Cryptographic functions from worst-case complexity assumptions

Daniele Micciancio*

August 16, 2007

Abstract

Lattice problems have been suggested as a potential source of computational hardness to be used in the construction of cryptographic functions that are provably hard to break. A remarkable feature of lattice-based cryptographic functions is that they can be proved secure (that is, hard to break on the average) based on the assumption that the underlying lattice problems are computationally hard in the *worst-case*. In this paper we give a survey of the constructions and proof techniques used in this area, explain the importance of basing cryptographic functions on the *worst-case* complexity of lattice problems, and discuss how this affects the traditional approach to cryptanalysis based on random challenges.

1 Introduction

A lattice is the set of integer linear combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, and can be pictorially described as the set of intersection points of an infinite regular (but not necessarily orthogonal) grid. (See Figure 1.) A typical algorithmic problem on lattices is the following: given a lattice (represented by a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$), find a nonzero lattice vector that is as short as possible, or (in case approximate solutions are allowed) not much longer than the shortest.

Traditionally, in cryptography, lattices have been used mostly as an algorithmic tool for cryptanalysis. Since the development of the LLL basis reduction algorithm of Lenstra, Lenstra and Lovász [21] in the early 80's, lattices have been used to attack a wide range of public key cryptosystems (see survey papers [8, 19, 34] and references therein). Moreover, much work on improving lattice basis reduction algorithms and heuristics (e.g., [43, 40, 41, 42, 6, 20, 31]) has been directly motivated by cryptanalysis applications. Quoting [34], the success of basis reduction algorithms at breaking various cryptographic schemes over the past twenty years has arguably established lattice reduction techniques as the most popular tool in public-key cryptanalysis.

In this survey we are not concerned with the many applications of the LLL algorithm in cryptanalysis. Rather, based on the fact that after 25 years from its discovery, the LLL algorithm is still essentially unsurpassed, we consider the use of lattices as a source of computational hardness to be used in cryptography. More specifically, we consider the design of cryptographic schemes that are provably hard to break based on the conjecture that no efficient algorithm solves various lattice problems substantially better than LLL. We remark that while the LLL algorithm has been substantially improved in terms of running time, progress on improving the approximation

*Department of Computer Science and Engineering, University of California at San Diego, La Jolla CA 92093, USA. Supported in part by NSF grant CCF-0634909.

factor achieved by LLL (while maintaining polynomial running time) has been fairly limited so far: the approximation factor achieved by the currently best polynomial time (randomized) lattice approximation algorithms is $2^{O(n \log \log n / \log n)}$, only a modest improvement over the exponential approximation factor $2^{O(n)}$ achieved by LLL. So, it is reasonable to conjecture that no polynomial time algorithm can approximate lattice problems within factors $n^{O(1)}$ that are polynomial in the rank of the lattice.

In the late 90's, the computational complexity of lattice problems attracted renewed attention, stimulated by Ajtai's surprising discovery [3] of a connection between the worst-case and average-case complexity of certain lattice approximation problems. The importance of such connection relies on the potential applicability of lattices in the *design* of secure cryptosystems. It is a well establish fact that cryptography requires problems that are hard to solve *on the average*, so that when a cryptographic key is chosen at random, the corresponding function is hard to break with high probability. Ajtai's connection shows that such hard-on-average problems can be obtained from the qualitatively weaker assumption that lattice problems are intractable in the *worst case*. To date, all known cryptographic functions almost invariably rely on average-case complexity assumptions. In this respect, lattice problems are exceptional in their ability to provide provably secure cryptographic functions from worst-case complexity assumptions. As a result, Ajtai's discovery attracted a lot of interest within the theoretical cryptography and computational complexity communities, and stimulated substantial research efforts in the area. Following Ajtai's initial discovery, research has progressed on several fronts:

- Determining weaker and weaker worst-case assumptions on the complexity of lattice problems that still allow to distill average-case hardness [10, 26, 29].
- Improving the efficiency of lattice based functions both in terms of key size and computation time [27, 4, 24, 35, 25].
- Building more complex cryptographic primitives than simple one-way functions, like pubic key encryption schemes [5, 36, 37] and identification protocols [30].

These various lines of research, beside being individually interesting from both a theoretical and practical point of view, have a lot in common at a technical level. Ideas and techniques originally introduced in one setting have often found applications in the other contexts (e.g., the Gaussian perturbation techniques originally introduced by Regev [36] to improve the analysis of public key encryption schemes have been further developed and used by Micciancio and Regev [29, 27] for the study of other cryptographic functions). In this paper, we give a self contained introduction to this general area, review the current state of the art and describe the main open problems related to the construction of cryptographic functions based on the worst-case hardness of lattice problems. In particular, we highlight two important issues that have not received much attention so far in the traditional algorithms/complexity and cryptanalysis literature: the study of lattices with special properties, and the development of an appropriate methodology for the cryptanalysis of functions based on worst-case assumptions. For simplicity, at the technical level, we focus on collision resistant hash functions, which are described and analyzed to some depth. We also illustrate the main ideas behind the construction of public key encryption schemes and efficient cryptographic functions based on special classes of lattices, but at a more informal level. For an overview of the zero knowledge proof systems underlying the lattice based identification schemes of [30] the reader is referred to Regev's survey [38] in this volume.

The rest of the paper is organized as follows. In Section 2 we give some general background about lattices. In Section 3 we describe and analyze (a simplified version of) the collision resistant hash function of Micciancio and Regev [29]. In particular, in Section 3.3, we give a detailed and essentially self-contained description of a cryptographic function (and relative security proof) based on a worst-case complexity assumption. In Section 3.5 we also informally discuss the work of Micciancio [27] and subsequent developments [35, 24] on efficient hash functions based on special lattices. Next, in Section 4, we describe the main ideas behind the public key cryptosystems of Ajtai and Dwork [5] and Regev [36]. Section 5 concludes the paper with a discussion of the aforementioned issues related to the security evaluation of cryptographic functions with average-case/worst-case security guarantees.

2 Background

In this section we briefly define the concepts and notation used in the algorithmic study of lattices. For additional background, the reader is referred to [28].

We use $\tilde{O}(f(n))$ to denote the set of all functions $g(n)$ that are asymptotically bounded by $f(n)$ up to poly-logarithmic terms, i.e., $g(n) \leq f(n)\log^c f(n)$ for some constant c and all sufficiently large n. A function $\epsilon(n)$ is *negligible* if $\epsilon(n) < 1/n^c$ for any $c > 0$ and all sufficiently large n.

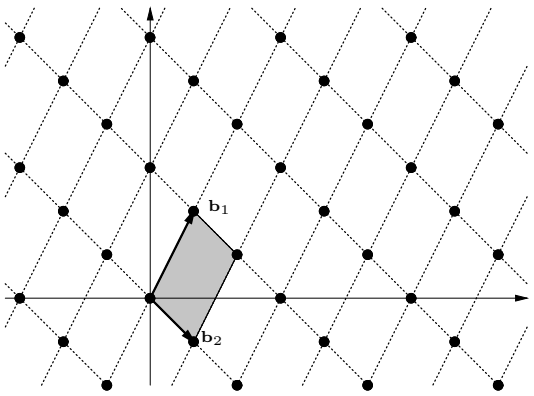

Figure 1: A 2-dimensional lattice.

A lattice (see Figure 1) is the set of all integer linear combinations $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \sum_i \mathbf{b}_i \cdot x_i$ (where $x_i \in \mathbb{Z}$) of a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, called the basis of the lattice. The integer n is called the *rank* of the lattice. Using matrix notation, if $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is the matrix with the basis vectors as columns, lattice vectors can be written as $\mathbf{Bx}$, and $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx}\colon \mathbf{x} \in \mathbb{Z}^n\}$. In this paper we will be mostly concerned with the Euclidean (also known as the ℓ_2) norm $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$. Another norm that will be occasionally used is the ℓ_∞ norm $\|\mathbf{x}\|_\infty = \max_i |x_i|$. Define the half-open parallelepiped $\mathcal{P}(\mathbf{B}) = \{\sum_i x_i \mathbf{b}_i : 0 \leq x_i < 1 \text{ for } 1 \leq i \leq n\}$. For any lattice basis $\mathbf{B}$ and point $\mathbf{x}$ in the linear span of $\mathbf{B}$, there exists a unique vector $\mathbf{y} \in \mathcal{P}(\mathbf{B})$ such that $\mathbf{y} - \mathbf{x} \in \mathcal{L}(\mathbf{B})$. This vector is denoted $\mathbf{y} = \mathbf{x} \bmod \mathbf{B}$, and it can be computed in polynomial time given $\mathbf{B}$ and $\mathbf{x}$. The dual of a lattice Λ is the set

$$\Lambda^* = \{\mathbf{x}\colon \forall \mathbf{y} \in \Lambda \ \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$$

of all vectors that have integer scalar product ($\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$) with all lattice vectors. It is easy to see that for any lattice Λ and $c \in \mathbb{R}$, the dual of the lattice obtained scaling all vectors in Λ by a factor c is $(c\Lambda)^* = (1/c)\Lambda^*$. The *determinant* of a lattice $\det(\mathcal{L}(\mathbf{B}))$ is the (n-dimensional) volume of the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$, and does not depend on the choice of the basis. The determinant of the dual lattice is $\det(\Lambda^*) = 1/\det(\Lambda)$.

The *minimum distance* of a lattice Λ, denoted $\lambda_1(\Lambda)$, is the minimum distance between any two distinct lattice points, and equals the length of the shortest nonzero lattice vector:

$$\begin{aligned} \lambda_1(\Lambda) &= \min\{\|\mathbf{x} - \mathbf{y}\| : \mathbf{x} \neq \mathbf{y} \in \Lambda\} \\ &= \min\{\|\mathbf{x}\| : \mathbf{x} \in \Lambda \setminus \{\mathbf{0}\}\} \ . \end{aligned}$$

This definition can be generalized to define the ith successive minimum as the smallest λ_i such that the ball $\lambda_i \mathcal{B} = \{\mathbf{x}\colon \|\mathbf{x}\| \leq \lambda_i\}$ contains at least i linearly independent lattice points. With some abuse of notation, we will often write $\lambda_i(\mathbf{B})$ to denote the successive minima of the lattice generated by basis $\mathbf{B}$.

A central problem in the computational study of lattices is the *Shortest Vector Problem* (SVP_γ): given a lattice basis $\mathbf{B}$, find a nonzero lattice vector $\mathbf{Bx} \neq \mathbf{0}$ achieving the minimum distance $\|\mathbf{Bx}\| = \lambda_1$. The problem can be defined with respect to any norm, but the Euclidean norm ℓ_2 is the most common. A γ-approximate solution to the shortest vector problem (SVP_γ) is a nonzero lattice vector of length at most $\gamma\lambda_1(\mathbf{B})$. In complexity theory it is common to consider the decision version of SVP_γ (typically denoted GapSVP_γ), defined below.

Definition 1 (Shortest Vector Decision Problem) *An input to $GapSVP_1$ is a pair $(\mathbf{B}, d)$ where $\mathbf{B}$ is an n-dimensional lattice basis and d is a rational number. The problem is to decide whether $\lambda_1(\mathbf{B}) \leq d$ or $\lambda_1(\mathbf{B}) > d$. More generally, for any $\gamma \geq 1$, an input to $GapSVP_\gamma$ is a pair $(\mathbf{B}, d)$ where $\mathbf{B}$ is an n-dimensional lattice basis and d is a rational number. The problem is to decide whether $\lambda_1(\mathbf{B}) \leq d$ or $\lambda_1(\mathbf{B}) > \gamma(n) \cdot d$. (If neither condition is satisfied, any answer is acceptable.)*

Clearly, the promise problem GapSVP_γ reduces to the problem of finding nonzero vectors of length at most $\gamma \cdot \lambda_1$. In the opposite direction, showing that solving GapSVP_γ is at least as hard as computing vectors of length at most $\gamma \cdot \lambda_1$ is a classic open problem. In Section 4 we will also consider a restricted version of GapSVP_γ (denoted $\mathrm{uGapSVP}_\gamma$) where the second minimum of the lattice satisfies $\lambda_2(\mathbf{B}) > \gamma d$. Interestingly, $\mathrm{uGapSVP}_\gamma$ is polynomial time equivalent (in both directions) to the corresponding search problem uSVP_γ: given a lattice basis $\mathbf{B}$ such that $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, find the shortest nonzero lattice vector.

Another classic problem in the study of lattices is the *Closest Vector Problem* (CVP_γ): Given a lattice basis $\mathbf{B}$ and a target vector $\mathbf{t}$, find the lattice point closest to the target. In the approximate version of the problem CVP_γ, the goal is to find a lattice vector within distance from the target no more than γ times the distance of the optimal solution. For any approximation factor γ, CVP_γ is at least as hard as SVP_γ [14].

The last successive minimum λ_n (where n is the rank of the lattice) and corresponding search problem defined below, play a central role in the construction of cryptographic functions with average-case/worst-case connection.

Definition 2 (Shortest Independent Vectors Problem) *An input to $SIVP_\gamma$ is a lattice basis $\mathbf{B}$. The goal is to output a set of n linearly independent lattice vectors $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$ such that $\|\mathbf{S}\| \leq \gamma(n) \cdot \lambda_n(\mathbf{B})$ where $\|\mathbf{S}\|$ is the maximum length of a vector in $\mathbf{S}$, and n is the rank of the lattice.*

The shortest independent vectors problem is closely related to the following variant of the closest vector problem.

Definition 3 (Guaranteed Distance Decoding) *An input to GDD$_\gamma$ is a lattice basis* $\mathbf{B}$ *and a target point* $\mathbf{t}$*. The goal is to output a lattice point* $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{t}-\mathbf{x}\| \leq \gamma(n) \cdot \nu(\mathbf{B})$*, where* $\nu = \max_{\mathbf{t} \in \mathcal{P}(\mathbf{B})} \min_{\mathbf{x} \in \mathcal{L}(\mathbf{B})} \|\mathbf{t}-\mathbf{x}\|$ *is the covering radius of the lattice.*

The difference between GDD$_\gamma$ and CVP$_\gamma$ is that the quality of the solution is measured with respect to the worst possible distance ν, rather than the distance of the given target. (A related and more common variant of CVP$_\gamma$ that arises in coding theory applications, but not used in this survey, is the "Bounded Distance Decoding", where the promise is that the target is very close to the lattice, typically within distance $\lambda_1/2$.) The relation between SIVP$_\gamma$ and GDD$_\gamma$ is easily explained. First, we recall (e.g., see [28, Theorem 7.9]) that for any n-dimensional lattice, ν and λ_n satisfy

$$\lambda_n \leq 2\nu \leq \sqrt{n}\lambda_n. \tag{1}$$

Next, we observe that the proof of (1) is constructive in the sense that it gives efficient reductions from GDD$_{\sqrt{n}\gamma}$ to SIVP$_\gamma$ and from SIVP$_{\sqrt{n}\gamma}$ to GDD$_\gamma$. Specifically, given a solution to SIVP$_\gamma$ instance $\mathbf{B}$, and a target vector $\mathbf{t}$, one can efficiently find (e.g., using Babai's nearest plane algorithm [7]) a lattice point within distance $(\sqrt{n}/2)\gamma \cdot \lambda_n(\mathbf{B})$ from the target. (When $\gamma = 1$, this proves that $2\nu \leq \sqrt{n}\lambda_n$.) Conversely, given a lattice $\mathbf{B}$, one can efficiently find a set of linearly independent lattice vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of length $2\gamma\nu$ making n calls to a GDD$_\gamma$ oracle on input the same lattice $\mathbf{B}$ and n distinct target vectors, where each $\mathbf{t}_i$ is chosen adaptively as a vector orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}$ of length (slightly larger than) $\gamma\nu$. (Again, when $\gamma = 1$, this proves that $\lambda_n \leq 2\nu$.)

3 Collision resistant hashing

In this section we give a general introduction to the problem of designing cryptographic functions that are provably secure based on the worst-case intractability of lattice problems. For simplicity, in this section, we concentrate on collision resistant hashing (defined below), a fairly simple, but still useful, cryptographic primitive. Public key encryption schemes are discussed separately in Section 4.

A collision resistant hash function family is a collection of keyed functions $\{h_k \colon D \to R\}_{k \in K}$ with common domain D and range R. The hash functions are efficiently computable in the sense that there is an efficient algorithm that on input a key $k \in K$ and a value $x \in D$, computes the corresponding value $f_k(x) \in R$ in the range of the functions. Usually, the domain D is larger than the range R, so that by the pigeon-hole principle, each function f_k admits collisions, i.e., pairs $x \neq y$ of distinct inputs that are mapped to the same output $f_k(x) = f_k(y)$. A hash function family is called *collision resistant*, if such collisions are hard to find, i.e., no efficient algorithm on input a randomly chosen key $k \in K$ will find a collision for function f_k. In order to study hash function families in the asymptotic computational setting, one considers sequences $\{h_k \colon D_n \to R_n\}_{k \in K_n}$ of hash function families (indexed by a security parameter n,) with larger and larger domain, range and key space, e.g., $K_n = \{0,1\}^n$, $D_n = \{0,1\}^{2n}$ and $R_n = \{0,1\}^n$.

Most lattice based cryptographic functions have a subset-sum structure, and the constructions are fairly easy to describe. Let $(G, +, 0)$ be a commutative group with binary operation $+$ and identity element 0. Any sequence $\mathbf{a} = (a_1, \ldots, a_m) \in G^m$ of group elements defines a subset-sum

function $f_{\mathbf{a}}(\mathbf{x})$ mapping binary string $\mathbf{x} \in \{0,1\}^m$ to group element

$$f_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^{m} a_i \cdot x_i. \tag{2}$$

A collision for function $f_{\mathbf{a}}$ is a pair of binary vectors $\mathbf{x} \neq \mathbf{y}$ such that $f_{\mathbf{a}}(\mathbf{x}) = f_{\mathbf{a}}(\mathbf{y})$. Equivalently, collisions can be represented by a single vector $\mathbf{z} = \mathbf{x} - \mathbf{y}$ such that $\|\mathbf{z}\|_\infty = 1$ and $f_{\mathbf{a}}(\mathbf{z}) = 0$ is the identity element in G.

Notice that if $m > \log_2 |G|$, then $f_{\mathbf{a}}$ is a hash function, i.e., it compresses the input, and collisions are guaranteed to exist. We want to prove that for appropriate choice of the group G and parameter m, finding collisions to function $f_{\mathbf{a}}$ with non-negligible probability (when the key $\mathbf{a} \in G^m$ is chosen uniformly at random) is at least as hard as solving various lattice approximation problems in the worst case.

We remark that, for any subset-sum function (2), the set of integer vectors $\mathbf{z} \in \mathbb{Z}^m$ satisfying $f_{\mathbf{a}}(\mathbf{z}) = 0$ forms a lattice. So, the problem of breaking (i.e., finding collisions to) hash function $f_{\mathbf{a}}$ can be equivalently formulated as the problem of finding shortest nonzero vectors (in the ℓ_∞ norm) in a random lattice. In this sense, the construction of lattice based hash functions establishes a connection between the worst-case and average-case complexity of (different) lattice problems.

In the rest of this section we first review some mathematical material that will be used in the analysis of the hash functions. Next, we present an oversimplified instantiation and analysis of a lattice based function, which, although technically inaccurate, conveys most of the intuition behind the actual constructions. In Section 3.3 we give a self contained analysis (using the mathematical facts stated in Section 3.1) of a simple lattice based hash function. In this section we strive for simplicity, rather than achieving the best quantitative results. The current state of the art in the construction of lattice based hash functions is discussed in Section 3.4, followed in Section 3.5 by a description of recent work on the construction of very efficient cryptographic functions based on special classes of lattices.

3.1 Background

The mathematical techniques used in the analysis of the currently best results in the area of lattice based cryptography involve Gaussian distributions and the n-dimensional Fourier transform. Fortunately, just a few basic facts in the area are enough to analyze a simplified version of the hash function of [29]. In this section we briefly review all necessary definitions and properties used in our analysis in section 3.3.

For any vector $\mathbf{c} \in \mathbb{R}^n$ and parameter $s > 0$, let $D_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|(\mathbf{x}-\mathbf{c})/s\|^2}/s^n$ be (the probability density function of) the Gaussian distribution with center $\mathbf{c}$ and variance $ns^2/(2\pi)$. For any n-dimensional lattice Λ, let $D_{\Lambda,s,\mathbf{c}}$ be the discrete probability distribution obtained conditioning $D_{s,\mathbf{c}}(\mathbf{x})$ on the event that $\mathbf{x} \in \Lambda$. More precisely, for any $\mathbf{x} \in \Lambda$ let

$$D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{D_{s,\mathbf{c}}(\mathbf{x})}{\sum_{\mathbf{y}\in\Lambda} D_{s,\mathbf{c}}(\mathbf{y})}.$$

In [29], Micciancio and Regev introduce a lattice parameter $\eta_\epsilon(\Lambda)$, called the *smoothing parameter* of the lattice. (To be precise, η_ϵ is a *family* of parameters, indexed by a real number ϵ. Typically, one considers families of lattices Λ_n in increasing dimension, and smoothing parameter η_ϵ where $\epsilon(n)$ is some fixed negligible function of n.) Informally, the smoothing parameter satisfies

the following fundamental property [29, Lemma 4.1]: if $s \geq \eta_\epsilon(\Lambda)$, then adding Gaussian noise with distribution $D_{s,\mathbf{c}}$ to a lattice Λ results in an almost uniform distribution over $\mathbb{R}^n$. More precisely, the property satisfied by the smoothing parameter is that for any n-dimensional lattice $\mathcal{L}(\mathbf{B})$, vector $\mathbf{c} \in \mathbb{R}^n$, and parameter $s \geq \eta_\epsilon(\Lambda)$, the statistical distance between $D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B})$ and the uniform distribution over $\mathcal{P}(\mathbf{B})$ is at most

$$\Delta(D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B}))) \leq \epsilon/2. \tag{3}$$

Here, the operation of adding noise D to a lattice $\mathcal{L}(\mathbf{B})$ is expressed as reducing D (which is a distribution over $\mathbb{R}^n$) modulo the lattice, yielding a distribution over $\mathcal{P}(\mathbf{B})$. Intuitively, this represents a distribution over the entire space obtained by tiling $\mathbb{R}^n$ with copies of $\mathcal{P}(\mathbf{B})$. We refer the reader to [29] for the exact definition of the smoothing parameter, which involves the dual lattice, and is somehow technical. The reader can think of (3) essentially as the definition of the smoothing parameter. Below we state the two only other important properties of the smoothing parameter that will be used in this paper. The first property [29, Lemma 4.2] is that the smoothing parameter is not much larger than λ_n: for any n-dimensional lattice Λ and positive real $\epsilon > 0$

$$\eta_\epsilon(\Lambda) \leq \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}} \cdot \lambda_n(\Lambda). \tag{4}$$

As a special case, if $\epsilon = n^{-\log n}$, then $\eta_\epsilon(\Lambda) \leq \log(n) \cdot \lambda_n(\Lambda)$. The second property is that if $s \geq \eta_\epsilon(\Lambda)$, then the discrete distribution $D_{\Lambda,s,\mathbf{c}}$ behaves in many respect like the continuous distribution $D_{s,\mathbf{c}}$. In particular, (and this is all we need in this paper) [29, Lemma 4.4] shows that if $s \geq \eta_\epsilon(\Lambda)$, then

$$\Pr_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} \{\|\mathbf{x} - \mathbf{c}\| > s\sqrt{n}\} \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}, \tag{5}$$

i.e., the discrete distribution $D_{\Lambda,s,\mathbf{c}}$ is highly concentrated in a sphere of radius $s\sqrt{n}$ around $\mathbf{c}$.

3.2 An oversimplified construction

In the limit, the hash function family that we are going to describe and analyze corresponds to the subset-sum problem over the additive group $\mathbb{R}^n$ of n-dimensional real vectors. Using matrix notation, the subset-sum functions are indexed by a real matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ and map input $\mathbf{x} \in \{0,1\}^m$ to $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \in \mathbb{R}^n$. Clearly, using $\mathbb{R}^n$ as underlying group is not a valid choice, because it results in a function with finite domain $\{0,1\}^m$ and infinite range $\mathbb{R}^n$ that cannot possibly compress the input and be a hash function. Still, the intuition behind the actual construction can be easily illustrated using real numbers. So, let us ignore this finiteness issue for now, and observe that the range of the function does not depend on the parameter m. So, if the range were finite, we could easily make the domain $\{0,1\}^m$ bigger than the range by choosing a sufficiently large m.

We want to prove that collisions are hard to find, in the sense that any procedure that finds collisions to $f_{\mathbf{A}}$ for random $\mathbf{A}$ (with arbitrarily small, but non-negligible probability) can be converted into a worst-case approximation algorithm for lattice problems, e.g., finding short vectors in an arbitrary input lattice $\mathbf{B}$. Technically, we want to give a reduction that on input a lattice basis $\mathbf{B}$ selects a key $\mathbf{A}$ such that the ability to find a collision for $f_{\mathbf{A}}$ helps to find short vectors in $\mathcal{L}(\mathbf{B})$. The challenge is that in order for this to be a valid worst-case to average-case reduction, the key $\mathbf{A}$ selected by the reduction should be (almost) independent from the input lattice $\mathbf{B}$. In

other words, no matter what lattice $\mathcal{L}(\mathbf{B})$ we start from, we should end up generating keys $\mathbf{A}$ with essentially the same (uniformly random) probability distribution.

This can be achieved as follows. Consider the result of selecting a random lattice point $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ and adding a small amount of noise $\mathbf{r}$ to it. Again, here we are being informal. Since $\mathcal{L}(\mathbf{B})$ is a countably infinite set of points, we cannot choose $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ uniformly at random. We will solve this and other technical problems in the next section. Going back to the proof, if the amount of noise is large enough (say, $\mathbf{r}$ is chosen uniformly at random from a sphere of radius $n\lambda_n$ sufficiently larger than the smoothing parameter of the lattice) the resulting point $\mathbf{y} + \mathbf{r}$ will be very close to being uniformly distributed over the entire space $\mathbb{R}^n$. So, we can select a random key $\mathbf{A}$ by choosing, for $i = 1, \ldots, m$, a random lattice point $\mathbf{y}_i \in \mathcal{L}(\mathbf{B})$ and random small error vector $\mathbf{r}_i$, and setting $\mathbf{a}_i = \mathbf{y}_i + \mathbf{r}_i$. The resulting key $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$ won't be distributed exactly at random, but will be sufficiently close to uniform that the hypothetical collision finder algorithm, on input $\mathbf{A}$, will find a collision $f_{\mathbf{A}}(\mathbf{z}) = \mathbf{0}$ (where $\|\mathbf{z}\|_\infty = 1$) with non-negligible probability. (If the collision finder does not produce a valid collision, we simply repeat the procedure again by choosing a new $\mathbf{A}$ independently at random. Since the success probability of the collision finder is non-negligible, a collision will be found with very high probability after at most a polynomial number of iterations.) Let $\mathbf{z}$ be the collision found by the algorithm. We have $\sum_i \mathbf{a}_i z_i = \mathbf{0}$. Substituting $\mathbf{a}_i = \mathbf{y}_i + \mathbf{r}_i$ and rearranging we get

$$\sum_{i=1}^{m} z_i \cdot \mathbf{y}_i = \sum_{i=1}^{m} (-z_i) \cdot \mathbf{r}_i. \tag{6}$$

Notice that the vector on the left hand side of (6) is a lattice vector because it is an integer linear combination (with coefficients $z_i \in \mathbb{Z}$) of lattice vectors $\mathbf{y}_i \in \mathcal{L}(\mathbf{B})$. At the same time, the vector on the right hand side of (6) is a short vector because it is a small linear combination (with coefficients $-z_i \in \{0, 1, -1\}$) of short vectors $\mathbf{r}_i$. In particular, if the error vectors $\mathbf{r}_i$ have length at most $n\lambda_n$, then the vector in (6) has length at most $mn\lambda_n$. So, we have found a lattice vector of length not much bigger than λ_n.

The construction we just described and informally analyzed has several shortcomings. The main problem is that the range of the function is infinite, so the assumption that no algorithm can find collisions is vacuous. (Specifically, when $\mathbf{A} \in \mathbb{R}^{n \times m}$ is chosen at random, the corresponding function is injective with probability 1.) The conclusion that we can find a short vector in a lattice is also trivial, as our proof sketch does not rule out the possibility that the vector $\sum_i \mathbf{y}_i \cdot z_i$ equals $\mathbf{0}$. All these problems are solved in the next section by replacing $\mathbb{R}^n$ with the finite group $\mathbb{Z}_q^n$ of n-dimensional vectors modulo q, and using a variant of the closest vector problem (GDD_γ) as the worst-case problem to be solved.

3.3 Simple cryptographic hash function with worst-case/average-case connection

Consider the subset-sum function family over the group $\mathbb{Z}_q^n$ of integer vectors modulo q. Using matrix notation, this function family is indexed by $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and each function maps binary string $\mathbf{x} \in \{0,1\}^m$ to vector

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n. \tag{7}$$

For concreteness, let us fix $q = 2^n$ and $m = 2n^2$. Notice that the corresponding subset-sum function has a domain of size $|\{0,1\}^m| = 2^{2n^2}$ and range of size $|\mathbb{Z}_q^n| = 2^{n^2}$. So, the function compresses the length of the input by a factor 2, and collisions $f_{\mathbf{A}}(\mathbf{z}) = \mathbf{0}$ (with $\|\mathbf{z}\|_\infty = 1$) are guaranteed

to exist. We prove that finding collisions is computationally hard, even on the average when $\mathbf{A}$ is chosen uniformly at random.

Theorem 1 *Let $\{f_{\mathbf{A}}: \{0,1\}^m \to \mathbb{Z}_q^n\}_{\mathbf{A}}$ be the function family defined in (7), with $q(n) = 2^n$, $m(n) = 2n^2$, and $\mathbf{A} \in \mathbb{Z}^{n \times m}$. If no polynomial time algorithm can solve GDD$_\gamma$ in the worst case within a factor $\gamma = n^3$ (where n is the rank of the input lattice), then $\{f_{\mathbf{A}}\}_{\mathbf{A}}$ is collision resistant.*

Proof. Assume for contradiction that there exists an algorithm $\mathcal{F}$ that on input a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, finds a collision $\mathbf{z}$ with non-negligible probability p. We can assume, without loss of generality, that $\mathcal{F}$ always outputs a vector $\mathbf{z} = \mathcal{F}(\mathbf{A})$ of norm $\|\mathbf{z}\|_\infty = 1$, which may or may not satisfy $\mathbf{Az} = \mathbf{0} \pmod q$. We know that $\Pr\{\mathbf{Az} \bmod q = \mathbf{0}\} = p$ is non-negligible. Notice that for any possible output $\mathbf{z} = \mathcal{F}(\mathbf{A})$, there exists an index $j \in \{1, \ldots, m\}$ such that $z_j \neq 0$. So, by union bound, there is an index $j_0 \in \{1, \ldots, m\}$ such that $\Pr\{\mathbf{Az} \bmod q = \mathbf{0} \ \wedge \ z_{j_0} \neq 0\} \geq p/m = p/2n^2$ is also non-negligible, where the probability is computed over the random selection of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{z} = \mathcal{F}(\mathbf{A})$.

We use $\mathcal{F}$ to approximate the GDD$_\gamma$ variant of the closest vector problem within a factor n^3. Specifically, given a lattice basis $\mathbf{B} \in \mathbb{R}^{n \times n}$ and target $\mathbf{t} \in \mathbb{R}^n$, we find a lattice vector in $\mathcal{L}(\mathbf{B})$ within distance $n^3 \nu(\mathbf{B})$ from $\mathbf{t}$, where $\nu(\mathbf{B})$ is the covering radius of the lattice. As remarked in Section 2, it is easy to see that this problem is at least as hard as approximating SIVP$_\gamma$ within (slightly larger) polynomial factors $\gamma' = n^{3.5}$. Assume that a value $r \in [n^3 \nu(\mathbf{B})/2, n^3 \nu(\mathbf{B})]$ is known. (In general, one can try all possible values $r = 2^k$, and stop as soon as the reduction is successful.) Oracle $\mathcal{F}$ is used as follows:

1. Apply the LLL basis reduction algorithm to $\mathbf{B}$, so that $\|\mathbf{B}\| \leq (2/\sqrt{3})^n \lambda_n(\mathbf{B})$.
2. Choose a random index $j \in \{1, \ldots, m\}$.
3. Choose error vectors $\mathbf{r}_i$ with distribution $D_{s, \delta_{ij}\mathbf{t}}$ where $s = r/(2m\sqrt{n})$, and $\delta_{ij} \in \{0,1\}$ equals 1 if and only if $i = j$. In other words, $\mathbf{r}_i$ is chosen according to a Gaussian distribution of parameter s centered around either $\mathbf{0}$ or $\mathbf{t}$.
4. Let $\mathbf{c}_i = \mathbf{r}_i \bmod \mathbf{B}$ and $\mathbf{y}_i = \mathbf{c}_i - \mathbf{r}_i \in \mathcal{L}(\mathbf{B})$.
5. Let $\mathbf{a}_i = \lfloor q\mathbf{B}^{-1}\mathbf{c}_i \rfloor$, and call the oracle $\mathcal{F}(\mathbf{A})$ on input $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$.
6. Output the vector
$$\mathbf{x} = z_j(\mathbf{BA}/q - \mathbf{Y})\mathbf{z}$$
where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$ and $\mathbf{z} = \mathcal{F}(\mathbf{A})$ is the collision returned by the oracle.

We need to prove that the reduction is correct, i.e., it outputs a lattice vector $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ within distance r from the target $\mathbf{t}$ with non-negligible probability. (Since the success probability depends only on the internal randomness of the reduction algorithm, the failure probability can be made exponentially small using standard repetition techniques.)

Since j is chosen uniformly at random, $j = j_0$ with non-negligible probability $1/m = 1/(2n^2)$. In the rest of the proof, we consider the conditional success probability of the reduction given that $j = j_0$, and show that this conditional probability is still non-negligible.

Notice that, by our assumption on r, and using (4), the Gaussian parameter satisfies $s = r/(2m\sqrt{n}) \geq n^3\nu/4m\sqrt{n} \geq (\sqrt{n}/16) \cdot \lambda_n(\mathbf{B}) \geq \eta_\epsilon(\mathbf{B})$ for any $\epsilon = 2^{o(n)}$. So, by (3), the distribution

of the vectors $\mathbf{c}_i = \mathbf{r}_i \bmod \mathbf{B}$ is within statistical distance $\epsilon/2$ from the uniform distribution over $\mathcal{P}(\mathbf{B})$. Since the function $\mathbf{c} \mapsto \lfloor q\mathbf{B}^{-1}\mathbf{c} \rfloor$ maps the uniform distribution over $\mathcal{P}(\mathbf{B})$ to the uniform distribution over $\mathbb{Z}_q^n$, we get that each vector $\mathbf{a}_i = \lfloor q\mathbf{B}^{-1}\mathbf{c}_i \rfloor$ is also within statistical distance $\epsilon/2$ from the uniform distribution over $\mathbb{Z}_q^n$. Overall, the key $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ is within negligible distance $\epsilon m/2$ from uniform and

$$\Pr\{\mathbf{Az} = \mathbf{0} \pmod q \wedge z_{j_0} \neq 0\} \geq \frac{p}{m} - \frac{\epsilon m}{2} \geq \frac{p}{2m}.$$

Notice that if $\mathbf{Az} = \mathbf{0} \pmod q$, then the output of the reduction

$$\mathbf{x} = z_j \mathbf{B}\frac{\mathbf{Az}}{q} - z_j \mathbf{Yz}$$

is certainly a lattice vector because it is a linear combination of lattice vectors $\mathbf{B}$ and $\mathbf{Y}$ with integer coefficients $z_j(\mathbf{Az}/q)$ and $-z_j\mathbf{z}$. It remains to prove that the conditional probability that $\|\mathbf{t} - \mathbf{x}\| \leq r$ (given that $j = j_0$, $\mathbf{Az} = \mathbf{0} \pmod q$ and $z_{j_0} \neq 0$) is non-negligible. We will prove something even stronger: the conditional probability, given $j = j_0$, $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{z}$ satisfying $\mathbf{Az} = \mathbf{0} \pmod q$ and $z_{j_0} \neq 0$, is exponentially close to 1. We bound the distance $\|\mathbf{t} - \mathbf{x}\|$ as follows:

$$\begin{aligned}
\|\mathbf{t} - \mathbf{x}\| &\leq \|t - z_j(\mathbf{BA}/q - \mathbf{Y})\mathbf{z}\| \\
&= \|z_j \cdot t + (\mathbf{C} - \mathbf{BA}/q)\mathbf{z} + (\mathbf{Y} - \mathbf{C})\mathbf{z}\| \\
&\leq \frac{n}{q}\|\mathbf{B}\| \cdot \|(q\mathbf{B}^{-1}\mathbf{C} - \lfloor q\mathbf{B}^{-1}\mathbf{C} \rfloor)\mathbf{z}\|_\infty + \sum_i |z_i| \cdot \|\mathbf{y}_i - (\mathbf{c}_i - \delta_{ij}\mathbf{t})\| \\
&\leq \frac{nm}{q}\|\mathbf{B}\| + m \cdot \max_i \cdot \|\mathbf{y}_i - (\mathbf{c}_i - \delta_{ij}\mathbf{t})\|.
\end{aligned}$$

Finally, observe that the distribution of $\mathbf{y}_i$, given $\mathbf{c}_i$ and j, is $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}_i - \delta_{ij}\mathbf{t}}$. So, by (5), $\|\mathbf{y}_i - (\mathbf{c}_i - \delta_{ij}\mathbf{t})\| \leq \sqrt{n}s$ with probability exponentially close to 1, and

$$\|\mathbf{t} - \mathbf{x}\| \leq \frac{2n^3(2/\sqrt{3})^n}{2^n}\lambda_n(\mathbf{B}) + m\sqrt{n}\frac{r}{2m\sqrt{n}} = \frac{2n^3}{3^{n/2}}\lambda_n(\mathbf{B}) + \frac{r}{2} < r.$$

□

3.4 Improving the construction and analysis

The main difference between the construction described in Section 3.3 and those studied and analyzed in [2, 10, 26, 29] is that we used a very large value of q (exponential in n), while [2, 10, 26, 29] use $q = n^{O(1)}$. Using a large value of q seems necessary if one starts from an LLL reduced basis and wants to find a short vector with just one application of the reduction. The approach taken in [2, 10, 26, 29] is to use the reduction to find lattice vectors that are shorter than the vectors in the input basis $\mathbf{B}$ by just a small (constant or polynomial) factor. A set of very short lattice vectors can then be found by successive improvements, starting from a basis $\mathbf{B}$ that contains vectors potentially as large as $2^n\lambda_n$, and progressively finding shorter and shorter bases for the lattice.

Using a large value of q has a negative impact on both the efficiency and provable security of the hash function. On the efficiency side, since $2^m > q^n$, using $q = 2^n$ yields hash functions with key size and computation time at least $n \cdot m \cdot \log(q) > n^4$. By contrast, using $q = n^{O(1)}$ yields key

size and computation time $\tilde{O}(n^2)$. On the security side, our proof shows what breaking the hash function is at least as hard as approximating SIVP_γ within a factor $\gamma = n^{3.5}$. In fact, with little more effort, one can reduce the factor to $\gamma = \tilde{O}(\sqrt{mn})$. Still, using $q = 2^n$ results in factors larger than $n^{1.5}$. Using smaller $q = n^{O(1)}$ yields collision resistant hash functions that are at least as hard to break as approximating SIVP_γ within factors $\gamma = \tilde{O}(n)$ almost linear in the dimension of the lattice. This is the best currently known result, proved by Micciancio and Regev in [29].

Theorem 2 *For any sufficiently large polynomial q (e.g., $q(n) = n^3$) the subset-sum function $f_{\mathbf{A}}: \{0,1\}^m \to \mathbb{Z}_q^n$ defined in (2) is collision resistant, assuming that no polynomial time algorithm can solve SIVP_γ (as well as various other lattice problems, e.g., GapSVP_γ) in the worst case within a factor $\gamma = \tilde{O}(n)$ almost linear in the rank of the lattice.*

We remark that the collision resistant hash function of Micciancio and Regev [29] is essentially the same as the one-way function originally proposed by Ajtai [3]. The difference across [3, 10, 26, 29] is mostly in the techniques used to analyze the function, choice of the parameters, and corresponding worst-case factor achieved. Further reducing the factor in the worst-case inapproximability assumption to $\sqrt{n}$ (or even to $n^{1-\epsilon}$ for some constant $\epsilon > 0$) is currently one of the main open problems in the area.

3.5 Using special lattices to improve efficiency

From a practical point of view, a drawback of the cryptographic functions of [3, 10, 26, 29] is that they require a key size $\tilde{O}(n^2)$, approximately quadratic in the natural security parameter n. Unfortunately, it seems hard to build lattice based hash functions with key size sub-quadratic in the dimension of the underlying lattice. Intuitively, the reason is that an n-dimensional lattice is described by an $n \times n$ integer matrix, and this representation is essentially optimal. (For example, it can be easily shown that there are $2^{\Omega(n^2)}$ distinct integer lattices with determinant 2^n.) More concretely, the keys to the hash functions studied in Sections 3.3, 3.4 are matrices

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} & a_{1,n+1} & \cdots & a_{1,2n} & a_{1,2n+1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & a_{n,n+1} & \cdots & a_{n,2n} & a_{n,2n+1} & \cdots & a_{n,m} \end{bmatrix}$$

where the number of rows is the rank n of the underlying lattice, and the number of columns m is strictly larger than n in order to achieve compression. So, even if the entries $a_{i,j}$ are small integers, the key size is still at least $O(n^2)$. A possible approach to overcome this difficulty and get more efficient and *provably secure* lattice based cryptographic functions was suggested by Micciancio in [27], inspired in part by NTRU [17], a very fast commercial cryptosystem. NTRU is a ring based cryptosystem that can equivalently be described using certain lattices with special structure, but for which no proof of security is known. The idea common to [27] and [17] is to use lattices with special structure that can be described with only $\tilde{O}(n)$ bits. For example, [27] considers cyclic lattices, i.e., lattices that are invariant under cyclic rotations of the coordinates. Many such lattices can be

described by giving a single lattice vector $\mathbf{v}$, and a basis for the corresponding lattice

$$\mathbf{C_v} = \begin{bmatrix} v_1 & v_n & v_{n-1} & \cdots & v_2 \\ v_2 & v_1 & v_n & \cdots & v_3 \\ v_3 & v_2 & v_1 & \cdots & v_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_n & v_{n-1} & v_{n-2} & \cdots & v_1 \end{bmatrix}$$

can be obtained by taking $\mathbf{v}$ together with its $n-1$ cyclic rotations.

The novelty in Micciancio's work [27] (which sets it apart from [17]), is that lattices with special structure are used to build very efficient cryptographic functions that also have strong *provable security* guarantees, similar to those of Ajtai's original proposal [3] and subsequent work [10, 26, 29]. Specifically, Micciancio [27] gives a one-way function much more efficient than the one of [3, 29], and still provably secure based on the worst-case intractability of SIVP_γ (and other lattice problems) within almost linear factors over the class of *cyclic lattices*. In particular, the one-way function of [27] has key size and computation time $\tilde{O}(n)$ almost linear in the security parameter n. The adaptation of the proof of [29] to the cyclic setting is non-trivial, and several subtle issues come up during the proof. For example, [27] only shows that the proposed cryptographic function is one-way, but not necessarily collision resistant. In fact, it was later shown (independently, by Peikert and Rosen [35] and Lyubashevsky and Micciancio [24]) that the function of [27] is *not* collision resistant, but it can be easily modified to become a collision resistant hash function provably secure essentially under the same assumptions as in [27]. Lyubashevsky and Micciancio [24] also extend the construction to a wider class of lattices, named "ideal" lattices in that paper.

We only describe the construction of efficient cryptographic functions (called "generalized compact knapsacks"), and refer the reader to the original papers for the proofs of security, which combine the geometric techniques described in Section 3.3 with new algebraic methods. The fundamental idea of [27] is that the key $\mathbf{A}$ to function $f_\mathbf{A}(\mathbf{x}) = \mathbf{Ax} \pmod q$ does not have to be chosen at random from the set $\mathbb{Z}_q^{n\times m}$ of *all* possible matrices. One can restrict, for example, matrix $\mathbf{A}$ to be the concatenation

$$\mathbf{A} = [\mathbf{C}_{\mathbf{a}^1}\mathbf{C}_{\mathbf{a}^2}\cdots\mathbf{C}_{\mathbf{a}^{m/n}}] = \begin{bmatrix} a_1^1 & \cdots & a_2^1 & a_1^2 & \cdots & a_2^2 & a_1^3 & \cdots & a_2^{m/n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ a_n^1 & \cdots & a_1^1 & a_n^2 & \cdots & a_1^2 & a_n^3 & \cdots & a_1^{m/n} \end{bmatrix}$$

of a small number of circulant matrices, i.e., matrices $\mathbf{C}_{\mathbf{a}^i}$ where each column is the cyclic rotation of the previous column. Perhaps unexpectedly, an algorithm to invert random instances of the corresponding function $f_\mathbf{A}$ can be used to approximate SIVP_γ and other lattice problems in the worst case, when restricted to the special case of cyclic lattices, i.e., lattices that are invariant under cyclic rotation of the coordinates. As already remarked, adapting the reduction of [29] from general lattices to the cyclic case is non-trivial, and, in particular achieving collision resistance requires some new algebraic ideas. The nice feature of the new function is that the key $\mathbf{A}$ can now be represented using only $m \log q$ bits (e.g., just the vectors $\mathbf{a}^1, \ldots, \mathbf{a}^{m/n}$, rather than the entire matrix $\mathbf{A}$), and the corresponding function $f_\mathbf{A}$ can also be computed in $\tilde{O}(m \log q)$ time using the fast Fourier transform.

The idea of using structured matrices is further developed in [35, 24] where it is shown how to turn $f_\mathbf{A}$ into a collision resistant hash function. The approaches followed in the two (independent)

papers are closely related, but different. In [35] collision resistance is achieved by restricting the domain of the function to a subset of all binary strings. In [24] it is shown how to achieve collision resistance by appropriately changing the constraint on matrix $\mathbf{A}$. Here we follow the approach of [24] which better illustrates the algebraic ideas common to both papers. The fundamental idea (already present in [27, 17]) is that the ring of circulant matrices is isomorphic to the ring of polynomials $\mathbb{Z}[X]$ modulo $(X^n - 1)$. It turns out that the collision resistance properties of $f_{\mathbf{A}}$ are closely related to the factorization of $X^n - 1$: the linear factor $(X - 1)$ allows to efficiently find collisions, while if we replace $(X^n - 1)$ with an irreducible polynomial $p(X)$, we get a collision resistant function. The proof of security is based on the worst-case intractability of lattice problems over the corresponding class of "ideal" lattices: lattices that can be expressed as ideals of the ring $\mathbb{Z}[X]/p(X)$. (Notice that when $p(X) = X^n - 1$, the ideals of $\mathbb{Z}[X]/p(X)$ correspond exactly to cyclic lattices.) A specific choice of $p(X)$ that results in very efficient implementation [25] is $p(X) = X^n + 1$, which is irreducible when n is a power of 2. In terms of matrices, $X^n + 1$ corresponds to using a variant of circulant matrices

$$\mathbf{C}^-_{\mathbf{v}} = \begin{bmatrix} v_1 & -v_n & -v_{n-1} & \cdots & -v_2 \\ v_2 & v_1 & -v_n & \cdots & -v_3 \\ v_3 & v_2 & v_1 & \cdots & -v_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_n & v_{n-1} & v_{n-2} & \cdots & v_1 \end{bmatrix}$$

where each column is a cyclic shift of the previous one, with the element wrapping around negated.

We emphasize that the lattice intractability assumption underlying the constructions of [27, 35, 24] is a worst-case one, but over a restricted class of lattices. The main open question is whether lattice problems over cyclic (or, more generally, ideal) lattices are indeed hard. Very little is known about them, but state of the art lattice reduction algorithms do not seem to perform any better over cyclic lattices than arbitrary ones, supporting the conjecture that lattice problems over cyclic (or similarly restricted) lattices are as hard as the general case.

4 Public key encryption scheme

One-way functions and collision resistant hash functions (treated in the previous section) are useful cryptographic primitives, and can be used (at least in theory, via polynomial time but not necessarily practical constructions) to realize many other cryptographic operations, like pseudo-random generation, private key encryption, message authentication, commitments schemes, and digital signatures. Unfortunately, this is not the case for *public key encryption*, one of the most fundamental operations of modern cryptography, defined below.

Definition 4 *A public key encryption scheme is a triple of probabilistic polynomial time algorithms (G, E, D) where*

- *G, the key generation algorithm, on input a security parameter n, outputs (in time polynomial in n) a pair of keys (pk, sk), called the public and secret key.*
- *E, the encryption algorithm, on input the public key pk and a message string m (called plaintext), outputs a string $E(pk, m)$, called ciphertext.*
- *D, the decryption algorithm, on input the secret key sk and a ciphertext $c = E(pk, m)$, recovers the original message $D(sk, E(pk, m)) = m$.*

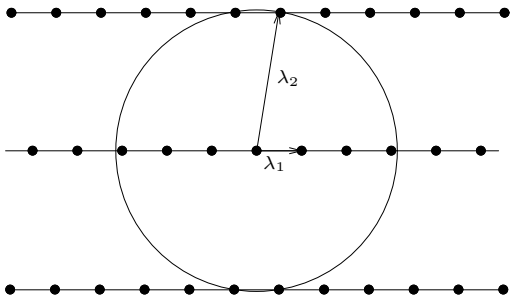

Figure 2: A lattice with "unique" shortest vector.

The typical application of public key encryption schemes is the transmission of confidential information over a public network. Here the intended recipient generates a pair of keys (pk, sk) using the key generation algorithm, and makes his public key pk widely available, e.g., by publishing it next to his name in a directory. Anybody wishing to send a message to this recipient can use the public key pk to encode the message m into a corresponding ciphertext $E(pk, m)$, which is transmitted over the network. The intended recipient can recover the underlying message m using the decryption algorithm and his knowledge of the secret key sk, but it is assumed that nobody else can efficiently perform the same task. This security property is formulated as follows: when the public key pk is generated at random using G, for any two messages m_0, m_1 no efficient (probabilistic polynomial time) adversary, given pk and the encryption $E(pk, m_b)$ of a randomly selected message, can guess the bit b with probability substantially better than $1/2$. This essentially the classic notion of security introduced by Goldwasser and Micali in [15], and typically referred to as security against *chosen plaintext attack* (or, CPA-security).

No public key encryption scheme based on an arbitrary one-way or collision resistant hash function family is known, and any such construction must necessarily be non black-box [18]. Still, public key encryption schemes can be built from many specific (average-case) computational hardness assumptions, e.g., the hardness of factoring random numbers, or computing discrete logarithms in finite fields, etc. Can a public key encryption scheme be constructed and proved secure based on the assumption that SVP_γ or SIVP_γ is hard to approximate in the worst-case?

Inspired by Ajtai's work on lattice-based one-way functions [3], Ajtai and Dwork [5] proposed a public key encryption scheme (subsequently improved by Regev [36], whose proof techniques are followed in this survey) that is provably secure based on the worst-case hardness of a lattice problem, although a seemingly easier one that those underlying the construction of one-way hash functions. The problem underlying the Ajtai-Dwork cryptosystem can be described as a restriction of SVP_γ to a special class of lattices, namely lattices such that $\lambda_2 > \gamma\lambda_1$ for some factor $\gamma = n^{O(1)}$ polynomial in the rank of the lattice. (See figure 2.) The restriction of SVP_γ to such lattices is usually referred to as the "unique shortest vector problem" (uSVP_γ).

Definition 5 (Unique Shortest Vector Problem, uSVP_γ) *On input a lattice basis* $\mathbf{B}$ *such that* $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$*, find a nonzero lattice vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *of length* $\lambda_1(\mathbf{B})$*.*

The name of this problem is motivated by the fact that in such lattices the shortest nonzero vector $\mathbf{v}$ is unique, in the sense that any vector of length less than $\gamma\lambda_1$ is parallel to $\mathbf{v}$. It is also easy to see that for such lattices, finding a γ-approximate solution to SVP_γ is equivalent to finding

the shortest nonzero lattice vector exactly: given a γ-approximate solution $\mathbf{v}$, the shortest vector in the lattice is necessarily of the form $\mathbf{v}/c$ for some $c \in \{1, \ldots, \gamma\}$, and can be found in polynomial time by checking all possible candidates for membership in the lattice. (Here we are using the fact that γ is polynomially bounded. Better ways to find the shortest vector exist, which work for any factor γ.) Interestingly, it can be shown [36] that solving uSVP_γ is also equivalent to the decision problem GapSVP_γ under the additional promise that the input instance $(\mathbf{B}, d)$ satisfies $\lambda_2(\mathbf{B}) > \gamma d$. We refer to this problem as uGapSVP_γ.

Definition 6 (Unique Shortest Vector Decision Problem, uSVP_γ) *Given a lattice basis* $\mathbf{B}$ *and a real* d*, distinguish between these two cases*

- $\lambda_1(\mathbf{B}) \leq d$ *and* $\lambda_2(\mathbf{B}) > \gamma d$*,*
- $\lambda_1(\mathbf{B}) > \gamma d$ *and* $\lambda_2(\mathbf{B}) > \gamma d$*.*

If the input satisfies neither condition, any answer is acceptable.

We remark that this is different from the variant of GapSVP_γ considered in [9], and proved not to be NP-hard for factor $\gamma = n^{1/4}$ under standard complexity assumptions. The problem studied in [9] corresponds to GapSVP_γ with the stronger additional promise that $\lambda_2(\mathbf{B}) > \gamma\lambda_1(\mathbf{B})$, and is not known to be equivalent to uSVP_γ.

Theorem 3 ([36]) *For any polynomially bounded factor* γ*,* $uSVP_\gamma$ *and* $uGapSVP_\gamma$ *are equivalent under polynomial time reductions.*

Another equivalent formulation of uSVP_γ, is the "hidden hyperplane problem", described below, which is the problem directly underlying the Ajtai-Dwork cryptosystem. Informally, the "hidden hyperplane problem" is the problem of distinguishing the uniform distribution over $\mathbb{R}^n$ from a distribution concentrated nearby a set of equally spaced parallel hyperplanes $H_i = \{\mathbf{v} \colon \langle \mathbf{v}, \mathbf{u} \rangle = i\}$ (where $\mathbf{u}$ is a vector orthogonal to the hyperplanes of length inversely proportional to the distance between consecutive hyperplanes, see Figure 3.) The relation between the hidden hyperplane

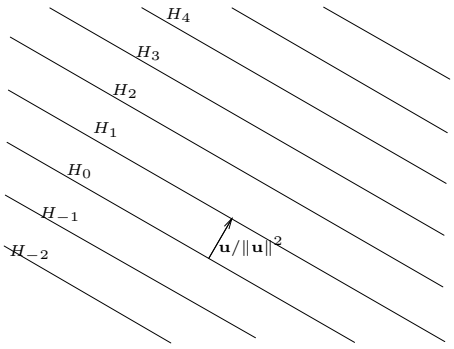

Figure 3: Hyperplanes $H_i = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{v} \rangle = i\}$ defined by a vector $\mathbf{u}$.

problem and uSVP_γ is easily explained, but it requires the use of dual lattices. Recall that the

dual lattice $\mathcal{L}(\mathbf{B})^*$ is the set of all vectors $\mathbf{u}$ that have integer scalar product with all lattice vectors $\mathbf{v} \in \mathcal{L}(\mathbf{B})$. So, any dual vector $\mathbf{u}$ partitions the lattice $\mathcal{L}(\mathbf{B})$ into consecutive hyperplanes

$$H_i = \{\mathbf{v} \colon \langle \mathbf{u}, \mathbf{v} \rangle = i\}$$

(for $i \in \mathbb{Z}$) at distance $1/\|\mathbf{u}\|$ from each other. If the dual lattice contains a very short vector $\mathbf{u}$, then the distance between the hyperplanes defined by $\mathbf{u}$ will be very large. Moreover, if the successive minimum $\lambda_2(\mathcal{L}(\mathbf{B})^*)$ is much larger than $\|\mathbf{u}\|$ (i.e., $\mathcal{L}(\mathbf{B})^*$ is a uSVP$_\gamma$ lattice), then the distance between the hyperplanes will be much larger than the characteristic distance between lattice points within each hyperplane. So, a uGapSVP$_\gamma$ instance $(\mathbf{B}, d)$ can be reduced to the hidden hyperplane problem as follows:

1. Pick a random lattice point in the dual lattice $\mathbf{u} \in \mathcal{L}(\mathbf{B})^*$. (Here we are being rather informal. Technical problems like the impossibility of choosing lattice points uniformly at random can be easily solved by methods similar to those described in Section 3.)

2. Perturb the dual lattice point by a random error vector $\mathbf{r}$.

3. Run the hidden hyperplane distinguisher on the perturbed dual lattice point $\mathbf{u} + \mathbf{r}$.

We observe that if the length of the error vector is appropriately chosen, the reduction is correct. We only describe the main idea here, and refer the reader to [5, 36] for the technical details. If $(\mathbf{B}, d)$ is a "yes" instance of uGapSVP$_\gamma$, then $\mathcal{L}(\mathbf{B})$ contains a vector $\mathbf{u}$ of length at most d, and the dual lattice $\mathcal{L}(\mathbf{B}^*)$ can be partitioned into hyperplanes at distance $1/d$ from each other. Moreover, the additional uGapSVP$_\gamma$ promise that $\lambda_2 > \gamma d$ guarantees that the distance between dual lattice points within each hyperplane is much smaller than $1/d$ (essentially proportional to $1/\gamma d$). So, by adding some noise (sufficiently bigger than $1/\gamma d$, but smaller than $1/d$) to the dual lattice points, we can erase the fine structure of the dual lattice within each hyperplane, and obtain a distribution concentrated nearby (and almost uniform over) these hyperplanes. On the other hand, if $(\mathbf{B}, d)$ is a "no" instance of uGapSVP$_\gamma$, then $\lambda_1(\mathbf{B}) > \gamma d$, and the characteristic distance between *all* dual lattice points is proportional to $1/\gamma d$. So, by adding an amount of noise much larger than $1/\gamma d$ we get a distribution that is very close to uniform over the entire space. This shows that the perturbed dual lattice point $\mathbf{u} + \mathbf{r}$ is chosen according to one of the two hidden hyperplane distributions, almost uniform either over the entire space or nearby the hidden hyperplanes. So, the reduction is correct.

The nice feature of the hidden hyperplane problem is that it is, in a sense, random self reducible: given an arbitrary instance of the problem, we can obtain a random instance by applying a random rotation and scaling. (Technically, the hidden hyperplane problem is not a computational problem in the standard sense, as its instances are not strings, but probability distributions. What we mean by reducing a given instance to a random instance of the hidden hyperplane problem is that there is an efficient oracle algorithm that, given black-box access to a hidden hyperplane distribution corresponding to a fixed vector $\mathbf{u}$, produces samples according to the distribution corresponding to a random rotation $\mathbf{Qu}$.) So, if we can solve the hidden hyperplane problem for a non-negligible fraction of the "hidden" vectors $\mathbf{u}$, then we can efficiently solve it for all vectors $\mathbf{u}$ with very high probability. Equivalently, solving the hidden hyperplane problem on the average is at least as hard as solving uSVP$_\gamma$ in the worst case. From the average-case hardness of the hidden hyperplane problem it is easy to derive a secure public key cryptosystem for single bit messages (longer messages can be encrypted bit by bit):

- the secret key is a random vector $\mathbf{u}$
- the public key is a set of polynomially many points $\mathbf{p}_1, \ldots, \mathbf{p}_n$ chosen at random nearby the hyperplanes defined by $\mathbf{u}$
- The bit 0 is encrypted by adding up a small subset of the public vectors, and perturbing it by a small amount of noise. Notice that this results in a vector distributed at random nearby the hyperplanes.
- The bit 1 is encrypted by picking a random point in space, which, with high probability, will be far from the hyperplanes.

Notice that on the one hand the encrypted bit can be efficiently recovered using the secret key $\mathbf{u}$ by computing the scalar product $\langle \mathbf{u}, \mathbf{c} \rangle$, where $\mathbf{c}$ is the received cipher-text. If $\mathbf{c}$ is the encryption of 0, then the product will be close to an integer, while if $\mathbf{c}$ is the encryption of 1, the product will be close to $\mathbb{Z} + \frac{1}{2}$ with high probability. On the other hand, distinguishing encryptions of 0 from encryptions of 1 is essentially a random instance of the hidden hyperplane problem, which is hard to solve on the average. So, the encryption scheme is secure based on the worst-case hardness of uSVP_γ. We remark that many applications require a stronger notion of security than the CPA-security considered here. Informally, CPA-security corresponds to passive adversary that can eavesdrop, but not alter, the transmitted ciphertexts. Security with respect to active adversaries that are able to alter transmitted messages, and trick the legitimate receiver into decrypting adversarially chosen ciphertexts (called security against *chosen ciphertext attacks*, or CCA-security), is often desirable, but no lattice based public key encryption scheme (provably secure in the standard model of computation) achieving CCA security is currently known.

The Ajtai-Dwork cryptosystem introduces the possibility of decryption errors: it is possible that the bit 1 is encrypted (by chance) as a vector close to the hyperplanes, which would subsequently be decrypted as 0. This problem can be easily solved (as suggested by Goldreich, Goldwasser and Halevi, [13]) by encrypting 0's and 1's as points close to alternating sequences of hyperplanes. Another problem with Ajtai-Dwork cryptosystem is that it relies on the hardness of uSVP_γ for a fairly large value of γ. This has also been substantially improved by Regev [36] to $\gamma = n^{1.5}$ using Fourier analysis techniques similar to those described in Section 3. Regev [36] also shows that the hidden hyperplane problem can be reduced to a one-dimensional problem, yielding a subset-sum style cryptosystem where the public vectors $\mathbf{p}_1, \ldots, \mathbf{p}_n$ are replaced by single numbers $p_1, \ldots, p_n$. The use of numbers p_i rather than vectors $\mathbf{p}_i$ seems to be essentially a matter of style, without much of an impact on performance, because the numbers p_i require very high precision.

The most important (and unsolved) theoretical problem raised by [5] is whether public key encryption can be built from the assumption that SVP_γ, SIVP_γ or other standard lattice problem is hard to approximate in the worst case over arbitrary lattices, rather than the special uSVP_γ lattices with unique shortest vector.

A very interesting result in this direction is another public-key cryptosystem of Regev [37], which can be proved secure under the assumption that no *quantum* algorithm can efficiently approximate SVP_γ (in the worst case over arbitrary lattices) within polynomial factors. We remark that the cryptosystem of [37] is entirely classical: encryption and decryption can be efficiently implemented on standard computers. Only the reduction from SVP_γ to the problem of breaking the cryptosystem involves quantum computation.

5 Concrete security of lattice based cryptography

The importance of basing lattice cryptography on worst-case complexity assumptions cannot be overemphasized. The worst-case approximation factor achieved by lattice reduction algorithms [21, 40] is known to be exponential in the dimension of the lattice [21, 2]. However, lattice reduction algorithms have often been reported to perform much better in practice than the worst-case theoretical upper bound, when run on random problem instances like those arising in cryptanalysis applications. (See also [32] for a recent empirical study showing that the approximation factor achieved by the LLL algorithm on the average, when the input lattice is random, is still exponential in the rank, but with a much smaller constant in the exponent than the worst-case factor.) So, while it seems reasonable to conjecture, based on our current knowledge, that many computational problems on lattices may be hard to approximate in the worst-case even within moderately large factors, extreme caution should be exercised when making conjectures on the average-case complexity of the same problems. In summary, if worst-case intractability assumptions are preferable to average-case ones in general, this is especially true when dealing with point lattices.

The use of worst-case intractability assumptions also frees us from a major burden associated to making average-case complexity conjectures: finding appropriate distributions on problem instances for which no heuristics perform satisfactorily. In the case of lattice problems, heuristics (e.g., [43, 44]) seem to perform reasonably well in practice, and the lattices demonstrating the worst-case performance of known algorithms [21, 2] seem more pathological examples than a source of real practical concern. Worst-case intractability assumptions do not require the selection of an appropriate distribution over input lattices, and the performance of heuristic approaches does not say much about the validity of the assumption. Since the conjecture is a worst-case intractability one, if the algorithm fails to achieve good approximation guarantees even on just a few examples, the conjecture still stands.

All this seems good news for the cryptographic designer, but it also raises new issues when it comes to instantiating the function with specific values of the security parameter. Traditionally, the concrete hardness of average-case problems has been evaluated through a challenge/cryptanalysis process: the cryptographic designer proposing a new intractability assumption produces a list of random problem instances (for increasing values of the security parameter) as a challenge to the rest of the cryptography community. If cryptanalysts are successful in breaking a challenge, the corresponding value of the security parameter is considered too small to achieve security in practice. Typically, there is a cost associated to breaking challenges in varying dimension, which can be experimentally evaluated for moderately small values of the security parameter, and extrapolated to larger values as an estimate of the time required to break bigger instances. (See for example [22].)

Now consider worst-case intractability assumptions. If heuristic approaches to lattice cryptanalysis cannot be used to disprove worst-case intractability assumptions, how can we possibly gain confidence on the validity of the assumption? And how can we select appropriate values of the security parameter to be used in practice? Several approaches come to mind:

1. The *worst-case challenge* approach: we ask the cryptographer to come up with a list of (not necessarily random) challenge problems. Since the cryptographer has complete freedom in the choice of the challenges, he can choose the *worst* problem instances that are hardest to solve by any algorithm. The problem with this approach is that the cryptographer may not know how to select hard instances. In fact, worst-case intractability assumptions do not

even require such hard instances to be easy to find. One of the nicest features of worst-case intractability assumptions is that they do not require to find such hard instances. By asking the cryptographer to come up with hard challenges, the advantage of basing cryptography on worst-case problems would be largely lost.

2. The *reverse challenge* approach: after the cryptographer formulates a worst-case intractability conjecture, the cryptanalyst produces a challenge to the conjecture in the form of a heuristic algorithm and a claim on its worst-case performance. The challenge is for the cryptographer to come up with an input instance for which the heuristic algorithm does not meet the promised performance bound.

3. *Direct cryptanalysis* of the cryptographic function: instead of evaluating the strength of the underlying worst-case complexity assumption, directly attack the cryptographic application.

We remark that the second approach, although unconventional, seems at least more appropriate than the first one. Since the intractability assumption is that no efficient algorithm solves every single problem instance, in an experimental setting it seems more appropriate to first fix the algorithm and then search for the input that triggers the worst case performance, rather then the other way around. Still, producing a counterexample each time somebody comes up with a new heuristic algorithm with worst-case performance claims may seem too much of a burden for the cryptographers.

A disadvantage of both the first and second approach is that it does not necessarily give an indication of the true security of the cryptographic scheme. It may well be the case that breaking the cryptographic function is even harder than solving the worst case problem underlying the security proof. This issue is addressed by the last approach, which focuses directly on the cryptographic function rather than the underlying complexity assumption. This last approach has also the advantage that the proof of security already gives a precise probability distribution according to which challenges should be chosen. For example, in the case of lattice based hash functions, for any set of parameters n, m, q, the challenge should be a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ chosen uniformly at random. There is no need to have the cryptographer to come up with a list of challenges. The cryptanalyst can select the challenges on her own, as long as the challenge is selected at random according to the prescribed probability distribution.

A possible objection to the last approach is that it bypasses the proof of security. If the concrete security of the cryptographic function is evaluated directly by cryptanalysis, what is the value of providing a security proof? We believe that even in this scenario, security proofs are very valuable. They may not help in assessing the concrete security of the function for specific values of the security parameter, but they ensure that the construction has no structural flaw, and (in the case of worst-case to average-case reductions) they provide invaluable guidance when selecting the appropriate distribution over random keys.

A more serious disadvantage of the last approach is that any new cryptographic primitive requires a separate cryptanalytic effort to assess its concrete security. This way, one of the main advantages of the provable security approach is lost: the ability to concentrate cryptanalytic efforts on a small number of selected computational problems, and derive the (concrete) security of old and new cryptographic constructions by means of mathematical proofs.

One possible way to address some of the issues brought up in the previous discussion is to study the tightness of worst-case to average-case security proofs. E.g., we know that breaking the cryptographic hash functions described in Section 3 with non-negligible probability is at least

as hard as approximating SIVP_γ within a factor $\tilde{O}(n)$. Is it possible to prove also that given an oracle that solves SIVP_γ within a factor $\tilde{O}(n)$ (in the worst-case) one can find collisions to the hash function for randomly chosen keys? Some results of this kind are proved by Nguyen and Stern [33], who show that solving SVP_γ within a factor $n^{0.5-\epsilon}$ allows to break the Ajtai-Dwork cryptosystem. However, there are several gaps between the two reductions: in [33] the worst-case lattice problems are different (SVP_γ or CVP_γ, rather than SIVP_γ or GapSVP_γ), the required polynomial approximation factors are smaller, and the lattice dimension is larger than in the proof of security. Giving a tight worst-case to average-case reduction between lattice problems is an important open problem as it would allow to evaluate the concrete worst-case complexity of lattice problems by cryptanalyzing random challenges of the average-case problem. Well understood worst-case assumptions could then be used to prove the security of other average-case problems, without the need of new challenges and cryptanalysis.

We conclude the paper with a discussion of two other issues that have received little attention so far, but are certainly very important in the context of cryptanalysis of lattice based functions. The first issue is that most lattice cryptanalysis problems are more naturally formulated as lattice approximation problems in the ℓ_∞ norm. For example, we observe that finding collisions to the hash function described in Section 3 is exactly the same as solving SVP_γ in the ℓ_∞ norm in the lattice naturally associated to the function. Still, most algorithmic work has focused on the Euclidean norm ℓ_2. While the ℓ_2 norm may be more convenient when designing algorithms, there are several reasons to prefer the ℓ_∞ norm when working with applications of lattices: the ℓ_∞ norm often leads to faster and easier to implement functions, and there is theoretical evidence [11, 39] that lattice problems in the ℓ_∞ norm are harder (or at least not easier) than the same problems in the Euclidean norm. For example, [39] gives reductions from many lattice problems in the ℓ_2 norm to the same problems in the ℓ_∞ norm. Moreover, hardness results for SVP_γ in the ℓ_2 norm lag behind similar results in the ℓ_∞ norm. For example, SVP_γ in the ℓ_∞ norm can be shown to be hard to approximate within almost polynomial factors $n^{1/O(\log\log n)}$ under the assumption that NP $\neq$ P [11]. A similar result can be proved for the ℓ_2 norm [16], but only under the much stronger assumption that NP does not have randomized sub-exponential time algorithms.

Despite the importance of the ℓ_∞ norm in cryptanalysis, the study of lattice reduction algorithms in norms different from ℓ_2 has seen little progress so far. For example, the lattice reduction algorithm of [23] for general norms is not even known to terminate in polynomial time for arbitrary dimension.

A second very interesting issue that requires further investigation is the complexity of lattice problems when restricted to special classes of lattices, as those arising in the design of lattice based public key cryptosystems, and efficient hash functions. Most work on lattice algorithms has been focused on the solution of SVP_γ or other lattice problems in the ℓ_2 norm on arbitrary lattices. (See [12] for an interesting exception.) We remark that the class of lattices underlying the construction of efficient one-way hash functions and public key encryption schemes is restricted in very different ways:

- In the case of one-way functions, the restriction is, in a sense, *algebraic*, as clearly illustrated by the most recent work on collision resistance and ideal lattices [35, 24, 25].
- In the case of public key encryption, the restriction is more *geometric* (e.g., there is a gap between λ_1 and λ_2).

Here we may call *geometric* those properties that are (approximately) preserved under small perturbations of the lattice. It is easy to see that even a small perturbation to a cyclic or ideal lattice would

immediately destroy the algebraic structure. Is there any reason to prefer one kind of restriction over the other one? Ajtai's work [1] (which essentially conjectures that any non-trivial geometric property of a lattice is hard to decide) seems to express more confidence in the hardness of lattice problems under geometric restrictions. On the other hand, cryptanalysis experiments [34] suggest that lattice reduction algorithms perform better on the average when there is a big gap between λ_1 and λ_2. Improving our understanding of the computational complexity of lattice approximation problems when restricted to special classes of lattices, as well as classifying the restrictions into meaningful categories (like, algebraic or geometric restrictions) is a very interesting open problem, both from a structural point of view (identifying properties that make computational problems on lattices easier or harder to solve) and for the cryptographic applications of lattices.

References

[1] M. Ajtai. Random lattices and a conjectured 0-1 law about their polynomial time computable properties. In *Proceedings of the 43rd annual symposium on foundations of computer science - FOCS 2002*, pages 733–742, Vancouver, British Columbia, Canada, Nov. 2002. IEEE.

[2] M. Ajtai. The worst-case behavior of Schnorr's algorithm approximating the shortest nonzero vector in a lattice. In *Proceedings of the thirty-fifth annual ACM symposium on theory of computing - STOC 2003*, pages 396–406, San Diego, CA, USA, June 2003. ACM.

[3] M. Ajtai. Generating hard instances of lattice problems. *Complexity of Computations and Proofs, Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[4] M. Ajtai. Representing hard lattices with O(n log n) bits. In *Proceedings of the thirty-seventh annual ACM symposium on theory of computing - STOC 2005*, pages 94–103. ACM, June 2005.

[5] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on theory of computing - STOC '97*, pages 284–293, El Paso, Texas, USA, May 1997. ACM.

[6] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the thirty-third annual ACM symposium on theory of computing - STOC 2001*, pages 266–275, Heraklion, Crete, Greece, July 2001. ACM.

[7] L. Babai. On Lovasz' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[8] E. F. Brickell and A. M. Odlyzko. Cryptanalysis: A survey of recent results. In G. J. Simmons, editor, *Contemporary Cryptology*, chapter 10, pages 501–540. IEEE Press, 1991.

[9] J.-Y. Cai. A relation of primal-dual lattices and the complexity of the shortest lattice vector problem. *Theoretical Computer Science*, 207(1):105–116, Oct. 1998.

[10] J.-Y. Cai and A. P. Nerurkar. An improved worst-case to average-case connection for lattice problems (extended abstract). In *Proceedings of the 38th annual symposium on foundations of computer science - FOCS '97*, pages 468–477, Miami Beach, Florida, USA, Oct. 1997. IEEE.

[11] I. Dinur. Approximating SVP_∞ to within almost-polynomial factors is NP-hard. *Theoretical Computer Science*, 285(1):55–71, 2002.

[12] N. Gama, N. Howgrave-Graham, and P. Nguyen. Symplectic lattice reduction and NTRU. In S. Vaudenay, editor, *Advances in cryptology - EUROCRYPT 2006, proceedings of the international conference on the theory and application of cryptographic techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 233–253, St. Petersburg, Russia, May 2006. Springer.

[13] O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In B. S. Kaliski, Jr., editor, *Advances in cryptology - CRYPTO '97, Proceedings of the 17th annual international cryptology conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111, Santa Barbara, California, USA, Aug. 1997. Springer.

[14] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[15] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sience*, 28(2):270–299, 1984. Preliminary version in Proc. of STOC 1982.

[16] I. Haviv and O. Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, 2007. To appear.

[17] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In J. P. Buhler, editor, *Algorithmic number theory: Third international symposium - ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288, Portland, OR, USA, June 1998. Springer.

[18] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on the theory of computing - STOC '89*, pages 44–61, Seattle, Washington, USA, May 1989. ACM.

[19] A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

[20] R. Kumar and D. Sivakumar. On polynomial-factor approximations to the shortest lattice vector length. *SIAM Journal on Discrete Mathematics*, 16(3):422–425, 2003.

[21] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[22] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[23] L. Lovász and H. Scarf. The generalized basis reduction algorithm. *Mathematics of Operations Research*, 17(3):754–764, 1992.

[24] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In I. Wegener, V. Sassone, and B. Preneel, editors, *Proceedings of the 33rd international colloquium on automata, languages and programming - ICALP 2006*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155, Venice, Italy, July 2006. Springer.

[25] V. Lyubashevsy, D. Micciancio, C. Peikert, and A. Rosen. Provably secure FFT hashing. In *2nd NIST cryptographic hash workshop*, Santa Barbara, CA, USA, Aug. 2006.

[26] D. Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004. Preliminary version in STOC 2002.

[27] D. Micciancio. Generalized compact knapsaks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 2007. To appear in special issue on average-case complexity. Preliminary version in FOCS 2002.

[28] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[29] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measure. *SIAM Journal on Computing*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[30] D. Micciancio and S. Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In D. Boneh, editor, *Advances in cryptology - CRYPTO 2003, proceedings of the 23rd annual international cryptology conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 282–298, Santa Barbara, California, USA, Aug. 2003. Springer.

[31] P. Nguyen and D. Stehlé. Floating-point LLL revisited. In R. Cramer and I. Damgard, editors, *Advances in cryptology - EUROCRYPT 2005, proceedings of the international conference on the theory and application of cryptographic techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233, Aarhus, Denmark, May 2005. Springer.

[32] P. Nguyen and D. Stehlé. LLL on the average. In F. Hess, S. Pauli, and M. E. Pohst, editors, *Algorithmic number theory: 7th international symposium - ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 238–256, Berlin, Germany, July 2006. Springer.

[33] P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In H. Krawczyk, editor, *Advances in cryptology - CRYPTO '98, Proceedings of the 18th annual international cryptology conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242, Santa Barbara, California, USA, Aug. 1998. Springer.

[34] P. Nguyen and J. Stern. The two faces of lattices in cryptology. In J. Silverman, editor, *Cryptography and lattices conference – CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180, Providence, RI, USA, Mar. 2001. Springer.

[35] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In S. Halevi and T. Rabin, editors, *Theory of cryptography conference - Proceedings of TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166, New York, NY, USA, Mar. 2006. Springer.

[36] O. Regev. New lattice based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in STOC 2003.

[37] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on theory of computing - STOC 2005*, pages 84–93. ACM, June 2005.

[38] O. Regev. On the complexity of lattice problems with polynomial approximation factors. 2007. These proceedings.

[39] O. Regev and R. Rosen. Lattice problems and norm embeddings. In *Proceedings of the thirty-eigth annual ACM symposium on theory of computing - STOC 2006*, pages 447–456. ACM, June 2006.

[40] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2–3):201–224, 1987.

[41] C.-P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, Mar. 1988.

[42] C. P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, Jan. 2006.

[43] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, Aug. 1994. Preliminary version in FCT 1991.

[44] C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in cryptology - EUROCRYPT '95, Proceedings of the international conference on the theory and application of cryptographic techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12, Saint-Malo, France, May 1995. Springer.