\|_{\mathbf{G}} := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbf{G}}}$.

For any basis $\mathbf{B}$, its gram matrix $\mathbf{G} = \mathbf{B}^\top \mathbf{B}$, and any two vectors $\mathbf{v} = \mathbf{B}\mathbf{x}, \mathbf{w} = \mathbf{B}\mathbf{y} \in \operatorname{span}(\mathcal{L})$, we have

$$\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{B}\mathbf{x}, \mathbf{B}\mathbf{y} \rangle = \mathbf{x}^\top (\mathbf{B}^\top \mathbf{B}) \mathbf{y} = \mathbf{x}^\top \mathbf{G} \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{G}}.$$

The Gram matrix stores all geometric information, but not the particular embedding of the lattice into $\mathbb{R}^d$, e.g., for any orthonormal transformation $\mathbf{O} \in \mathcal{O}_d(\mathbb{R})$, the bases $\mathbf{B}$ and $\mathbf{OB}$ have the same Gram matrix.

2.2.2 Properties

We discuss some important properties of a lattice. Because a lattice $\mathcal{L}$ is a normal subgroup of $\operatorname{span}(\mathcal{L})$, we can consider their quotient group.

Definition 7 (Lattice Torus). *For a lattice $\mathcal{L}$ we define its lattice torus $\mathbb{T}(\mathcal{L})$ as*

$$\mathbb{T}(\mathcal{L}) := (\operatorname{span} \mathcal{L})/\mathcal{L}.$$

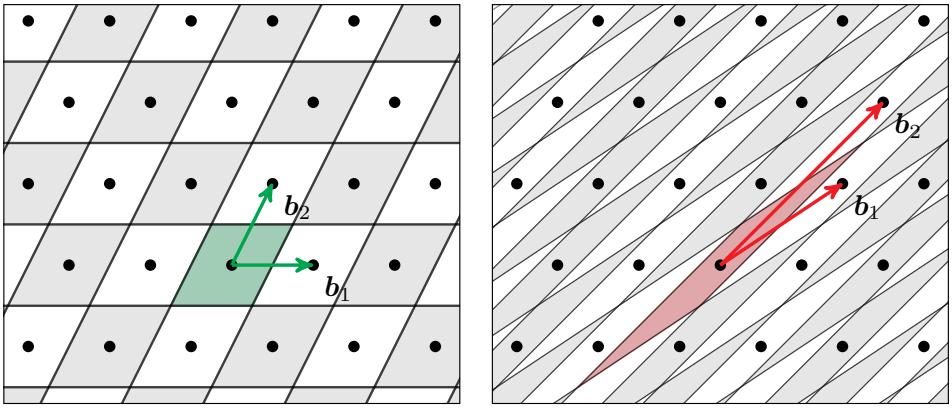

Figure 2.1: The fundamental domain $\mathcal{P}(\mathbf{B})$ for two different bases. They both have the same volume $\mathrm{vol}(\mathcal{L})$ and tile the space.

One can view $\mathbb{T}(\mathcal{L})$ as the span $\mathrm{span}(\mathcal{L})$ up to the translation by lattice vectors. We call a set $\mathcal{P} \subset \mathrm{span}(\mathcal{L})$, that represent each element of $\mathbb{T}(\mathcal{L})$ precisely once, a *fundamental domain* of the lattice $\mathcal{L}$. Geometrically, translated copies $\mathbf{v} + \mathcal{P}$ for $\mathbf{v} \in \mathcal{L}$ give a tiling of $\mathrm{span}(\mathcal{L})$.

For any basis $\mathbf{B}$ of a rank n lattice $\mathcal{L}$, the map $\mathbf{x} \mapsto \mathbf{Bx}$ gives an isomorphism from $\mathbb{R}^n/\mathbb{Z}^n$ to $\mathbb{T}(\mathcal{L})$, so $\mathbb{T}(\mathcal{L})$ can be viewed as a n-torus. Since $[-\frac{1}{2}, \frac{1}{2})^n$ is a fundamental domain of $\mathbb{Z}^n \subset \mathbb{R}^n$, the parallelepiped $\mathbf{B} \cdot [-\frac{1}{2}, \frac{1}{2})^n$ is a fundamental domain of $\mathcal{L}$.

Definition 8. *For a basis $\mathbf{B}$ of a rank n lattice $\mathcal{L} \subset \mathbb{R}^d$ we define the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$ as*

$$\mathcal{P}(\mathbf{B}) = \mathbf{B} \cdot [-\tfrac{1}{2}, \tfrac{1}{2})^n = \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i : x_i \in [-\tfrac{1}{2}, \tfrac{1}{2}) \right\},$$

which is a fundamental domain of $\mathcal{L}$, i.e., a set of representatives of the quotient group $\mathbb{T}(\mathcal{L})$.

The volume $\mathrm{vol}(\mathcal{P}(\mathbf{B}) = \mathrm{vol}(\mathbb{T}(\mathcal{L}))$ of such a fundamental domain does not depend on the basis $\mathbf{B}$ of $\mathcal{L}$, and is called the (co)volume of a lattice.

Definition 9 ((Co)volume). *The (co)volume* $\mathrm{vol}(\mathcal{L})$ *of a lattice* $\mathcal{L}$ *is defined as*

$$\mathrm{vol}(\mathcal{L}) := \mathrm{vol}((\mathrm{span}\,\mathcal{L})/\mathcal{L}) = \det(\mathbf{B}^\top\mathbf{B})^{1/2},$$

which is independent of the basis $\mathbf{B}$ *of* $\mathcal{L}$.

The discreteness of a lattice implies that all distinct lattice vectors have at least some minimum distance $\lambda_1(\mathcal{L})$ between them. Equivalently, by the additivity, we only have to look at the minimum distance between the origin and any nonzero lattice vector.

Definition 10 (First Minimum). *The first minimum* $\lambda_1(\mathcal{L}) \in \mathbb{R}_{>0}$ *of a lattice* $\mathcal{L}$ *is defined as the minimal norm of a nonzero lattice vector*

$$\lambda_1(\mathcal{L}) := \min_{\mathbf{v}\in\mathcal{L}\setminus\{0\}} \|\mathbf{v}\|\,.$$

We introduced the volume of a lattice and its first minimum. Minkowski proved, already in 1889, that these two properties are related in the following way.

Theorem 11 (Minkowski's Theorem). *For a lattice* $\mathcal{L}$ *of rank* n *we have the following bound on the first minimum*

$$\lambda_1(\mathcal{L}) \le 2\cdot\frac{\mathrm{vol}(\mathcal{L})^{1/n}}{(\mathrm{vol}\,\mathcal{B}^n)^{1/n}} \le \sqrt{n}\cdot\mathrm{vol}(\mathcal{L})^{1/n}.$$

Proof. Let us assume without loss of generality that the lattice is full rank, i.e., $\mathrm{span}(\mathcal{L}) = \mathbb{R}^n$. We will give a rather visual proof for the first inequality, following Figure 2.2. For this consider the Voronoi domain $\mathcal{V}(\mathcal{L})$, which contains all vectors in the span of the lattice for which $\mathbf{0}$ is a closest lattice vector, namely

$$\mathcal{V}(\mathcal{L}) := \{\mathbf{v}\in\mathbb{R}^n : \|\mathbf{v}\| = \mathrm{dist}(\mathcal{L},\mathbf{v})\}.$$

See the grey box in Figure 2.2 as an example. Because every vector $\mathbf{t}\in\mathbb{R}^n$ has at least one closest lattice vector, we obtain a covering $\mathcal{V}(\mathcal{L})+\mathcal{L}$ of $\mathbb{R}^n$. Furthermore, the intersection $(\mathbf{v}_1+\mathcal{V}(\mathcal{L}))\cap(\mathbf{v}_2+\mathcal{V}(\mathcal{L}))$ for distinct lattice vectors $\mathbf{v}_1\neq\mathbf{v}_2$, is by definition contained in the rank $n-1$ hyperplane of vectors that are equidistant to both $\mathbf{v}_1$ and $\mathbf{v}_2$. In particular, their overlap has trivial n-dimensional volume. So

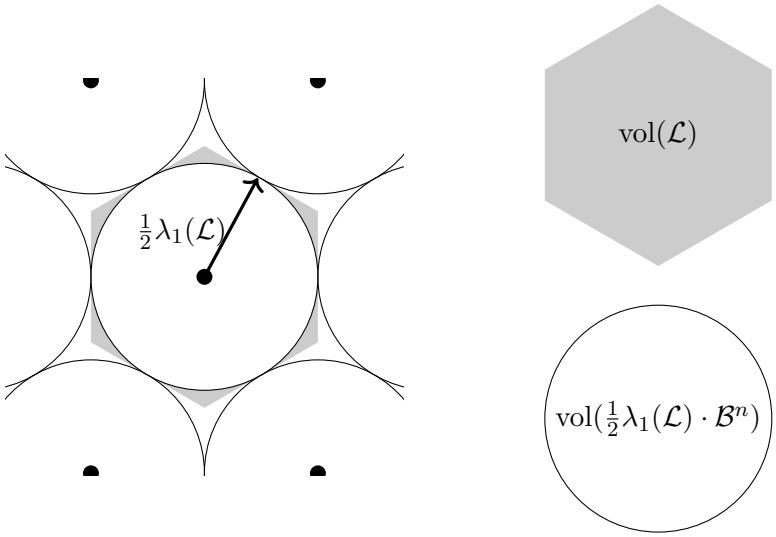

Figure 2.2: Visual proof of Minkowski's Theorem.

the translated Voronoi domains $\mathcal{L} + \mathcal{V}(\mathcal{L})$ form an essentially disjoint tiling of $\mathbb{R}^n$. We can conclude that $\mathrm{vol}(\mathcal{V}(\mathcal{L})) = \mathrm{vol}(\mathcal{L})$.

Since the ball $\frac{1}{2}\lambda_1(\mathcal{L}) \cdot \mathcal{B}^n$ of radius $\frac{1}{2}\lambda_1(\mathcal{L})$ is by definition included in $\mathcal{V}(\mathcal{L})$, we have

$$\tfrac{1}{2}\lambda_1(\mathcal{L}) \cdot \mathrm{vol}(\mathcal{B}^n)^{1/n} = \mathrm{vol}(\tfrac{1}{2}\lambda_1(\mathcal{L}) \cdot \mathcal{B}^n)^{1/n} \leq \mathrm{vol}(\mathcal{V}(\mathcal{L})) = \mathrm{vol}(\mathcal{L}),$$

and the first part of the theorem follows. The second inequality follows by lower bounding the volume $\mathrm{vol}\,\mathcal{B}^n$. □

By the Minkowski's Theorem we can also ask for each rank n what the largest possible ratio is for $\lambda_1(\mathcal{L})/\mathrm{vol}(\mathcal{L})^{1/n}$. The lattice attaining the maximal ratio corresponds to the best n dimensional lattice packing.

Definition 12 (Hermite's constant). *The Hermite constant γ_n is defined as the maximal ratio*

$$\gamma_n := \max_{\mathcal{L}:\mathrm{rk}(\mathcal{L})=n} \frac{\lambda_1(\mathcal{L})^2}{\mathrm{vol}(\mathcal{L})^{2/n}},$$

over all rank n lattices.

We have $\gamma_2 = \frac{4}{3}$ and the maximum is attained by the hexagonal lattice in Figure 2.2. More generally, we can define the n successive minima $\lambda_1(\mathcal{L}), \ldots, \lambda_n(\mathcal{L})$ of a rank n lattice.

Definition 13 (Successive Minima). *For $1 \leq i \leq n$ the i-th minimum $\lambda_i(\mathcal{L})$ of an n-dimensional lattice $\mathcal{L}$ is defined as*

$$\lambda_i(\mathcal{L}) := \min\{\lambda > 0 : \dim(\mathcal{B}_\lambda^n \cap \mathcal{L}) \geq i\}.$$

2.2.3 Dual lattice

Any lattice $\mathcal{L}$ has a corresponding dual lattice.

Definition 14 (Dual Lattice). *For a lattice $\mathcal{L}$ we define the dual lattice, denoted by $\mathcal{L}^*$, as*

$$\mathcal{L}^* := \{\mathbf{w} \in \operatorname{span}(\mathcal{L}) : \langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z} \text{ for all } \mathbf{v} \in \mathcal{L}\}.$$

Given any basis $\mathbf{B}$ of $\mathcal{L}$, the dual lattice $\mathcal{L}^*$ has a unique *dual basis* $\mathbf{D}$ of $\mathbf{B}$ such that $\mathbf{D}^\top \mathbf{B} = \mathbf{I}_n$. Explicitly we have $\mathbf{D} = (\mathbf{B}^\top)^{-1}$ for a full rank lattice, and $\mathbf{D} = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}$ for the general case. The dual of a dual lattice is again the primal lattice $(\mathcal{L}^*)^* = \mathcal{L}$. The volumes of a lattice and its dual are also related.

Lemma 15 (Dual Volume). $\operatorname{vol}(\mathcal{L}^*) = 1/\operatorname{vol}(\mathcal{L})$

Next to their volume, a lattice and its dual have more subtle geometric connections, known as transference relations. We consider one of them by Banaszczyk [Ban93].

Theorem 16 (Transference bound [Ban93]). *For a lattice $\mathcal{L}$ of rank n and its dual $\mathcal{L}^*$ we have*

$$1 \leq \lambda_1(\mathcal{L}) \cdot \lambda_n(\mathcal{L}^*) \leq n.$$

Note that from applying Minkowski's Theorem 11 both to the primal and the dual lattice, and by using Lemma 15, we can already obtain $\lambda_1(\mathcal{L}) \cdot \lambda_1(\mathcal{L}^*) \leq n$. Theorem 16 is a strengthening of this result.

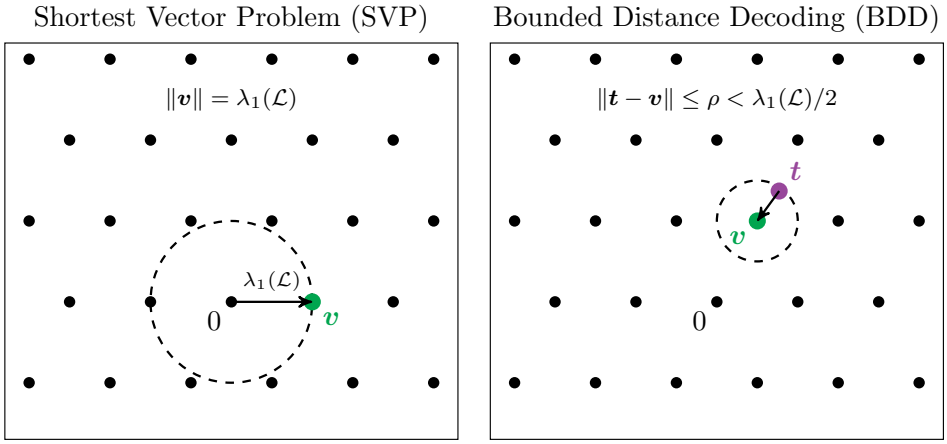

Figure 2.3: The Shortest Vector Problem and Bounded Distance Decoding.

2.3 Lattice problems

2.3.1 Short and close vectors

Cryptography relies on hard problems. For lattices the most famous of them all is the Shortest Vector Problem (SVP).

Definition 17 (Shortest Vector Problem (SVP)). *Given as input a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, compute a nonzero vector* $\mathbf{v} \in \mathcal{L}$ *of length* $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

The Closest Vector Problem could be interpreted as the inhomogeneous version of SVP.

Definition 18 (Closest Vector Problem (CVP)). *Given as input a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, and a target vector* $\mathbf{t} \in \mathrm{span}(\mathcal{L})$*, compute a lattice vector* $\mathbf{c} \in \mathcal{L}$ *closest to* $\mathbf{t}$*, i.e.,* $\|\mathbf{t} - \mathbf{c}\| = \mathrm{dist}(\mathcal{L}, \mathbf{t})$.

We denote oracles solving SVP and CVP by $\mathbf{v} = \mathbf{SVP}(\mathcal{L})$, and $\mathbf{c} = \mathbf{CVP}(\mathcal{L}, \mathbf{t})$.

2.3.2 Approximate variants

Instead of the exact versions of the lattice problems, it is often enough to solve an approximate version to break lattice-based schemes.

Definition 19 (Approximate Shortest Vector Problem (γ-SVP)). *Given as input a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, and an approximation factor* $\gamma \geq 1$*, compute a nonzero vector* $\mathbf{v} \in \mathcal{L}$ *of length* $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$*.*

Definition 20 (Approximate Closest Vector Problem (γ-CVP)). *Given as input a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, a target vector* $\mathbf{t} \in \mathbb{R}^d$ *and an approximation factor* $\gamma \geq 1$*, compute a vector* $\mathbf{v} \in \mathcal{L}$ *such that* $\|\mathbf{t} - \mathbf{v}\| \leq \gamma \cdot \mathrm{dist}(\mathcal{L}, \mathbf{t})$*.*

The above problems can be hard to verify as $\lambda_1(\mathcal{L})$ and $\mathrm{dist}(\mathcal{L}, \mathbf{t})$ might be unknown. A common way to state the above problems such that they become efficiently verifiable is as follows.

Definition 21 (Hermite SVP (γ-HermiteSVP)). *Given as input a basis* $\mathbf{B}$ *of a rank* n *lattice* $\mathcal{L}$*, an approximation factor* $\gamma > 0$*, compute a nonzero vector* $\mathbf{v} \in \mathcal{L}$ *of length* $\|\mathbf{v}\| \leq \gamma \cdot \det(\mathcal{L})^{1/n}$*.*

For small values of γ it is possible that no solution exists. By Minkowski's Theorem 11 there does exist a solution to γ-HermiteSVP when $\gamma \geq \sqrt{n}$.

Definition 22 (Hermite CVP (γ-HermiteCVP)). *Given a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, a target vector* $\mathbf{t} \in \mathbb{R}^d$ *and an approximation factor* $\gamma > 0$*, compute a vector* $\mathbf{v} \in \mathcal{L}$ *such that* $\|\mathbf{t} - \mathbf{v}\| \leq \gamma \cdot \det(\mathcal{L})^{1/n}$*.*

2.3.3 Unique variants

The approximate variants above can have none or multiple solutions for certain inputs. Here we discuss some variants that are promised to always have precisely one unique solution (up to sign).

Definition 23 (Unique SVP (γ-uniqSVP)). *Given as input a factor* $\gamma > 1$ *and a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$ *satisfying* $\lambda_2(\mathcal{L}) \geq \gamma \cdot \lambda_1(\mathcal{L})$*, compute the (unique up to sign) nonzero vector* $\mathbf{v} \in \mathcal{L}$ *of length* $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$*.*

The Bounded Distance Decoding (BDD) problem is a variant of CVP. To make the solution unique the target is promised to lie at a distance less than $\frac{1}{2}\lambda_1(\mathcal{L})$ from the lattice. Two distinct solutions $\mathbf{v}, \mathbf{w} \in \mathcal{L}$ would give a nonzero lattice vector of length $\|\mathbf{v} - \mathbf{w}\| \leq \|\mathbf{v} - \mathbf{t}\| + \|\mathbf{t} - \mathbf{w}\| < \lambda_1(\mathcal{L})$, which gives a contradiction.

Definition 24 (Bounded Distance Decoding (δ-BDD)). *Given as input a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, a distance promise* $\delta \in [0, \frac{1}{2})$*, and a target vector* $\mathbf{t} \in \mathrm{span}(\mathcal{L})$ *satisfying* $\mathrm{dist}(\mathcal{L}, \mathbf{t}) \leq \delta \cdot \lambda_1(\mathcal{L})$*, compute the (unique) closest lattice vector* $\mathbf{c} \in \mathcal{L}$ *to* $\mathbf{t}$*, i.e.,* $\|\mathbf{t} - \mathbf{c}\| = \mathrm{dist}(\mathcal{L}, \mathbf{t})$.

These two problems are very much related. For any $\gamma > 1$, there is a polynomial time reduction from $(\frac{1}{2\gamma})$-BDD to γ-uniqSVP, and for any polynomially bounded γ there is a polynomial time reduction from γ-uniqSVP to $(1/\gamma)$-BDD [LM09]. Essentially, (up to a small approximation factor 2), these problems are equivalent. Note that γ-uniqSVP becomes easier for large γ, while δ-BDD becomes easier for small δ.

2.4 Projecting

Projections are a fundamental tool in lattice theory and lattice algorithms. They naturally reduce high-dimensional (lattice) problems to lower-dimensional ones.

Definition 25 (Projection). *Let* $V \subset \mathbb{R}^d$ *be a linear subspace. Any element* $\mathbf{x} \in \mathbb{R}^d$ *can uniquely be written as* $\mathbf{x} = \mathbf{x}_V + \mathbf{x}_{V^\perp}$ *where* $\mathbf{x}_V \in V$ *and* $\mathbf{x}_{V^\perp} \in V^\perp$*. We define the (orthogonal) projection* $\pi_V : \mathbb{R}^d \to \mathbb{R}^d$ *onto* V *as the linear map*

$$\pi_V : \mathbf{x} \mapsto \mathbf{x}_V \text{ for all } \mathbf{v} \in \mathbb{R}^d,$$

and the orthogonal projection $\pi_{V^\perp} : \mathbb{R}^d \to \mathbb{R}^d$ *away from* V *as the linear map*

$$\pi_{V^\perp} : \mathbf{x} \mapsto \mathbf{x}_{V^\perp} \text{ for all } \mathbf{v} \in \mathbb{R}^d.$$

Note that it is only meaningful to use a projection once, i.e., for any $\mathbf{v} \in \mathbb{R}^d$ we have that $\pi^2(\mathbf{v}) = \pi(\mathbf{v})$. Projections are useful because they not only decrease the dimensionality, but also the length of elements.

Lemma 26. *For any orthogonal projection $\pi : \mathbb{R}^d \to \mathbb{R}^d$, and any $\mathbf{v} \in \mathbb{R}^d$ we have*

$$\|\pi(\mathbf{v})\| \leq \|\mathbf{v}\|.$$

2.4.1 Lattice projections

When projecting lattices we do want the discrete lattice structure to be preserved, which is not in general the case. This only happens if the projection is properly aligned with the lattice. One such a properly aligned example is a projection away from a lattice vector.

Lemma 27. *Let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice. For any vector $\mathbf{v} \in \mathcal{L}$, let $\pi_{\mathbf{v}^\perp}$ be the projection away from $\mathbf{v}$, then $\pi_{\mathbf{v}^\perp}(\mathcal{L})$ is again a lattice (possibly of rank 0).*

Proof. Suppose $\mathcal{L}$ is of rank $n \geq 1$, and assume without loss of generality that $\mathbf{v}$ is scaled such that it is a primitive lattice vector. We can extend $\mathbf{v}$ to a basis $[\mathbf{v}, \mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ of $\mathcal{L}$. Every element $\mathbf{w} \in \mathcal{L}$ can thus be uniquely written as

$$\mathbf{w} = x_0 \cdot \mathbf{v} + \sum_{i=1}^{n-1} x_i \mathbf{b}_i,$$

with $x_i \in \mathbb{Z}$. After projecting we have

$$\pi_{\mathbf{v}^\perp}(\mathbf{w}) = \sum_{i=1}^{n-1} x_i \pi_{\mathbf{v}^\perp}(\mathbf{b}_i).$$

The vectors $\pi_{\mathbf{v}^\perp}(\mathbf{b}_1), \ldots, \pi_{\mathbf{v}^\perp}(\mathbf{b}_{n-1})$ are $\mathbb{R}$-linearly independent, because $\mathbf{v}, \mathbf{b}_1, \ldots, \mathbf{b}_{n-1}$ are, and thus they form a basis $\mathbf{B}'$. We can conclude that $\pi_{\mathbf{v}^\perp}(\mathcal{L}) = \{\sum_{i=1}^{n-1} x_i \pi_{\mathbf{v}^\perp}(\mathbf{b}_i) : x_i \in \mathbb{Z}\} = \mathcal{L}(\mathbf{B}')$ is a lattice. □

Lemma 27 shows that we can project away from a lattice vector, and keep the discrete lattice structure. In the proof we use the fact that $\mathcal{L}$ has a basis starting with (a primitive multiple of) $\mathbf{v}$. In other words, we only have to consider the case where we project away from the first basis vector. Similarly, for the more general case, we only have to consider the case where we project away from the first i basis vectors. A useful object to consider in this setting is the Gram-Schmidt Orthogonalisation (GSO) of a basis.

Definition 28 (Gram-Schmidt Orthogonalisation (GSO)). *Let* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ *be a basis. The Gram-Schmidt Orthogonalisation of* $\mathbf{B}$*, denoted by* $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_{n-1}]$*, is defined as*

$$\begin{aligned}\tilde{\mathbf{b}}_0 &= \mathbf{b}_0\\ \tilde{\mathbf{b}}_i &= \mathbf{b}_i - \pi_{\mathrm{span}(\mathbf{b}_0,\ldots,\mathbf{b}_{i-1})}(\mathbf{b}_i) = \pi_i(\mathbf{b}_i),\end{aligned}$$

where $\pi_i := \pi_{\mathrm{span}(\mathbf{b}_0,\ldots,\mathbf{b}_{i-1})^\perp}$.

The GSO $\tilde{\mathbf{B}}$ of a basis $\mathbf{B}$ is in general not a basis of the same lattice, but it is a basis of the $\mathbb{R}$-span $\mathrm{span}(\mathcal{L}(\mathbf{B}))$. The GSO has a connection with the $\mathbf{QR}$ factorization of the basis $\mathbf{B}$, where $\mathbf{Q}$ is orthonormal and $\mathbf{R}$ is upper triangular, namely the diagonal of $\mathbf{R}$ then consists of $\|\tilde{\mathbf{b}}_0\|, \ldots, \|\tilde{\mathbf{b}}_{n-1}\|$. The same is true for the upper triangular matrix $\mathbf{C}$ in the Cholesky decomposition $\mathbf{G} = \mathbf{C}^\top\mathbf{C}$ of the Gram matrix $\mathbf{G}$ of $\mathbf{B}$. In later chapters the GSO norms $(\|\tilde{\mathbf{b}}_0\|, \ldots, \|\tilde{\mathbf{b}}_{n-1}\|)$ will play an important role, and we call them the *profile* of a basis.

Lemma 29. *Let* $\mathcal{L}$ *be a lattice with basis* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$*, then* $\pi_i(\mathcal{L})$ *is a lattice for* $0 \leq i < n$*, with basis* $[\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_{n-1})]$ *and GSO* $[\tilde{\mathbf{b}}_i, \ldots, \tilde{\mathbf{b}}_{n-1}]$.

Proof. We prove this by induction on $i = 0, \ldots, n-1$. For $i = 0$, π_0 is the identity, and the result is trivially true. For $0 < i = j + 1 \leq n - 1$, we assume by the induction hypothesis that $\mathcal{L}' = \pi_j(\mathcal{L})$ is a lattice, with basis $[\pi_j(\mathbf{b}_j), \ldots, \pi_j(\mathbf{b}_{n-1})]$, and GSO $[\tilde{\mathbf{b}}_j, \ldots, \tilde{\mathbf{b}}_{n-1}]$. The first basis vector of $\mathcal{L}'$ is $\pi_j(\mathbf{b}_j) = \tilde{\mathbf{b}}_j$. Note that $\pi_{j+1} = \pi_{\tilde{\mathbf{b}}_j^\perp} \circ \pi_j$. In particular

$$\pi_{j+1}(\mathcal{L}) = \pi_{\tilde{\mathbf{b}}_j^\perp}(\pi_j(\mathcal{L})) = \pi_{\tilde{\mathbf{b}}_j^\perp}(\mathcal{L}'),$$

which is a lattice by Lemma 27. Furthermore, from the proof of Lemma 27 it follows that $\pi_{\tilde{\mathbf{b}}_j^\perp}(\mathcal{L}')$ has basis

$$[\pi_{\tilde{\mathbf{b}}_j^\perp}(\pi_j(\mathbf{b}_{j+1})), \ldots, \pi_{\tilde{\mathbf{b}}_j^\perp}(\pi_j(\mathbf{b}_{n-1}))] = [\pi_{j+1}(\mathbf{b}_i), \ldots, \pi_{j+1}(\mathbf{b}_{n-1})].$$

The GSO is clear from its definition. We can conclude by induction. □

For any rank n lattice $\mathcal{L}$, for which the basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ is clear from the context, we denote the rank $n - l$ projected lattice

$\pi_l(\mathcal{L}(\mathbf{B}))$ by $\mathcal{L}_{[l:n)}$. We refer to $[l:n)$ as the *context* in which we work, and we call $[0:n)$ the *full context*. Similarly, we denote the rank $0 < r \leq n$ sublattice $\mathcal{L}([\mathbf{b}_0, \ldots, \mathbf{b}_{r-1}]) \subset \mathcal{L}(\mathbf{B})$, by $\mathcal{L}_{[0:r)}$, and the corresponding context by $[0:r)$. Projections and sublattices can be combined, we denote for $0 \leq l < r \leq n$ the basis $[\pi_l(\mathbf{b}_l), \ldots, \pi_l(\mathbf{b}_{r-1})]$ by $\mathbf{B}_{[l:r)}$. We call the rank $r-l$ lattice generated by this basis a *projected sublattice* $\mathcal{L}_{[l:r)} := \mathcal{L}(\mathbf{B}_{[l:r)})$ of $\mathcal{L}$, and refer to the context as $[l:r)$. Furthermore, if the GSO of the full basis $\mathbf{B}$ is $\tilde{\mathbf{B}} := [\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_{n-1}]$, then the GSO of $\mathbf{B}_{[l:r)}$ is given by $\tilde{\mathbf{B}}_{[l:r)} := [\tilde{\mathbf{b}}_i, \ldots, \tilde{\mathbf{b}}_{j-1}]$.

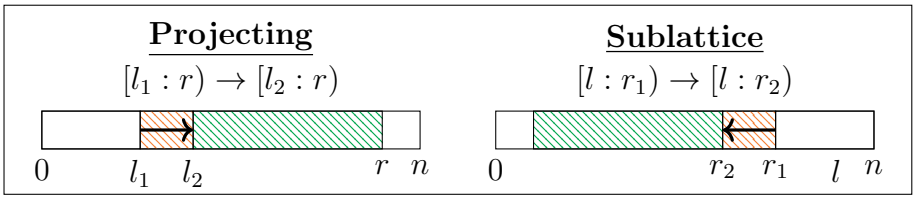

Figure 2.4: Moving to a smaller context by projecting (left), or by going to a sublattice (right).

Projections can be viewed as a way to move from a context $[l_1 : r)$, to a smaller context $[l_2 : r)$ for $l_1 < l_2 \leq r$, while taking a sublattice can be viewed as a way to move from a context $[l, r_1)$, to a smaller context $[l : r_2)$ for $l \leq r_2 < r_1$. See Figure 2.4 for an illustration.

2.4.2 Lifting

So far we have considered the technique of moving from a large context to a smaller context by projecting or going to a sublattice. Now suppose we have done such a move, but we want to revert it, i.e., we want to move back to a larger context. Moving a lattice vector from $\mathcal{L}'$ to a superlattice $\mathcal{L} \supset \mathcal{L}'$ is trivial, since for any vector $\mathbf{v} \in \mathcal{L}'$ we have by definition that $\mathbf{v} \in \mathcal{L}$.

For projections this is not so simple. To move from a context $[l_2 : r)$, back to $[l_1 : r)$ for $l_1 < l_2 \leq r$, we have to undo the projection $\pi_{(\tilde{\mathbf{b}}_{l_1}, \ldots, \tilde{\mathbf{b}}_{l_2-1})^\perp}$ away from $\text{span}(\mathcal{L}_{[l_1:l_2)})^\perp$. Concretely, given a vector $\mathbf{w}' \in \mathcal{L}_{[l_2:r)}$ we have to find a preimage $\mathbf{w} \in \mathcal{L}_{[l_1:r)}$, such that $\pi_{(\tilde{\mathbf{b}}_{l_1}, \ldots, \tilde{\mathbf{b}}_{l_2-1})^\perp}(\mathbf{w}) = \mathbf{w}'$. We call this process *lifting*, and $\mathbf{w}$ a *lift* of $\mathbf{w}'$ from the context $[l_2 : r)$ to $[l_1 : r)$.

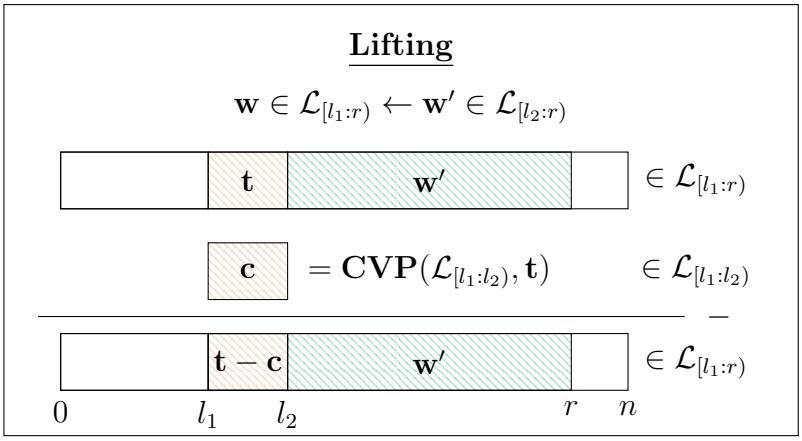

Figure 2.5: Undoing the projection from the context $[l_2 : r)$ to $[l_1 : r)$ for $l_1 < l_2 \leq r$ gives some target $\mathbf{t}$ in the context $[l_2 : r)$. To obtain a small lift we need to find a close vector $\mathbf{c} \in \mathcal{L}_{[l_1:l_2)}$ to $\mathbf{t}$ and subtract it.

Computing some lift is not hard, since $\mathbf{B}_{[l_2:r)}$ is a basis of $\mathcal{L}_{[l_2:r)}$ we can write

$$\mathbf{w}' = \sum_{i=l_2}^{r-1} x_i \cdot \pi_{l_2}(\mathbf{b}_i),$$

with $x_i \in \mathbb{Z}$. A lift of $\mathbf{w}'$ is then given by $\mathbf{w} = \sum_{i=l_2}^{r-1} x_i \cdot \pi_{l_1}(\mathbf{b}_i) \in \mathcal{L}_{[l_1:r)}$. However, such a lift is not unique, and the default lift above can be very long. Preferably, we want some control over the length of the lift. Computing the shortest lift reduces to CVP in the lattice $\mathcal{L}_{[l_1:l_2)}$, see Figure 2.5.

Lemma 30. *Let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice, with basis $[\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$. For $l_1 < l_2 \leq r$, consider the vector $\mathbf{w}' \in \mathcal{L}_{[l_2:r)}$. Given any lift $\mathbf{w} = \mathbf{t} + \mathbf{w}' \in \mathcal{L}_{[l_1:r)}$ of $\mathbf{w}'$, satisfying $\pi_{l_2}(\mathbf{w}) = \mathbf{w}'$, and $\mathbf{t} \in \operatorname{span} \mathcal{L}_{[l_1:l_2)}$, a shortest lift of $\mathbf{w}'$ is given by*

$$(\mathbf{t} - \mathbf{c}) + \mathbf{w}',$$

where $\mathbf{c} = \mathbf{CVP}(\mathcal{L}_{[l_1:l_2)}, \mathbf{t})$ is a closest vector to $\mathbf{t}$.

Proof. Given the lift $\mathbf{w}$ of $\mathbf{w}'$, the full set of lifts is given by $\mathbf{w} - \mathbf{c}$ for $\mathbf{c} \in \mathcal{L}_{[l_1:l_2)} \subset \mathcal{L}_{[l_1:r)}$, since we require that $\pi_{l_2}(\mathbf{c}) = 0$. For any such

$\mathbf{c} \in \mathcal{L}_{[l_1:l_2)}$, the squared norm of the lift $\mathbf{w} - \mathbf{c}$ is given by

$$\|\mathbf{w} - \mathbf{c}\|^2 = \|\mathbf{t} - \mathbf{c} + \mathbf{w}'\|^2 = \|\mathbf{t} - \mathbf{c}\|^2 + \|\mathbf{w}'\|^2.$$

To minimize the above we have to minimize $\|\mathbf{t}-\mathbf{c}\|$ for $\mathbf{c} \in \mathcal{L}_{[l_1:l_2)}$. This is exactly a CVP instance in the lattice $\mathcal{L}_{[l_1:l_2)}$ with target $\mathbf{t}$. A closest vector $\mathbf{c} \in \mathcal{L}_{[l_1:l_2)}$ to $\mathbf{t}$ will thus give a shortest lift $(\mathbf{t} - \mathbf{c}) + \mathbf{w}'$. □

In general solving CVP instances is hard, and thus we cannot always obtain an optimal lift. In dimension 1, however, CVP is as simple as rounding.

Corollary 31. *Let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice, with basis $[\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$, and let $\mathbf{w}' = \sum_{i=l+1}^{r-1} x_i \pi_{l+1}(\mathbf{b}_i) \in \mathcal{L}_{[l+1:r)}$, with $x_i \in \mathbb{Z}$, be a projected lattice vector. A shortest lift $\mathbf{w}$ of $\mathbf{w}'$ from $[l+1:r)$ to $[l:r)$ is given by*

$$\mathbf{w} = \left(\sum_{i=l+1}^{r-1} x_i \pi_l(\mathbf{b}_i)\right) - \left\lfloor \sum_{i=l+1}^{r-1} x_i \cdot \frac{\langle \tilde{\mathbf{b}}_l, \mathbf{b}_i \rangle}{\langle \tilde{\mathbf{b}}_l, \tilde{\mathbf{b}}_l \rangle} \right\rceil \cdot \tilde{\mathbf{b}}_l.$$

In particular the squared norm of $\mathbf{w}'$ to $\mathbf{w}$ increases by at most $\frac{1}{4}\|\tilde{\mathbf{b}}_l\|^2$.

Proof. One lift of $\mathbf{w}'$ is $\mathbf{w} = \sum_{i=l+1}^{r-1} x_i \pi_l(\mathbf{b}_i) \in \mathcal{L}_{[l:r)}$. We obtain the orthogonal sum $\mathbf{w} = \mathbf{t} + \mathbf{w}' \in \mathcal{L}_{[l:r)}$ for $\mathbf{t} = \left(\sum_{i=l+1}^{r-1} x_i \cdot \frac{\langle \tilde{\mathbf{b}}_l, \mathbf{b}_i \rangle}{\langle \tilde{\mathbf{b}}_l, \tilde{\mathbf{b}}_l \rangle}\right) \cdot \tilde{\mathbf{b}}_l$, where we also use that $\langle \tilde{\mathbf{b}}_l, \pi_l(\mathbf{b}_i) \rangle = \langle \tilde{\mathbf{b}}_l, \mathbf{b}_i \rangle$. By Lemma 30 a shortest lift $(\mathbf{t} - \mathbf{c}) + \mathbf{w}'$ now follows from computing a closest vector $\mathbf{c} \in \mathcal{L}_{[l:l+1)} = \tilde{\mathbf{b}}_l \cdot \mathbb{Z}$ to $\mathbf{t}$. For $\mathbf{t} = \lambda \cdot \tilde{\mathbf{b}}_l$ with $\lambda \in \mathbb{R}$, a closest vector is simply given by $\mathbf{c} = \lfloor \lambda \rceil \cdot \tilde{\mathbf{b}}_l$. □

2.4.3 Babai's Nearest Plane Algorithm

Computing the optimal shortest lift reduces to a CVP instance. So optimally lifting a lattice vector from a context of $[l:r)$ to $[0:r)$, is hard for large l. However, by Corollary 31, lifting from $[l:r)$ to $[l-1:r)$ is as simple as rounding, and the squared norm increases by at most $\frac{1}{4}\|\tilde{\mathbf{b}}_{l-1}\|^2$. If we repeat this l times, we arrive at $[0:r)$, with an increase of at most $\frac{1}{4}\sum_{i=0}^{l-1}\|\tilde{\mathbf{b}}_i\|^2$.

Let us take this idea and build a general approximate CVP algorithm from it. First we extract an important property from Corollary 31.

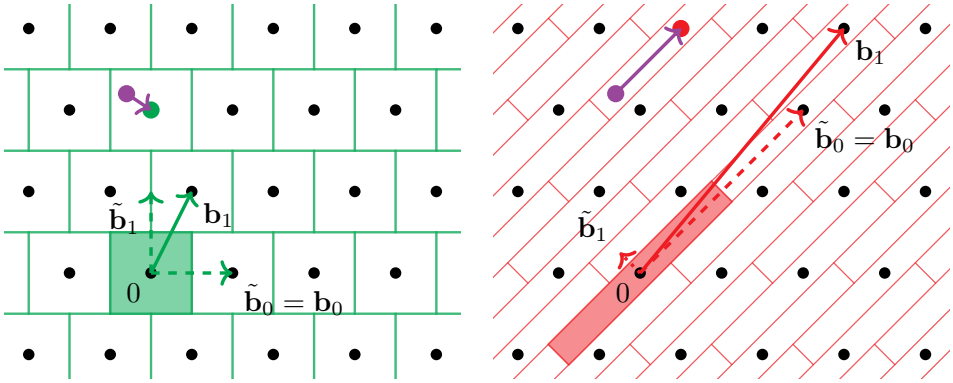

Figure 2.6: Babai's fundamental domain and Babai's nearest plane algorithm for two different bases of the same lattice.

Definition 32 (Size-reduction). *We call a vector* $\mathbf{t} \in \mathbb{R}^d$ *size-reduced w.r.t. a vector* $\mathbf{v} \in \mathbb{R}^d$ *if* $\langle \mathbf{t}, \mathbf{v} \rangle \in [-\frac{1}{2}, \frac{1}{2}) \cdot \|\mathbf{v}\|^2$. *We call a vector* $\mathbf{t} \in \mathbb{R}^d$ *size-reduced w.r.t. a basis* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ *if* $\mathbf{t}$ *is size-reduced w.r.t.* $\tilde{\mathbf{b}}_i$ *for all* $0 \leq i < n$.

Rephrased, any target $\mathbf{t} \in \operatorname{span} \mathcal{L}(\mathbf{B})$ that is size-reduced w.r.t. the basis $\mathbf{B}$ lies in Babai's fundamental domain.

Definition 33 (Babai's fundamental domain). *For a basis* $\mathbf{B}$ *with GSO* $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_{n-1}]$, *we define Babai's fundamental domain as*

$$\mathcal{P}(\tilde{\mathbf{B}}) := \tilde{\mathbf{B}} \cdot [-\tfrac{1}{2}, \tfrac{1}{2})^n = \left\{ \sum_{i=0}^{n-1} \lambda_i \tilde{\mathbf{b}}_i : \lambda_i \in [-\tfrac{1}{2}, \tfrac{1}{2}) \text{ for all } 0 \leq i < n \right\}.$$

Babai's nearest plane algorithm (see Algorithm 1) computes for any target $\mathbf{t} \in \mathcal{P}(\tilde{\mathbf{B}})$ a coset representative $\mathbf{e} \in \mathbf{t} + \mathcal{L}(\mathbf{B})$ that is size-reduced w.r.t. the basis, i.e. such that $\mathbf{e} \in \mathcal{P}(\tilde{\mathbf{B}})$. In other words it computes a close vector $\mathbf{c} = \mathbf{t} - \mathbf{e} \in \mathcal{L}(\mathbf{B})$ to $\mathbf{t}$. It does so by size-reducing one step at a time, from $i = n - 1$ to $i = 0$.

Lemma 34. *For any lattice* $\mathcal{L}$ *with basis* $\mathbf{B}$, *Babai's fundamental domain* $\mathcal{P}(\tilde{\mathbf{B}})$ *is a fundamental domain of the lattice. Let* $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ *be any target and* $(\mathbf{c}, \mathbf{e}) \leftarrow \mathbf{Babai}(\mathbf{B}, \mathbf{t})$ *the output of Algorithm 1. Then* $\mathbf{e} := \mathbf{t} - \mathbf{c} \in \mathcal{P}(\tilde{\mathbf{B}})$, *i.e. Babai's nearest plane algorithms computes the unique close vector with error in* $\mathcal{P}(\tilde{\mathbf{B}})$.

Proof. After iteration i in Algorithm 1, we have by construction that $\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle \in [-\frac{1}{2}, \frac{1}{2}) \cdot \|\tilde{\mathbf{b}}_i\|^2$. In later iterations $j < i$ we only add integer multiples of $\mathbf{b}_j$ to $\mathbf{e}$, and because $\mathbf{b}_j \perp \tilde{\mathbf{b}}_i$ the above inner product stays unchanged. So after all iterations of Algorithm 1 we have $\mathbf{e} \in \mathcal{P}(\tilde{\mathbf{B}})$. Because at the start $\mathbf{e} = \mathbf{t}$, and we only added lattice vectors to $\mathbf{e}$ we have that $\mathbf{t} - \mathbf{e} = \mathbf{c} \in \mathcal{L}$. So Algorithm 1 is correct.

From this correctness it follows that $\mathcal{P}(\tilde{\mathbf{B}})$ represents each coset $\mathbf{t} + \mathcal{L}$ for $\mathbf{t} \in \text{span}(\mathcal{L})$ at least once. For uniqueness let $\mathbf{e}, \mathbf{e}' \in \mathcal{P}(\tilde{\mathbf{B}})$ with $\mathbf{e} + \mathcal{L} = \mathbf{e}' + \mathcal{L}$. Then their difference $\mathbf{v} = \mathbf{e} - \mathbf{e}' \in \mathcal{L}$ is a lattice vector. We can write

$$\mathbf{v} = \sum_{i=0}^{n-1} x_i \mathbf{b}_i,$$

with $x_i \in \mathbb{Z}$. Suppose that $\mathbf{v} \neq 0$, and let $j = \arg\max_i \{x_i \neq 0\}$. Since $\mathbf{e}, \mathbf{e}' \in \mathcal{P}(\tilde{\mathbf{B}})$, and $|x_j| \geq 1$, we get the following contradiction

$$\begin{aligned} \|\tilde{\mathbf{b}}_j\|^2 &> |\langle \mathbf{e}, \tilde{\mathbf{b}}_j \rangle - \langle \mathbf{e}', \tilde{\mathbf{b}}_j \rangle| = |\langle \mathbf{v}, \tilde{\mathbf{b}}_j \rangle| = |\langle \sum_{i=0}^{j} x_i \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle| \\ &= |x_j \cdot \langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle| \geq \|\tilde{\mathbf{b}}_j\|^2. \end{aligned}$$

So $\mathbf{v} = 0$, and $\mathbf{e} = \mathbf{e}'$. □

Algorithm 1: Babai's nearest plane algorithm $\mathbf{Babai}(\mathbf{B}, \mathbf{t})$

Input : A basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ of a lattice $\mathcal{L}$ with Gram-Schmidt orthogonalization $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_{n-1}]$, and a target $\mathbf{t} \in \text{span}(\mathcal{L})$.

Output: $(\mathbf{c}, \mathbf{e})$ such that $\mathbf{c} + \mathbf{e} = \mathbf{t}$, with $\mathbf{c} \in \mathcal{L}$ and $\mathbf{e} \in \mathcal{P}(\tilde{\mathbf{B}})$.

$\mathbf{e} := \mathbf{t}$, $\mathbf{c} := \mathbf{0}$
for $i = n - 1$ *down to* 0 **do**
  $k := \left\lceil \frac{\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\|^2} \right\rfloor$
  $\mathbf{e} := \mathbf{e} - k\mathbf{b}_i$
  $\mathbf{c} := \mathbf{c} + k\mathbf{b}_i$
end
return $(\mathbf{c}, \mathbf{e})$

The shape of Babai's fundamental domain, and thus the usefulness of Babai's nearest plane algorithm, depends on the norms of the GSO vectors.

Corollary 35. *For any lattice $\mathcal{L}$ with basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$, and target $\mathbf{t} \in \mathrm{span}(\mathcal{L})$, let $(\mathbf{c}, \mathbf{e}) \leftarrow \mathbf{Babai}(\mathcal{L}, \mathbf{t})$. Then*

$$\|\mathbf{t} - \mathbf{c}\|^2 \leq \frac{1}{4}\sum_{i=0}^{n-1}\|\tilde{\mathbf{b}}_i\|^2 \leq \frac{1}{4}\sum_{i=0}^{n-1}\|\mathbf{b}_i\|^2.$$

Furthermore, if $\mathrm{dist}(\mathcal{L}, \mathbf{t}) < \frac{1}{2}\min_i\|\tilde{\mathbf{b}}_i\|$, then $\mathbf{c}$ is the unique closest lattice vector to $\mathbf{t}$.

For a uniformly random target $\mathbf{t} \in \mathbb{T}(\mathcal{L})$ Babai's nearest plane algorithm recovers a uniform error in $\mathcal{P}(\tilde{\mathbf{B}})$. The expected squared distance is thus $\frac{1}{12}\sum_{i=0}^{n-1}\|\tilde{\mathbf{b}}_i\|^2$, a factor 3 better than the worst-case.

Since $\mathcal{P}(\tilde{\mathbf{B}})$ is a fundamental domain of $\mathcal{L}$, it has volume $\mathrm{vol}(\mathcal{L})$. As a result we obtain the following invariant.

Corollary 36. *For a rank n lattice with basis $\mathbf{B}$ and GSO $\tilde{\mathbf{B}}$, we have*

$$\mathrm{vol}(\mathcal{P}(\tilde{\mathbf{B}})) = \prod_{i=0}^{n-1}\|\tilde{\mathbf{b}}_i\| = \mathrm{vol}(\mathcal{L}).$$

More generally we have that $\mathrm{vol}(\mathcal{L}_{[l:r)}) = \prod_{i=l}^{r-1}\|\tilde{\mathbf{b}}_i\|$. Due to the invariant $\prod_{i=0}^{n-1}\|\tilde{\mathbf{b}}_i\| = \mathrm{vol}(\mathcal{L})$, the relevant quantities $\sum_{i=0}^{n-1}\|\tilde{\mathbf{b}}_i\|^2$ and $\min_i\|\tilde{\mathbf{b}}_i\|$, for the performance of Babai's nearest plane algorithm, are optimal when $\|\tilde{\mathbf{b}}_0\| = \ldots = \|\tilde{\mathbf{b}}_{n-1}\| = \mathrm{vol}(\mathcal{L})^{1/n}$, i.e. when the GSO norms are perfectly balanced. In Chapter 6 we will see that balancing these GSO norms is the main objective of *basis reduction* algorithms.

It is more than natural to size-reduce a basis $\mathbf{B}$ w.r.t. itself.

Definition 37 (Size-reduced basis). *A basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ is called size-reduced if $\mathbf{b}_j$ is size-reduced with respect to the preceding basis vectors $[\mathbf{b}_0, \ldots, \mathbf{b}_{j-1}]$ for all $1 \leq j < n$. In particular, we have*

$$|\langle\tilde{\mathbf{b}}_i, \mathbf{b}_j\rangle| \leq \frac{1}{2}\|\tilde{\mathbf{b}}_i\|^2 \text{ for all } 0 \leq i < j < n.$$

To size-reduce a basis we can simply apply Babai's nearest plane algorithm several times (in the order $\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}$). Size-reduction relates the basis norms $(\|\mathbf{b}_i\|)_i$ to the GSO norms $(\|\mathbf{b}_i\|)_i$.

Lemma 38 (Relations). *For a size-reduced basis* $\mathbf{B}$ *we have*

$$\|\tilde{\mathbf{b}}_j\|^2 \leq \|\mathbf{b}_j\|^2 \leq \|\tilde{\mathbf{b}}_j\|^2 + \frac{1}{4}\sum_{i<j}\|\tilde{\mathbf{b}}_i\|^2.$$

2.4.4 Dual basis

Recall that for a lattice $\mathcal{L}$ and its dual $\mathcal{L}^*$ we have the volumetric relation $\mathrm{vol}(\mathcal{L}^*) = 1/\mathrm{vol}(\mathcal{L})$. More generally, there also exists a relation between the GSO norms of a basis and the GSO norms of the corresponding dual basis, after some changes to the ordering. Let us consider a basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ with dual basis $\mathbf{D} = [\mathbf{d}_0, \ldots, \mathbf{d}_{n-1}]$. Note that $\mathbf{d}_{n-1}$ is by definition orthogonal to $\mathbf{b}_0, \ldots, \mathbf{b}_{n-2}$, and thus $\mathbf{d}_{n-1}$ lies in the span of $\tilde{\mathbf{b}}_{n-1}$. Furthermore we know that $\langle \mathbf{d}_{n-1}, \mathbf{b}_{n-1}\rangle = 1$, which gives us the following relation

$$\|\mathbf{d}_{n-1}\| \cdot \|\tilde{\mathbf{b}}_{n-1}\| = \langle \mathbf{d}_{n-1}, \tilde{\mathbf{b}}_{n-1}\rangle = \langle \mathbf{d}_{n-1}, \mathbf{b}_{n-1}\rangle = 1.$$

Increasing the last GSO norm $\|\tilde{\mathbf{b}}_{n-1}\|$ thus corresponds to decreasing the length of the last dual basis vector $\mathbf{d}_{n-1}$. If we apply the GSO process in the reversed order, then we obtain relations like this for every GSO norm.

Lemma 39. *Let* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ *be a basis with dual basis* $\mathbf{D} = [\mathbf{d}_0, \ldots, \mathbf{d}_{n-1}]$. *Let* $\mathbf{D}_{-1} = [\mathbf{d}_{n-1}, \ldots, \mathbf{d}_0]$ *be the reversed dual basis with GSO* $[\tilde{\mathbf{d}}_{n-1}, \ldots, \tilde{\mathbf{d}}_0]$, *then* $[\tilde{\mathbf{d}}_i]$ *is a (reversed) dual basis of* $[\tilde{\mathbf{b}}_i]$, *i.e.,*

$$\mathrm{span}(\tilde{\mathbf{d}}_i) = \mathrm{span}(\tilde{\mathbf{b}}_i), \text{ and } \|\tilde{\mathbf{d}}_i\| \cdot \|\tilde{\mathbf{b}}_i\| = 1.$$

In the rest of this thesis, we will refer to $(\|\tilde{\mathbf{d}}_{n-1}\|, \ldots, \|\tilde{\mathbf{d}}_0\|) = (\|\tilde{\mathbf{b}}_{n-1}\|^{-1}, \ldots, \|\tilde{\mathbf{b}}_0\|^{-1})$ as the *profile of the dual basis*, i.e., we will implicitly refer to the GSO norms of the *reversed* dual basis.

2.5 Volumes & distributions

2.5.1 Volumes

We will discuss some higher dimensional geometric objects and their volumes, which play an important role in the analysis of lattice algorithms.

Definition 40 (Sphere and Ball). *For $n \geq 1$ we define the $(n-1)$-sphere $\mathcal{S}_r^{n-1} \subset \mathbb{R}^n$ and the n-ball $\mathcal{B}_r^n \subset \mathbb{R}^n$ of radius $r > 0$ by*

$$\mathcal{S}_r^{n-1} := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = r\}, \text{ and } \mathcal{B}_r^n := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq r\},$$

and we denote $\mathcal{S}^{n-1}$ and $\mathcal{B}^n$ for the sphere and ball of radius 1.

Their volumes are given by

$$\begin{aligned} \mathrm{vol}_n(\mathcal{B}_r^n) &= r^n \cdot V_n, \\ \mathrm{vol}_{n-1}(\mathcal{S}_r^{n-1}) &= r^{n-1} \cdot V_n \cdot n, \end{aligned}$$

where $V_n := \mathrm{vol}_n(\mathcal{B}^n) = \pi^{n/2}/\Gamma(n/2+1)$, with Γ the gamma function.

Definition 41 (Half-space). *For any $\mathbf{v} \in \mathbb{R}^n$ and $a \in \mathbb{R}$ we define the half-space $\mathcal{H}_{\mathbf{v},a}$ by*

$$\mathcal{H}_{\mathbf{v},a} := \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{v}, \mathbf{x} \rangle \geq a\}.$$

The cap in direction $\mathbf{v}$ of a sphere consists of all spherical vectors that are somewhat close to $\mathbf{v}$, or equivalently that have a small angle with $\mathbf{v}$.

Definition 42 (Cap). *For any $\mathbf{v} \in \mathcal{S}^{n-1}$ and $a \in [0,1]$ we define the (spherical) cap $\mathcal{C}_{\mathbf{v},a}^{n-1}$ by the intersection*

$$\mathcal{C}_{\mathbf{v},a}^{n-1} := \mathcal{S}^{n-1} \cap \mathcal{H}_{\mathbf{v},a},$$

and we denote its relative $(n-1)$-dimensional volume, which is independent of $\mathbf{v}$, by $\mathcal{C}^{n-1}(a) := \mathrm{vol}_{n-1}(\mathcal{C}_{\mathbf{v},a}^{n-1}) / \mathrm{vol}_{n-1}(\mathcal{S}^{n-1})$.

The intersection of two caps is called a wedge.

Definition 43 (Wedge). *For any $\mathbf{v}, \mathbf{w} \in \mathcal{S}^{n-1}$, and $a \in \mathbb{R}^n$, we define the (spherical) wedge $\mathcal{W}^{n-1}_{\mathbf{v},\mathbf{w},a}$ by the intersection*

$$\mathcal{W}^{n-1}_{\mathbf{v},\mathbf{w},a} := \mathcal{C}^{n-1}_{\mathbf{v},a} \cap \mathcal{C}^{n-1}_{\mathbf{w},a},$$

and for $c \in [-1, 1]$ and any $\mathbf{v}, \mathbf{w} \in \mathcal{S}^{n-1}$ such that $\langle \mathbf{v}, \mathbf{w} \rangle = c$, we denote its relative $(n-1)$-dimensional volume by $\mathcal{W}^{n-1}(a, c) := \mathrm{vol}_{n-1}(\mathcal{W}^{n-1}_{\mathbf{v},\mathbf{w},a}) / \mathrm{vol}_{n-1}(\mathcal{S}^{n-1})$.

In Chapter 3 we will see that these volumes play an important role in the complexity analysis of lattice sieving algorithms. To compute the exact volume of caps and wedges we rely on the (regularized) incomplete beta function.

Definition 44. *For constants $a, b \in \mathbb{R}^n$ we define the incomplete beta function $B(x; a, b)$ by*

$$B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1} dt,$$

and the regularized incomplete beta function by

$$I_x(a, b) := \frac{B(x; a, b)}{B(a, b)}.$$

The beta distribution $Beta(a, b)$ with $a, b > 0$, is the distribution with support $[0, 1]$ and CDF $x \mapsto I_x(a, b)$.

The regularized incomplete beta function allows us to give an explicit formula for the relative cap volume.

Lemma 45. *For any $a \in [0, 1]$, the relative volume $\mathcal{C}^{n-1}(a)$ of a spherical cap equals*

$$\mathcal{C}^{n-1}(a) = \frac{1}{2} I_{a^2}\left(\frac{1}{2}, \frac{n-1}{2}\right).$$

From this we can also compute wedge volumes by explicit integration. Computing the exact volumes will be useful when deriving concrete complexity estimates, but for the asymptotic complexity results we have simpler formulas.

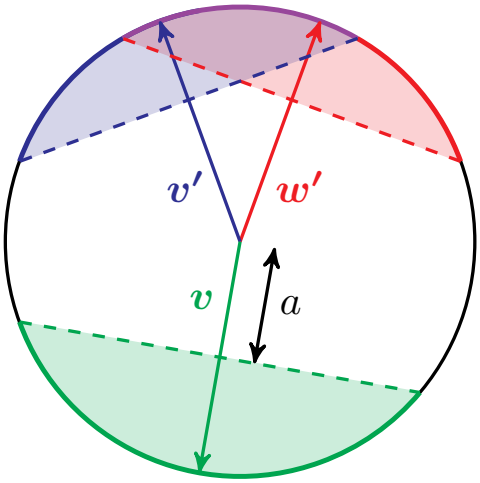

Figure 2.7: Illustration of a spherical cap $\mathcal{C}_{\mathbf{v},a}$ (in green) and a spherical wedge $\mathcal{C}_{\mathbf{v},\mathbf{w},a'} = \mathcal{C}_{\mathbf{v},a'} \cap \mathcal{C}_{\mathbf{w},a'}$ (in purple). The spherical variant consists of just the border, while the ball variant covers the filled in areas.

Lemma 46 (Volumes [MV10; BDGL16]). *For constants $a \in [0,1]$, $c \in [2a^2 - 1, 1]$, and growing n, we have*

$$\mathcal{C}^{n-1}(a) = (1-a^2)^{n/2} \cdot n^{O(1)}, \text{ and}$$
$$\mathcal{W}^{n-1}(a,c) = \left(1 - \frac{2a^2}{1+c}\right)^{n/2} \cdot n^{O(1)}.$$

We see that for fixed constants a, c the volumes scale single exponentially as $2^{Cn+o(n)}$ for some constant $C = C(a,c)$.

Caps and wedges can similarly be defined for balls, but as for large dimensions n most of the volume is located close to the edge, they are asymptotically similar (in particular Lemma 46 also applies to the ball variant).

2.5.2 Distributions

Now that we have introduced caps we can have a better understanding of the properties of uniform samples from a sphere or ball. To better understand the spherical and ball distributions we show how to

sample from them using nothing more than a (continuous) Gaussian distribution.

Definition 47. *We define a (continuous) Gaussian distribution χ_{μ,σ^2} with mean $\mu \in \mathbb{R}$, and standard deviation $\sigma > 0$ on $\mathbb{R}$ by the probability density function*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

Unless otherwise stated $\mu = 0$, and we simply denote $\chi_{\sigma^2} := \chi_{0,\sigma^2}$.

As implied we have mean $\mathbb{E}[\chi_{\mu,\sigma^2}] = \mu$ and variation $\mathbb{V}[\chi_{\mu,\sigma^2}] = \sigma^2$. We can now directly construct a uniform sample over the sphere $\mathcal{S}^{n-1}$ by sampling n Gaussian coefficients and renormalizing the vector.

Lemma 48. *Let $X_1, \dots, X_n \sim \chi_{\sigma^2}$ for any $\sigma > 0$ be independent Gaussian variables, then*

$$\frac{(X_1, \dots, X_n)}{\sqrt{X_1^2 + \dots + X_n^2}} \sim \mathcal{U}(\mathcal{S}^{n-1})$$

Note that for $\sigma = 1$ and large n the denominator $\sqrt{X_1^2 + \dots + X_n^2}$ is highly concentrated around $\sqrt{n}$, which implies that each individual spherical coordinate is distributed close to Gaussian with standard deviation $1/\sqrt{n}$. Using the cap volume formulas we can give an exact description of the coefficient distribution.

Lemma 49. *Let $(u_1, \dots, u_n) \sim \mathcal{U}(\mathcal{S}^{n-1})$, then $u_i^2 \sim Beta(\frac{1}{2}, \frac{n-1}{2})$ on $[0, 1]$ which has CDF $x \mapsto 2\mathcal{C}^{n-1}(\sqrt{x}) = I_x(\frac{1}{2}, \frac{n-1}{2})$.*

The uniform distribution over a ball is similar to that of a sphere, but in addition to the direction we now also have to sample a length.

Lemma 50 (Ball-sphere relation). *Let U be uniform over $[0, 1]$, and S uniform over $\mathcal{S}^{n-1}$, then*

$$U^{1/n} \cdot S \sim \mathcal{U}(\mathcal{B}^n).$$

This also directly implies that the expectation of $\|\mathbf{x}\|^2$ for $\mathbf{x} \sim \mathcal{U}(\mathcal{B}^n)$ is given by $\mathbb{E}[U^{1/n}] = \frac{n}{n+1}$, i.e., for growing n a sample from the unit ball $\mathcal{B}^n$ comes arbitrarily close to the boundary in expectation. Indeed up to a small difference in dimension the coefficients behave exactly the same as a spherical sample due to the following Lemma.

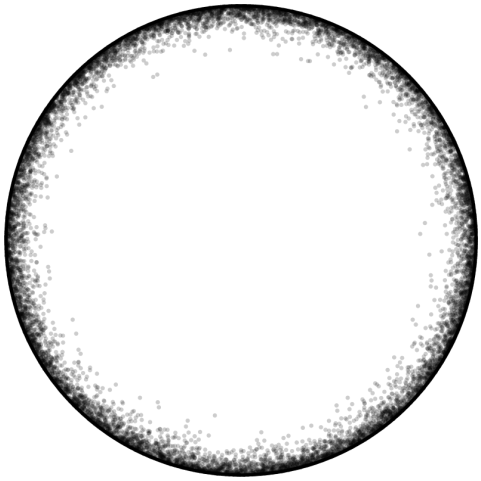

Figure 2.8: Visualisation of the norm of 10000 vectors sampled uniformly over $\mathcal{B}^{30}$, represented by vectors in $\mathcal{B}^2$ with the same norm and a uniform direction. Almost all volume of a high-dimensional ball lies near its border.

Lemma 51. *If $(u_1, \ldots, u_{n+2}) \sim \mathcal{U}(\mathcal{S}^{n+1})$ is spherically distributed, then $(u_1, \ldots, u_n) \sim \mathcal{U}(\mathcal{B}^n)$ is uniform over the ball.*

Corollary 52. *Let $(u_1, \ldots, u_n) \sim \mathcal{U}(\mathcal{B}^n)$, then $u_i^2 \sim Beta(\frac{1}{2}, \frac{n+1}{2})$ with CDF $x \mapsto I_x(\frac{1}{2}, \frac{n+1}{2})$.*

We write the chi-square distribution with k degrees of freedom as $\chi^2_{k,\sigma^2} := \sum_{i=1}^k X_i^2$, where $X_1, \ldots, X_k$ are independently distributed as χ_{σ^2}. The chi-square distribution has expectation $k\sigma^2$, and the following log-expectation.

Lemma 53. *Let X be distributed as χ^2_{k,σ^2}, then*

$$\mathbb{E}\left[\log\left(X\right)\right] = \log(2\sigma^2) + \psi(k/2),$$

where $\psi(x) := \Gamma'(x)/\Gamma(x)$ is the digamma function.

We will also define the discrete Gaussian distribution over $\mathbb{Z}$.

Definition 54. *We define the Gaussian function on $\mathbb{Z}$ with parameter $s > 0$ as $\rho_s(x) = \exp(-\pi x^2/s^2)$. Then the discrete (centered)*

Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma^2}$ on $\mathbb{Z}$ with parameter s is defined by

$$\Pr_{X\sim\mathcal{D}_{\mathbb{Z},\sigma^2}}[X=x]=\frac{\rho_s(x)}{\rho_s(\mathbb{Z})}=\frac{\rho_s(x)}{\sum_{y\in\mathbb{Z}}\rho_s(y)}.$$

For large enough s, say $s \geq 3$, the variance of $\mathcal{D}_{\mathbb{Z},\sigma^2}$ is very close to $s^2/(2\pi)$. The discrete Gaussian distribution can easily be generalized to any rank n lattice, but we will do this in Chapter 10 in the quadratic form setting.

2.6 Heuristics

For cryptanalysis it does not matter if your algorithm cannot solve a particular worst-case instance, as long as it can solve the instance that results in breaking the cryptosystem. Problem instances from most cryptosystems, and in particular those in lattice-based cryptography, rely heavily on randomness to hide secret information.

Often these systems are not perfectly random, but have some hidden secret structure. However, before finding this secret structure (and possibly breaking the scheme), these instances 'behave' like random instances. For cryptanalysis, we are thus interested in the behaviour of our algorithms on those random average-case instances.

Analysing how algorithms behave on average-case instances is often hard. In particular, such an analysis may involve many complex distributions with many dependencies between them. As a result there is a big gap between provable and practical algorithms for lattice problems. Heuristics allow to bridge this gap, and as such, they play an important role in this thesis. For example the best provable SVP algorithm runs in time $2^{n+o(n)}$, while the best practical algorithm runs heuristically in time $2^{0.292n+o(n)}$. We can heuristically assume that some distributions are simpler (and less dependent) then they actually are, and do our analysis with those. Heuristics are not necessarily true in a mathematical sense, it is often easy to give counterexamples. However, if they give estimates that match the practical behaviour of algorithms, then they are still very useful.

One should not blindly follow heuristics. Heuristics should describe practical behaviour, and thus it is important to validate them experimentally. Experiments of feasible size can validate heuristics,

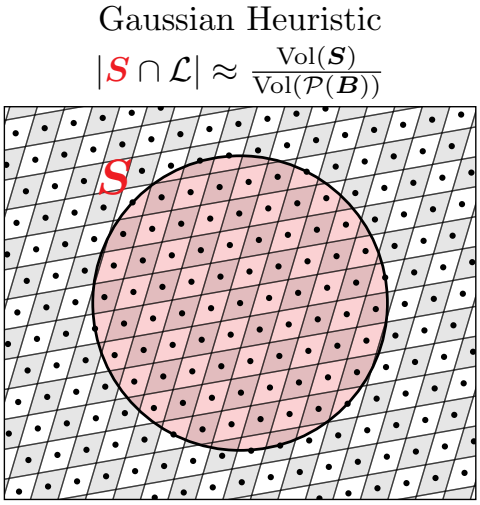

Figure 2.9: Gaussian Heuristic

and those heuristics can then be used to analyse how the algorithm would behave on instances of cryptographic size.

2.6.1 Gaussian Heuristic

The most commonly used, and heavily verified heuristic for lattice algorithms is the so-called Gaussian Heuristic. Heuristics are often inspired by provable average-case statements, and for the Gaussian Heuristic one could give the following origin.

Lemma 55. *For a rank n lattice $\mathcal{L}$, a measurable set $S \subset \mathrm{span}(\mathcal{L})$ and a uniform target $\mathbf{t} \in \mathbb{T}(\mathcal{L}) = \mathrm{span}(\mathcal{L})/\mathcal{L}$ we have*

$$\mathbb{E}_{\mathbf{t} \sim \mathcal{U}(\mathbb{T}(\mathcal{L}))} |(\mathbf{t} + \mathcal{L}) \cap S| = \frac{\mathrm{vol}_n(S)}{\mathrm{vol}(\mathcal{L})}.$$

Lemma 55 essentially says that for a volume S, the number of lattice vectors in $\mathcal{L} \cap S$ is $\mathrm{vol}(S)/\mathrm{vol}(\mathcal{L})$ in expectation if we translate S around. Here $1/\mathrm{vol}(\mathcal{L})$ should be interpreted as the density of the lattice inside its span, e.g. we expect about 1 lattice vector per volume of size $\mathrm{vol}(\mathcal{L})$. The Gaussian Heuristic builds from this interpretation.

Heuristic 56 (Gaussian Heuristic). *For a rank n lattice $\mathcal{L}$ and a measurable set $S \subset \operatorname{span}(\mathcal{L})$ the Gaussian Heuristic states that*

$$|\mathcal{L} \cap S| \approx \frac{\operatorname{vol}_n(S)}{\operatorname{vol}(\mathcal{L})}.$$

Furthermore, the lattice vectors in $\mathcal{L} \cap S$ are uniformly distributed over S.

In some sense the Gaussian Heuristic completely forgets about the lattice structure, and treats the lattice $\mathcal{L}$ as some kind of uniform cloud of vectors with density $1/\operatorname{vol}(\mathcal{L})$. In a reasonable set $S \subset \operatorname{span}(\mathcal{L})$, we then naturally expect about $\operatorname{vol}(S)/\operatorname{vol}(\mathcal{L})$ lattice vectors. The set S is often chosen as a (linear transformations of a) hypercube or ball. E.g., if we apply the Gaussian Heuristic to a ball we obtain an estimate for the first minimum of a lattice.

Heuristic claim 57. *Let $\mathcal{L}$ be a rank n lattice with volume $\operatorname{vol}(\mathcal{L})$. The expectation $\operatorname{gh}(\mathcal{L})$ of the first minimum $\lambda_1(\mathcal{L})$ under the Gaussian Heuristic is given by*

$$\operatorname{gh}(\mathcal{L}) := \frac{\operatorname{vol}(\mathcal{L})^{1/n}}{\operatorname{vol}(\mathcal{B}_1^n)^{1/n}} \approx \sqrt{n/(2\pi e)} \cdot \operatorname{vol}(\mathcal{L})^{1/n}.$$

We also denote $\operatorname{gh}(n) := \operatorname{vol}(\mathcal{B}_1^n)^{-1/n} \approx \sqrt{n/(2\pi e)}$ for the expected first minimum of a rank n lattice with volume 1.

Justification. Assume without loss of generality that the lattice is of full rank $n = d$. We apply the Gaussian Heuristic to the ball $S = \mathcal{B}_\lambda^n \subset \mathbb{R}^n$ of radius $\lambda > 0$. According to the Gaussian Heuristic the number $|\mathcal{L} \cap \mathcal{B}_\lambda^n|$ of lattice vectors in the ball is about $(\lambda/\operatorname{gh}(\mathcal{L}))^n$. E.g. for $\lambda < \operatorname{gh}(\mathcal{L})$ we expect (for large n), no lattice vectors in the ball (except $\mathbf{0}$), while for $\lambda > \operatorname{gh}(\mathcal{L})$ we expect many lattice vectors in the ball. We can conclude that $\lambda_1(\mathcal{L}) \approx \operatorname{gh}(\mathcal{L})$. △

The above estimate for $\lambda_1(\mathcal{L})$ works particularly well in dimensions $n \geq 50$. The expected minimum $\operatorname{gh}(\mathcal{L})$ is precisely a factor 2 smaller than the upper bound on $\lambda_1(\mathcal{L})$ by Minkowski's Theorem 11. So Minkowski's Theorem is rather tight, but a factor 2 can still be

significant if we want to consider the concrete performance of algorithms.

Under the Gaussian Heuristic one could say that average-case exact SVP is equal to γ-HermiteSVP with $\gamma \approx \sqrt{n/(2\pi e)}$. Under the same reasoning, a random target $\mathbf{t} \in (\operatorname{span} \mathcal{L})/\mathcal{L}$ lies at distance about $\operatorname{gh}(\mathcal{L})$ from the lattice. In particular, a random target lies at distance $\delta \cdot \operatorname{gh}(\mathcal{L})$ from the lattice with exponentially small probability $\approx \delta^n$ for a constant $0 < \delta < 1$.

Sometimes the Gaussian Heuristic is a stronger heuristic than what is actually necessary. Therefore there exists many (often weaker) variants of it. For example, sometimes it is enough to only assume that the direction $\mathbf{x}/\|\mathbf{x}\|$ of some lattice vector is uniform (over the sphere).

To avoid confusion we restrict the use of "Theorem", "Lemma", and "Corollary" to formal claims, and refer to "Heuristic claims" for claims that are based on heuristics.

2.6.2 Random lattices

We finish with a short remark on what we mean by average-case instances. For lattice problems the randomness can be in several places, but in particular we discuss here the randomness of the input lattice(s). Formally the set of full rank n lattices can be identified with the quotient group $\mathcal{GL}_n(\mathbb{R})/\mathcal{GL}_n(\mathbb{Z})$. By rescaling the set of all lattices $\mathcal{L}$ with unit (co)volume $\operatorname{vol}(\mathcal{L}) = 1$, can be identified with the group $\mathcal{SL}_n(\mathbb{R})/\mathcal{SL}_n(\mathbb{Z})$. Siegel [Sie45] showed that the Haar measure on $\mathcal{SL}_n(\mathbb{R})$ induces a natural finite measure μ_n on $\mathcal{SL}_n(\mathbb{R})/\mathcal{SL}_n(\mathbb{Z})$. We can thus speak of a uniform distribution over the set of full rank n lattice of unit (co)volume.

This gives a proper definition of what a 'random' lattice is, but such lattices do not naturally appear in cryptography. For example, we often restrict to integer, q-ary or other classes of lattices. Still it can be useful to see how certain lattice properties behave over this distribution. E.g., for a lattice taken uniform w.r.t. μ_n, the first minimum can be shown to have expectation $\operatorname{gh}(\mathcal{L})$ and also to be heavily concentrated around $\operatorname{gh}(\mathcal{L})$ for increasing n. Similarly there exist expectation and concentration results for statements like Lemma 55, see [AEN19] for a survey. These essentially say: almost all lattices follow the Gaussian Heuristic.

Goldstein-Mayer introduced a way to sample integer lattices that in the limit (after rescaling) is statistically close to uniform w.r.t. μ_n. Such lattices are often used to test the average-case behaviour of algorithms. For example the TU Darmstadt SVP challenges give a Goldstein-Mayer random lattice and require you to compute a short vector of length at most $1.05 \cdot \mathrm{gh}(\mathcal{L})$.

Throughout this thesis we will be a bit more lenient with the term random lattice. We might even turn things around and informally call a lattice random if it follows the expected average-case behaviour. E.g. if it follows the Gaussian Heuristic in the general sense, or if at least $\lambda_1(\mathcal{L}) \approx \mathrm{gh}(\mathcal{L})$. To motivate this further: from a cryptanalytic perspective, the average-case is often the hardest for lattice problems. If a lattice does not behave like the average-case, then it leaks some structure or bias that an attacker could exploit. Assuming randomness therefore does not make the problem easier, but does make it easier to analyse. Furthermore, in cryptanalysis these heuristics are often applied to sufficiently randomised (projected sub)lattices, which makes their analysis tight. A common rule of thumb is thus: lattices behave like the average-case, precisely until one recovers (part of) their (secret) structure. Because the heuristics remain valid up to that moment, they can be used to determine when this will happen.

2.7 Cryptography

2.7.1 Security reductions

In the rest of this section we will present multiple cryptographic constructions, such as for encryption and signatures. These constructions come with security notions, often of the form: no probabilistic polynomial-time attacker $\mathcal{A}$ can do X (decrypt, forge a signature). Of course, there is little hope of directly proving such statements. What we can do however, is show that if such an attacker exists, then that attacker can also solve some problem Y in polynomial-time. We call this a security reduction. The security of the (possibly complex) scheme then relies on the hardness of this problem.

One could argue that we just moved the problem somewhere else. However such a reduction gives three main advantages:

1. It shows that there are no particular weaknesses in the complex details of the scheme.

2. Cryptanalysis can focus on a (hopefully) simple to state problem.

3. It allows to reuse and standardize security assumptions.

In the early days of cryptography, before security reductions were a thing, schemes could often be broken by details that were overlooked, such as statistical leakage. A security reduction often finds these faulty details, and brings them to the foreground, because they naturally represent an obstacle in the proof.

Security reductions also allow the reuse of a security assumption: many different schemes can reduce to the same problem. For example, within lattice-based cryptography almost all schemes reduce to just a few problems (but with varying parameters). As a result many cryptanalysts have focussed on those few problems, and thereby increased (or decreased) our confidence in them. Such a toolkit of standard and versatile security assumptions prevents an explosion of ad-hoc assumptions with potentially subtle weaknesses. However, care should still be taken when picking the parameters of such problems. Wandering off the beaten path too much could lead to unexpected attacks.

2.7.2 Zero-Knowledge Proof of Knowledge

As our first cryptographic concept we consider a Zero-Knowledge Proof of Knowledge (ZKPoK). The setting is that we know a solution w to some NP problem x, and we want to convince someone else that we know a solution *without revealing it*. Given an NP relation $\mathcal{R}$, we assume to have a public problem x and two parties, the Prover that supposedly knows a witness w such that $(x, w) \in \mathcal{R}$, and a Verifier that must be convinced of this. For example, there could be a public graph $x := G$, of which the Prover knows an automorphism $w := \sigma$. The Prover then wants to convince some verified that it knows such an automorphism of G, without revealing any such automorphism.

We assume the Prover and Verifier interact with each-other following a sigma protocol, consisting of three phases: commitment, challenge and a response. As visualized in Figure 2.10, the Prover, with input the problem x and a (potential) witness w, starts by creating

Prover(x, w)		Verifier(x)
Generate commitment u		
	$\xrightarrow{u}$	
		Generate challenge $c \leftarrow \mathcal{C}$
	$\xleftarrow{c}$	
Generate response z		
	$\xrightarrow{z}$	Output accept or reject.

Figure 2.10: A Sigma-protocol.

some commitment u. The Verifier, on input of the problem x and the commitment u, responds with a uniform challenge c from some finite challenge set $\mathcal{C}$. Finally, the Prover, responds to the challenge with some response z, which the Verifier accepts or rejects. For a successful Zero-Knowledge Proof of Knowledge scheme we expect the protocol to have the following three properties. The first two relate to the Proof of Knowledge part, while the last one defines the Zero-Knowledge part.

Completeness. If a Prover honestly knows a solution, then the Verifier must be convinced. Formally, if a Prover knows a witness w for the problem x, then after the protocol the Verifier must accept with probability 1.

Special Soundness. If the Prover does not know a witness then it should not be able to convince the Verifier. By guessing the challenge a priori correctly the Prover can often cheat by constructing a commitment and response that makes the Verifier accept for this particular challenge. The initial guess has probability $1/|\mathcal{C}|$ to be correct, and thus the Prover can cheat with probability $1/|\mathcal{C}|$, e.g., a half when $|\mathcal{C}| = 2$. If the Prover can cheat with at most negligible advantage over $1/|\mathcal{C}|$, then by repeating the protocol the overall cheat probability quickly becomes negligible.

To formalize this we must introduce an efficient knowledge extractor $\mathcal{E}$ that given two accepting transcripts (u, c, z) and (u, c', z') with the same commitment but distinct challenges $c \neq c'$, recovers a witness w for the problem x such that $(x, w) \in \mathcal{R}$. I.e., if a Prover is

able to convince the Verifier on at least two distinct challenges with non-negligible probability, then it also knows or can produce a witness with non-negligible probability.

Honest-Verifier Zero-Knowledge (HVZK). The Verifier, if acting honestly, does not learn anything else from the interaction, except that the Prover knows a witness. With acting honestly we mean that the challenges are indeed uniform random over the challenge set $\mathcal{C}$. Given the existence of transformations that remove the need for the Verifier to generate the challenge, we can safely assume this.

To show that the verifier cannot learn anything from the interaction we must show that we can efficiently create a fake interaction, following the same distribution as the real one, without any knowledge of a witness. Formally, we need an efficient simulator that given x, generates accepting transcripts (u, c, z), with a distribution indistinghuisable (negligible statistical distance) from accepting transcripts in the original Sigma-protocol

2.7.3 Encryption and Key Encapsulation Mechanism

While cryptography is a rich field containing many security primitives, many non-experts might think cryptography is only about the encryption of data (or cryptocurrencies...). And indeed encryption, the act of securely transferring data between parties, is a central topic in cryptography. The area can roughly be split into two: symmetric and asymmetric cryptography. In symmetric cryptography one assumes that both parties already conversed a shared secret key (or two related keys), which is then used both for encryption and for decryption. In asymmetric encryption, also known as public key encryption, parties do no share any secret information a priori. Usually one party generates both a secret and a public key, other parties encrypt using the public key, and the generating party decrypts using the secret key.

Often these two methods of encryption are used in unison, first public key encryption is used to securely obtain a (small) shared secret key, also known as a Key Encapsulation Mechanism (KEM), after which more efficient symmetric encryption is used to transfer (large) amounts of data using the shared secret key (see [KL20, Theorem

12.12]). Here we will focus ourself on the public key encryption, or more specifically the KEM.

Definition 58 (Key-Encapsulation Mechanism (KEM) [KL20]). *Let $\lambda \in \mathbb{Z}_{>0}$ be the security parameter. A Key-Encapsulation Mechanism (KEM) $\mathcal{K}$ consists of three probabilistic polynomial-time (in λ) algorithms $\mathcal{K} := (\mathbf{Gen}, \mathbf{Encaps}, \mathbf{Decaps})$ such that:*

1. *Key-generation:* **Gen** *takes a input the security parameter λ, and outputs a secret and public key (sk, pk).*

2. *Encapsulation:* **Encaps** *takes as input a public key pk, and outputs a ciphertext c and a key $\mathbf{k} \in \{0,1\}^\ell$, where $\ell = \Omega(\lambda)$ is the key length. We write $(c, \mathbf{k}) \leftarrow \mathbf{Encaps}(pk)$.*

3. *Decapsulation:* **Decaps** *takes as input a private key sk and a ciphertext c, and outputs a key $\mathbf{k}$ or a failure symbol $\perp$. We write $\mathbf{k} \leftarrow \mathbf{Decaps}(sk, c)$.*

*We call a KEM **correct** if with all but negligible probability over the randomness of* **Gen** *and* **Encaps**, *if* $\mathbf{Encaps}(pk)$ *outputs $(c, \mathbf{k})$, then* $\mathbf{Decaps}(sk, c)$ *also outputs* $\mathbf{k}$.

We now introduce a security notion versus Chosen-Plaintext Attacks (CPA). These definitions often take the form of a game that interacts with the adversary.

Definition 59 (CPA-Security [KL20]). *Let $\mathcal{K} = (\mathbf{Gen}, \mathbf{Encaps}, \mathbf{Decaps})$ be a KEM and $\mathcal{A}$ any adversary. The KEM CPA indistinguishability experiment $\mathbf{KEM}^{\mathbf{cpa}}_{\mathcal{A},\mathcal{K}}(\lambda)$ is as follows:*

- $\mathbf{Gen}(\lambda)$ *is run to obtain a public key $pk = P$. Then* $\mathbf{Encaps}(pk)$ *is run to generate $(c, \mathbf{k})$ with $\mathbf{k} \in \{0,1\}^\ell$ of key length $\ell = \Omega(\lambda)$.*

- *A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$, set $\hat{\mathbf{k}} := \mathbf{k}$, if $b = 1$, choose a uniform $\hat{\mathbf{k}} \in \{0,1\}^\ell$.*

- *Given $(pk, c = (\mathbf{c}, Z), \hat{\mathbf{k}})$ the adversary $\mathcal{A}$ guesses b. If correct the experiment returns 1, otherwise it returns 0.*

The KEM $\mathcal{K}$ is ***CPA-secure*** *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ we have*

$$\Pr[\mathbf{KEM}^{\mathbf{cpa}}_{\mathcal{A},\mathcal{K}}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

One can trivially guess correctly with probability at least $\frac{1}{2}$, and we call the difference away from $\frac{1}{2}$ the *advantage*. So the goal is to achieve a non-negligible advantage in distinguishing the real key from a uniformly random one given the ciphertext and public key.

Recall that usually we want to reduce the CPA-security to some (hopefully) hard problem. Given the distinguishing nature of the CPA experiment, these problems are often of a decisional or distinguishing type, even though for an actual key or message recovery attack one might need to solve a search variant.

CPA-security can be seen as passive security, i.e., it shows that the scheme is secure from a passive attacker that can only eavesdrop on the public channels, but cannot interfere in any way. A stronger security notion is that of chosen-ciphertext attacks (CCA), where the adversary can actively interact with the party holding the secret key, and is even allowed to ask for the decapsulation of ciphertexts (except for the one given in the experiment). Since we can transform a CPA-secure KEM into a CCA-secure KEM by a Fujisaki-Okomoto type of transform [FO99] we restrict our focus to CPA-security.

2.7.4 Signatures

The use of signatures on physical documents or artworks dates back as far as 3000 BC, and they have since been used to associate an identity to a physical object. In the digital world signatures are just as important to verify that e.g., a digital document or software installer was created by some known trusted party, and not by some ill-intentioned adversary. Adding some random scribble to a document is not going to work in the digital world, as one can perfectly copy a signature from one file to another. The signature should be tied to the message, i.e., changing the message should invalidate the signature. We need cryptography to solve this problem for us.

We want the trusted party, called the *sender*, to be able to sign a message, after which anyone, acting as a *verifier*, should be able to

verify the signature. Again this brings us to the public key cryptography setting, the sender uses a secret key to sign, while the public key can be used to verify the signature. Note that in some sense the identity of the sender is represented by its public key, which we assume to be known by all parties. The latter might be seen as a strong requirement, and it is, as we will still need a reliable way to distribute the public key. What a signature scheme does however is to reduce the authentication of many messages to that of authenticating the public key *once*. The remaining problem of sharing public keys is in practice solved by establishing by a Public-Key Infrastructure with a single trusted root. The public key belonging to the trusted root can for example be pre-baked into operating systems, which can then be used to bootstrap the distribution of other public keys. For more information about this see [KL20, Chapter 13.6].

Definition 60 (Signature Scheme [KL20]). *Let $\lambda \in \mathbb{Z}_{>0}$ be the security parameter. A signature scheme $\mathcal{S}$ consists of three probabilistic polynomial-time (in λ) algorithms $\mathcal{S} := (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ such that:*

1. *Key-generation:* **Gen** *takes a input the security parameter λ, and outputs a secret and public key (sk, pk).*

2. *Signing:* **Sign** *takes as input a private key sk and a message m from some message space $\mathcal{M}$, and outputs signature σ. We write $\sigma \leftarrow \mathbf{Sign}(sk, m)$.*

3. *Verification:* **Verify** *takes as input a public key pk, a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We write $b \leftarrow \mathbf{Verify}(pk, m, \sigma)$.*

We call a signature scheme ***correct*** *if with all but negligible probability over the randomness of* **Gen**, *if* $\mathbf{Sign}(sk, m)$ *outputs σ, then* $\mathbf{Verify}(pk, m, \sigma)$ *is valid for every legal message $m \in \mathcal{M}$.*

Given any message and a valid signature we should be convinced that the message was signed by the sender. A message together with a valid signature is called a *forgery* if the message was not signed before by the sender.

Definition 61 (EUF-CMA secure [KL20]). *Let* $\mathcal{S} := (\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ *be a signature scheme and* $\mathcal{A}$ *any adversary. The signature forging experiment* $\mathbf{Sig\text{-}forge}_{\mathcal{A},\mathcal{K}}(\lambda)$ *is as follows:*

- $\mathbf{Gen}(\lambda)$ *is run to obtain keys* (pk, sk).
- *The adversary* $\mathcal{A}$ *is given the public key* pk *and access to an oracle* $\mathbf{Sign}(sk, \cdot)$. *The adversary outputs* (m, σ) *where* m *was not queried before to the oracle.*
- *The output of the experiment is given by* $\mathbf{Verify}(pk, m, \sigma)$, *and we say it succeeds if it is* 1.

The signature scheme $\mathcal{S}$ *is* ***existentially unforgeable under an adaptive chosen-message attack*** *(EUF-CMA) or just* ***secure****, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *we have*

$$\Pr[\mathbf{Sig\text{-}forge}_{\mathcal{A},\mathcal{K}}(\lambda) = 1] \leq \mathrm{negl}(\lambda).$$

2.7.5 Randomness extractors

A randomness extractor allows, using a publicly known random seed, to convert a non-uniform randomness source X with high min-entropy $H_\infty(X) := -\log_2(\max_x \Pr[X = x])$ to a near-uniform random variable [HILL99; Bar+11].[2]

Definition 62 (Extractor). *An efficient function* $\mathcal{E} : \mathcal{X} \times \{0,1\}^z \to \{0,1\}^v$ *is an* (m, ϵ)*-extractor, if, for all random variable* X *distributed over* $\mathcal{X}$ *and* $H_\infty(X) \geq m$, *it holds that*

$$\mathrm{dist}\left((Z, \mathcal{E}(X, Z)), (Z, V)\right) \leq \epsilon$$

where the seed $Z \leftarrow \mathcal{U}(\{0,1\}^z)$ *and* $V \leftarrow \mathcal{U}(\{0,1\}^v)$ *are drawn uniformly at random, and independently of* X.

In Chapter 10, we will rely on the existence of an (m, ϵ)-extractor with parameters $m = \Theta(v)$ and $\epsilon = 2^{-\Theta(m)}$.

[2] For our application in Chapter 10, we do not need to relax the source to only have average min-entropy, and therefore work with the simpler worst-case version.

2.7.6 Hash functions and the Random Oracle Model

Hash functions are an important tool both in cryptography and beyond. They take a (possibly) long input and output a short fixed length 'fingerprint'.

Definition 63 (Hash Function [KL20]). *Let $\lambda \in \mathbb{Z}_{>0}$ be the security parameter. A hash function $\mathcal{H}$ is a pair of probabilistic polynomial-time algorithms $(\mathbf{Gen}, H)$ such that:*

- *Key-generation:* **Gen** *takes as input the security parameter 1^λ, and outputs a key s.*
- *Hash: H is a deterministic algorithm that takes as input a key s and a string $x \in \{0,1\}^*$ and outputs a string $H^s(x) \in \{0,1\}^{\ell(\lambda)}$ of fixed length.*

An important application for hash functions is inside hash-maps, where in general a value x is stored at the memory adress $H(x)$ (or nearby). By using the right hash function, preferably acting somewhat random, and using the right parameters this allows for a map with constant time insertion and look-up.

A simple application in cryptography is to combine them with a signature scheme to obtain a so-called hash-and-sign signature scheme. Instead of directly signing a long message m, which might be inefficient, we sign the short hashed message $H(m)$. Note however that any other message $m' \neq m$ with $H(m) = H(m')$ would give a forgery in the hash-and-sign scheme. To make sure the resulting scheme is still secure we require the hash function to be collision resistant.

Definition 64 (Collision Resistance [KL20]). *Let $\mathcal{H} := (\mathbf{Gen}, H)$ be a hash function and $\mathcal{A}$ any adversary. The collision-finding experiment* $\mathbf{Hash\text{-}coll}_{\mathcal{A},\mathcal{K}}(\lambda)$ *is as follows:*

- $\mathbf{Gen}(1^\lambda)$ *is run to obtain a key s.*
- *The adversary $\mathcal{A}$ is given the key s, and outputs x, x'.*
- *The output of the experiment is defined to be 1 if and only if $x \neq x'$ and $H^s(x) = H^s(x')$. We call this a collision.*

*The hash function $\mathcal{H}$ is **collision resistant**, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ we have*

$$\Pr[\mathbf{Hash\text{-}coll}_{\mathcal{A},\mathcal{K}}(\lambda) = 1] \leq \mathrm{negl}(\lambda).$$

Note in particular that a collision resistant hash function is hard to invert, e.g., it is pre-image resistant. Hash functions that are constructed with the goal of having such cryptographic properties are often called cryptographic hash functions. In practice however most of the efficient cryptographic hash functions in use have no proof of collision resistance (even under some hardness assumptions), but have simply resisted the state-of-the-art cryptanalysis. Additionally proving security of for example a signature scheme often requires that the hash function 'behaves like random', i.e., in the ideal case the hash function is a completely random function from the space of functions from the domain to $\{0,1\}^{\ell(\lambda)}$. Although this is impossible to achieve (efficiently) in practice, we often heuristically assume this is the case. Proofs under this heuristic are known as proofs in the Random Oracle Model (ROM).

In this model there is a public function $H : \mathrm{Domain} \to \{0,1\}^{\ell(\lambda)}$, that on any new input outputs a uniformly random value from $\{0,1\}^{\ell(\lambda)}$. Furthermore, the output should stay the same on already requested values. In the security proof one can 'reprogram' the function H on certain inputs, as long as the output is still (statistically indistinguishably from) uniform, with the argument that from the adversary point of view the distribution of H is unchanged. This is useful for proofs that require simulations.