Universiteit
Leiden
The Netherlands

Lattice cryptography: from cryptanalysis to New Foundations

Woerden, W.P.J. van

Citation
Woerden, W. P. J. van. (2023, February 23). *Lattice cryptography: from cryptanalysis to New Foundations*. Retrieved from https://hdl.handle.net/1887/3564770

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3564770

Note: To cite this publication please use the final published version (if applicable).

Part I

Introduction and Preliminaries

CHAPTER 1

Introduction

The physical world is inherently noisy. An integer vector $\mathbf{x} \in \mathbb{Z}^d$ transmitted over a channel may only be received as some distorted vector $\mathbf{x} + \mathbf{e} \in \mathbb{R}^d$. If the Euclidean norm of the error $\mathbf{e}$ is strictly less than $\frac{1}{2}$, we know that simply rounding the coordinates will recover the original vector. If the error is larger we cannot be sure that the original vector is recovered. In two dimensions one can visually interpret this error correction as the integer *sphere packing* as shown on the left in Figure 1.1. The radius of the spheres corresponds to the largest error from which we can correctly recover the original message. The spheres

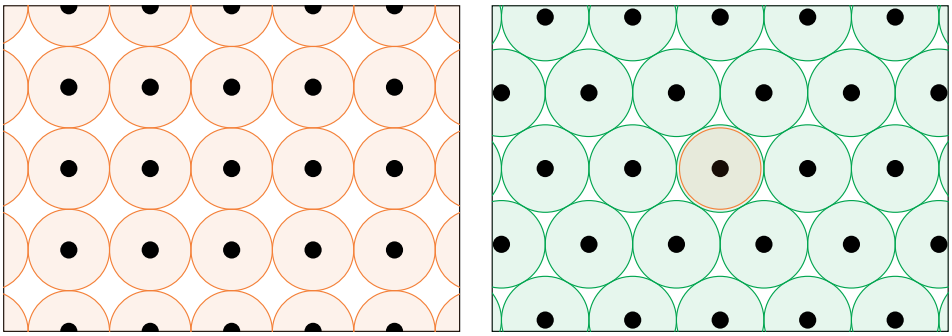


Figure 1.1: The integer and hexagonal sphere packings, with on average one sphere per unit area.

are not allowed to overlap, as that would imply ambiguity on what the original message was.

More generally, any sphere packing gives a way to recover from noise, by encoding each distinct message as the center of one of the spheres. The radius of the spheres then determines the robustness from noise. The number of spheres per fixed volume determines the (relative) number of distinct messages that can be encoded by a single signal, and thereby determines the capacity of the connection. A *denser* packing improves the robustness, while maintaining the same capacity. For example, the densest packing in dimension 2, namely the hexagonal packing in Figure 1.1, allows to recover from errors that are 7% larger than for the integer packing. The gain in dimension 2 may seem small, but in larger dimensions d this radius can increase from $\frac{1}{2}$ to $\Theta(\sqrt{d})$.

The densest sphere packing in dimension 2 follows a repetitive pattern, just as the integer packing. Both are examples of *lattice packings*, i.e., packings where the centers of the spheres form a (discrete) additive group, or a *lattice*. Such packings can be efficiently described by a generating set of vectors. This regularity is not unique to dimension 2, the densest known packings in dimensions up to 9, and in dimension 24, are attained by lattice packings. Also for higher dimensions lattice packings can be very dense. Their efficient description, and often high density is what makes lattice packings useful for error correcting purposes.

Having a dense lattice packing however is not enough, one also needs to be able to efficiently *decode* the error and recover the original message, such as the rounding algorithm for the integer lattice. While in general this is a very hard problem when the dimension grows to hundreds or even thousands, there do exist dense lattice packings with enough structure to decode efficiently up to a large radius. These well decodable dense lattice packings can thus be used for reliable and efficient communication over noisy channels.

Next to being noisy, the physical world is also inherently unsafe. Letters may be read, conversations can be overheard and wired or wireless channels may be tapped by an eavesdropper. To protect such communication we require *cryptography*. Cryptography allows to hide, or *encrypt* a message, such that no eavesdropper can learn what is communicated, in such a way that the desired receiver can still recover,

or *decrypt* the original message. Modern cryptography can roughly be divided into two categories. If the communicating parties know a common secret key, then there are efficient ways to encrypt and decrypt the message. This is known as *symmetric* cryptography. In case the parties have no common key, we require *asymmetric*, or *public-key* cryptography. In public-key encryption there are two keys, one secret key, known only to the receiver, and one public key, known to all and in particular the sender. A message can be encrypted using the public key, and decrypted using the secret key.

Again, lattice packings and noise can play a central role in building such public-key cryptography. In this case the noise is added deliberately to a message, and the security relies on the general hardness of removing such noise. The idea is to have a good and a bad description of the same lattice, the good one allows to efficiently decode, while the bad one does not. Naturally the good description forms the secret key, while the bad description forms the public key. Someone with only the bad description can 'hide' a message encoded as a lattice point by adding a small error to it, after which the receiver uses the good description to decode the error and recover the original message. To make this safe the good description should be hidden from everyone but the receiver, which for example can be attained by a randomized procedure that generates a lattice packing along with such a good description.

The main advantage of lattice-based cryptography is that, in contrast to currently used public-key cryptography (variants of RSA and Diffie-Hellman), it is believed to be hard to break even for quantum computers. Lattice-based cryptography is currently the main candidate to replace such quantum vulnerable cryptography. It is therefore of fundamental importance to study the (concrete) hardness of these constructions and underlying problems, and to search for potential weaknesses. Such research is better known as *cryptanalysis*, and it is the main topic of Parts II and III of this thesis.

The state-of-the-art cryptanalysis, to which we also contribute in this thesis, shows that the hardness is directly related to the geometry and efficient decoding capability of the lattice packing. The influence of the decoding capability is intuitive: the more noise we can add to the message the better it is hidden, and the harder it is to recover by an eavesdropper. The precise influence of the geometry of the lattice

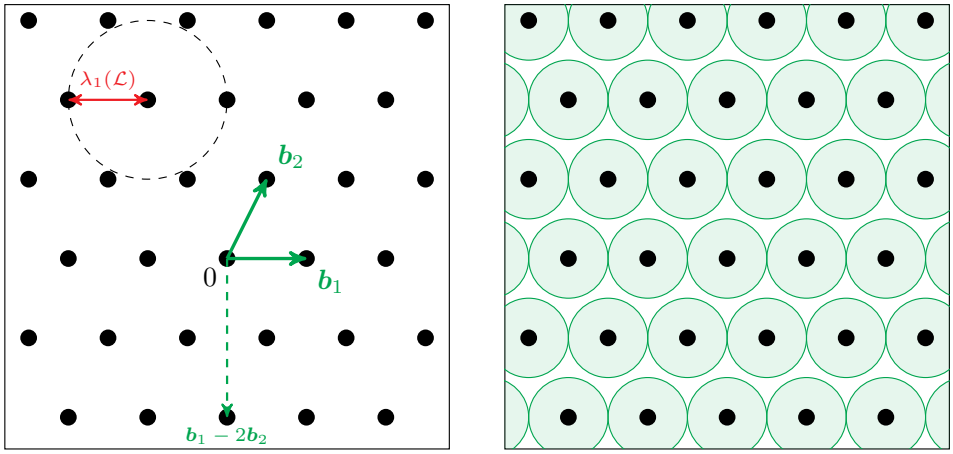


Figure 1.2: The lattice $\mathcal{L}$ with basis $[\mathbf{b}_1, \mathbf{b}_2] = [(1, 0), (\frac{1}{2}, 1)]$ and first minimum $\lambda_1(\mathcal{L}) = 1$. By adding balls of radius $\lambda_1(\mathcal{L})/2$ around each lattice point we obtain a sphere packing.

packing on the security can be much more subtle, and also needs to be taken into account.

Unfortunately, when we look at the current landscape of lattice-based cryptography, through the lens of cryptanalysis, we recognise a fundamental weakness. All currently used lattice packings have a rather poor density or decoding capability, and this is inherent to the randomized procedures that are used to generate a lattice packing along with a good description. In fact, these packings and their decoding algorithms geometrically resemble those of the integer lattice. The error added to hide a message is thus a factor $\Theta(\sqrt{d})$ smaller than optimal, which leads to weaknesses. To compensate for this, one has to increase the dimension, which in turn hurts the efficiency of currently used lattice-based cryptography.

We identify a missed opportunity: denser and efficiently decodable lattice packings are used for error correcting purposes, but not in cryptography. The main obstacle is that the remarkable structure of such lattice packings is assumed to be known to everyone, and thus at first may seem impossible to hide. In Part IV of this thesis we introduce a new foundation, based on lattice isometries, that does allow to hide the remarkable structure, and therefore does enable the use of such dense and efficiently decodable lattice packings in cryptography.

Lattices

In this thesis we define a lattice as a discrete additive subgroup $\mathcal{L} \subset \mathbb{R}^d$ of $\mathbb{R}^d$. An example of a lattice is the subgroup $\mathbb{Z}^d \subset \mathbb{R}^d$ of integer vectors. More generally, given $\mathbb{R}$-linearly independent vectors $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times d}$ the subgroup $\mathcal{L}(\mathbf{B}) := \{\sum_i x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$ consisting of all integer combinations is a lattice. In fact, every lattice can be described by such a basis $\mathbf{B}$, and the number n of basis vectors defines the rank of a lattice. Throughout this introduction we assume that all lattices have full rank $n = d$.

The discrete nature of a lattice implies that there is some minimum distance $\lambda_1(\mathcal{L}) > 0$ between any two distinct lattice points. Due to the additive structure of the lattice we can always assume that one of the two points is the origin $\mathbf{0} \in \mathcal{L}$, and thus $\lambda_1(\mathcal{L}) := \min_{\mathbf{0} \neq \mathbf{v} \in \mathcal{L}} \|\mathbf{v}\|$ can equivalently be defined as the minimum length of any nonzero lattice vector. For example, we have $\lambda_1(\mathbb{Z}^n) = 1$, which is attained by any unit vector $(0, \ldots, 0, \pm 1, 0, \ldots, 0) \in \mathbb{Z}^n$. Geometrically, we can place open balls around each lattice point of radius $\lambda_1(\mathcal{L})/2$ without overlapping them. This turns any lattice into a sphere packing as illustrated in Figure 1.2.

Cryptography and the Quantum Threat

A public-key cryptosystem enables an individual, such as Alice, to securely send a message to another individual, such as Bob, across an untrusted channel so that only Bob can read the message. Any eavesdropper, such as Eve, should not be able to learn Alice's message.

The message Alice wishes to send is encrypted with a public key **pk** that Bob previously announced publicly. She then sends Bob the encrypted message $\mathbf{Enc}_{\mathbf{pk}}(\text{message})$ via the unsafe channel, and Bob uses his secret key **sk** to decrypt it and recover the message (see Figure 1.3).

For security it must be computationally hard to derive the secret key **sk** from the public key. While decryption only works if the secret key and public key are somehow related. In particular the secret key should allow Bob to compute something that no one else can do efficiently.

Historically, the most famous examples of a public-key cryptosystems are Diffie-Hellman and RSA. Almost all currently used public-key cryptography is based on (variants) of these systems. For RSA, the secret key consists of two large primes $\mathbf{sk} = (p, q)$, and the public key is given by $\mathbf{pk} = N = p \cdot q$. To obtain the secret key from the public key one needs to factor N, which is assumed to be computationally hard in the bit-length $\log(N)$. Consider the function $f(x) := x^e \bmod N$ for some exponent $1 < e < \phi(N)$ such that $\gcd(e, \phi(N)) = 1$. Knowledge of the secret key p, q allows one to compute $\phi(N)$ and thus $d := e^{-1} \bmod \phi(N)$. One can then efficiently invert $f(x)$ because $f(x)^d \equiv x \bmod N$. Without the secret key this is seemingly hard[1]. The secret key thus creates a *trapdoor* function f: one can encrypt a message $m \in (\mathbb{Z}/N\mathbb{Z})^*$ using the public key $\mathbf{pk} = N$ as $c = f(m) = m^e \bmod N$, and the secret key $\mathbf{sk} = (p, q)$ allows to invert f and thus recover the original message.

While classically both the Diffie-Hellman and the RSA cryptosystems are yet unbroken, quantumly they are not. Shor's algorithm [Sho94] solves integer factorization and the discrete logarithm problem (underlying Diffie-Hellman) in polynomial time on a quantum computer. Given the recent advent of larger and more stable quantum-computers, we need to replace these conventional problems by new ones that are resistant to quantum attacks, but still allow to create trapdoor functions.

Hard Lattice Problems

To build a trapdoor function we require hard problems, and for lattices there are some natural candidates. The most well-known and simplest to state problem is the Shortest Vector Problem (SVP). As the name suggests it asks you to compute a shortest nonzero lattice vector $\mathbf{v} \in \mathcal{L}$ of length $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$. The input of this problem is any basis of the lattice. If such a basis already consists a shortest lattice vector then the problem is trivial, thus in general you can expect to receive a basis consisting of long vectors.

[1] Technically, knowing p, q is not a necessary condition for breaking RSA, but it is a sufficient one. There do exist (inefficient) variants of RSA where the two are equivalent [Rab79].

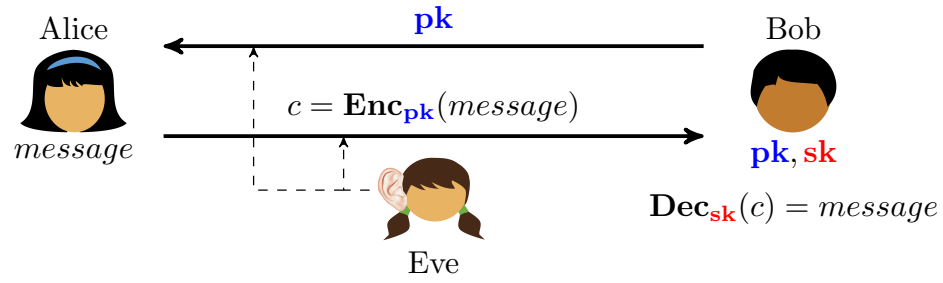


Figure 1.3: A public-key cryptosystem allows Alice to send a message to Bob, such that an eavesdropper Eve cannot learn the message.

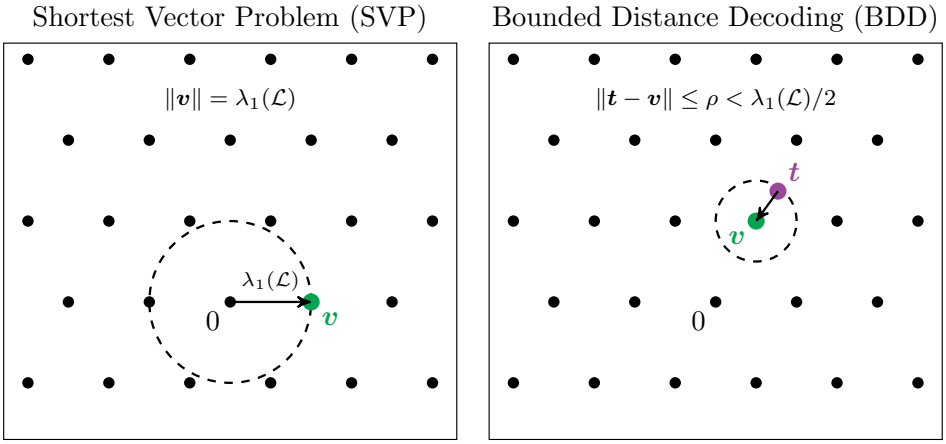


Figure 1.4: Two important lattice problems: the Shortest Vector Problem and Bounded Distance Decoding.

The inhomogeneous variant of SVP is the Closest Vector Problem (CVP). Recall that the lattice $\mathcal{L} \subset \mathbb{R}^n$ lives in a real space $\mathbb{R}^n$. Given any target $\mathbf{t} \in \mathbb{R}^n$ the Closest Vector Problem asks you to compute a lattice vector $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{t} - \mathbf{v}\|$. Note that if the distance of $\mathbf{t}$ to a closest lattice vector is strictly less than $\lambda_1(\mathcal{L})/2$ then the solution is unique. Moreover, to simplify the problem one could get the additional promise that the given target lies at distance at most some $\rho < \lambda_1(\mathcal{L})/2$ from the lattice — this is known as Bounded Distance Decoding (BDD). Intuitively, if ρ is very small, i.e., the target lies very close to some lattice vector, then it is easier to recover that lattice

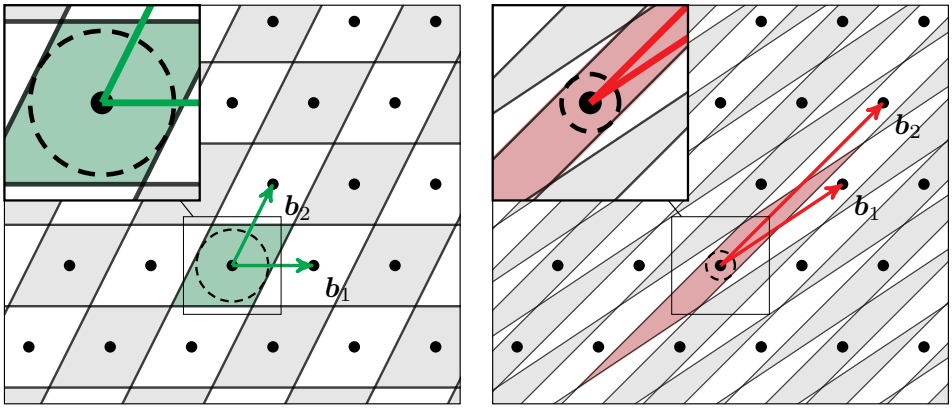


Figure 1.5: Radius for which the basis rounding algorithm solves BDD, for a good basis (left) versus a bad basis (right).

vector.

In two or even three dimensions the above problems are easy, certainly when looking at the examples in Figure 1.4. However the hardness of these problems, that is needed for cryptography, comes from increasing the dimension n to hundreds or even thousands.

Lattice-based Cryptography

Every lattice $\mathcal{L}(\mathbf{B})$ of dimension at least 2 can be described by an infinite number of bases $\mathbf{B}\cdot\mathbf{U}$ for unimodular $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})$, i.e., integer $\mathbf{U} \in \mathbb{Z}^{n\times n}$ with $\det(\mathbf{U}) = \pm 1$. We call a basis *good* if it consists of short and somewhat orthogonal vectors, and *bad* if the vectors are long. The hardness of the Shortest Vector Problem indicates that given a bad basis it is hard to compute a good basis. We will use the good and bad bases to create a trapdoor function.

Any basis gives a decoding algorithm. For a target $\mathbf{t} = \mathbf{By} \in \mathbb{R}^n$ we can simply round the coordinate vector $\mathbf{y}$ to obtain a lattice vector $\mathbf{v} = \mathbf{B}\cdot\lfloor\mathbf{y}\rceil \in \mathcal{L}(\mathbf{B})$. Then the error $\mathbf{e} := \mathbf{t} - \mathbf{v}$ lies in the fundamental parallelepiped $\mathcal{P}(\mathbf{B}) := \mathbf{B}\cdot[-\frac{1}{2},\frac{1}{2})^n$ of $\mathbf{B}$. For any $\mathbf{e} \in \mathcal{P}(\mathbf{B})$ we have $\|\mathbf{e}\| \leq \|\mathbf{B}\|\cdot\frac{1}{2}\sqrt{n}$, and thus the shorter the basis, the closer the lattice vector $\mathbf{v}$ lies to $\mathbf{t}$.

Furthermore, let ρ be the largest radius of a ball that is inscribed

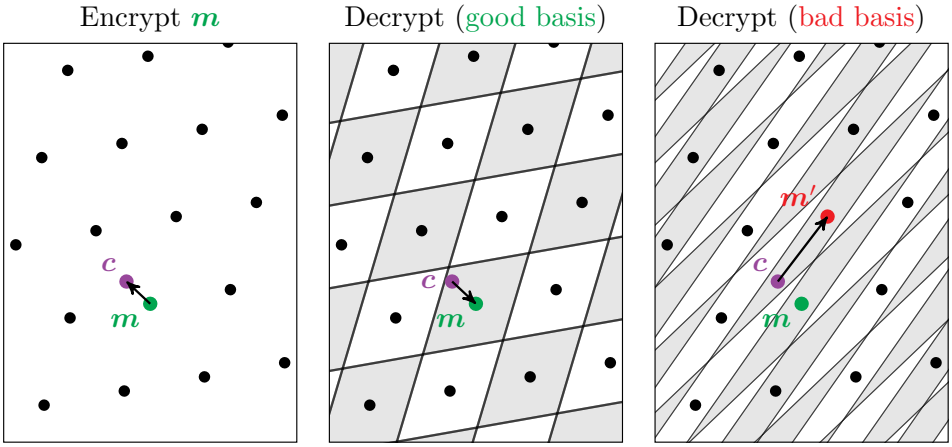
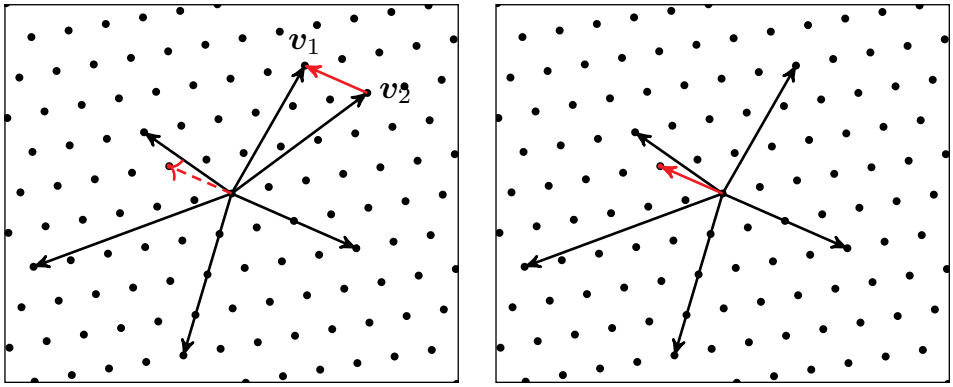


Figure 1.6: Encryption of a message $\mathbf{m} \in \mathcal{L}$ by adding a small error $\mathbf{e}$. Rounding with the good (secret) basis recovers $\mathbf{m}$, while rounding with the bad (public) basis recovers the wrong message $\mathbf{m}' \neq \mathbf{m}$.

in $\mathcal{P}(\mathbf{B})$. Then any BDD instance $\mathbf{t} \in \mathbb{R}^n$ at distance at most ρ from the lattice is decoded correctly to its unique closest vector $\mathbf{c} \in \mathcal{L}$ by the rounding algorithm. For a good basis the decoding radius ρ is generally large, while for a bad basis it is small, see Figure 1.5.

So a (secret) short basis allows us to efficiently recover from small errors, while given a (public) long basis this is a hard problem. The (randomized) trapdoor function now follows naturally. To encrypt a message $\mathbf{m} \in \mathcal{L}$ (described by the bad basis) we simply add a small uniform error $\mathbf{e}$ of length $\|\mathbf{e}\| = \rho$ to the message: $\mathbf{c} := \mathbf{m} + \mathbf{e}$. The rounding procedure together with the secret good basis allows to remove the error and recover $\mathbf{m}$. The same rounding procedure with the bad basis computes a wrong message $\mathbf{m}' \neq \mathbf{m}$ with overwhelming probability for large n, see Figure 1.6. Without the help of a good basis inverting the trapdoor function is a BDD instance, which in general is hard.

The idea behind the good-bad basis encryption scheme is easy to understand, but we did not specify so far how to obtain such a pair of bases. We call this part the key generation and this is precisely where different security assumptions come in. Note that this must be a randomized procedure, as otherwise someone else could just rerun it to obtain the same short basis. In fact, the lattice generated must be



Find close vectors $v_1, v_2 \in L$. Replace $v_2 \leftarrow v_1 - v_2$.

Figure 1.7: Lattice Sieving

random in some sense, as any short basis of the same lattice can be used to decrypt. Almost all of lattice-based cryptography is currently based on a few versatile assumptions that allow one to generate some random lattice along with a (partial) good basis: the Learning With Errors (LWE) assumption, the Short Integer Solution (SIS) assumption and the NTRU assumption. These assumptions are variants of the BDD and SVP problems, for some specific distributions of lattices.

Lattice Sieving and Heuristics

Before considering the security of these schemes directly, we first consider the basic task of computing a shortest vector of a lattice. This will form a building block for later attacks. Lattice sieving algorithms are currently the best way to compute a shortest vector of a lattice in terms of time complexity. They run in single exponential time $2^{O(n)}$ and space $2^{O(n)}$. The central idea of sieving algorithms is to start with a large list of (long) lattice vectors, and to find many sums and differences of these vectors that are shorter. These shorter combinations are inserted back into the list, possibly replacing longer vectors, and this process is repeated until the list contains many short vectors, among which (hopefully) a shortest one of the lattice. One such step is illustrated in Figure 1.7.

The best provable sieving algorithm runs in time $2^{2.456n+o(n)}$ and space $2^{1.233n+o(n)}$ [HPS11b]. Clearly, these algorithms already become infeasible in very low dimensions. However, in practice we can do much better. To understand and analyse these practical sieving algorithms we require the use of heuristics. Such an analysis should be seen as an average-case analysis that might be false in the worst-case. Fortunately, in the cryptanalytic setting these sieving algorithms are executed on lattices that are sufficiently randomized to follow these average-case heuristics. With such a heuristic average-case analysis the best sieving algorithm can be shown to run in time $2^{0.292n+o(n)}$ and space $2^{0.208n+o(n)}$ [BDGL16].

Just as there is a big gap between the complexity of provable and heuristic algorithms, there is also a large knowledge gap between an asymptotic complexity such as $2^{0.292n+o(n)}$ and a concrete estimate for the cost of solving SVP in say dimension 500. For cryptography it is extremely important to understand these concrete costs, as it will directly influence the concrete security.

Cryptanalysis and Basis Reduction

To understand the security of lattice-based schemes we have to know how hard it is to solve the specific lattice problem instances that occur. We have briefly discussed that for a usual average-case lattice the SVP instance in rank n requires about $2^{0.292n+o(n)}$ time to solve. However the lattices that arise from e.g. the LWE, SIS or NTRU assumption are geometrically non-standard. For example they contain either an unusually short vector, a dense sublattice, or the efficient decoding radius ρ is relatively small, see Figure 1.8.

Due to this divergent (secret) structure these instances can be significantly easier than the usual average-case. Often, to break these schemes we only need to find a somewhat short and orthogonal basis. Basis reduction algorithms use an average-case SVP oracle in some lower dimension $\beta \ll n$ to compute such a basis. The larger the parameter β, the better *reduced* the basis gets, but at a cost of solving SVP in dimension β. For large enough β the (secret) structure unravels and the scheme is broken. We observe, that for almost all lattice based schemes that work with n-dimensional lattices, basis reduction

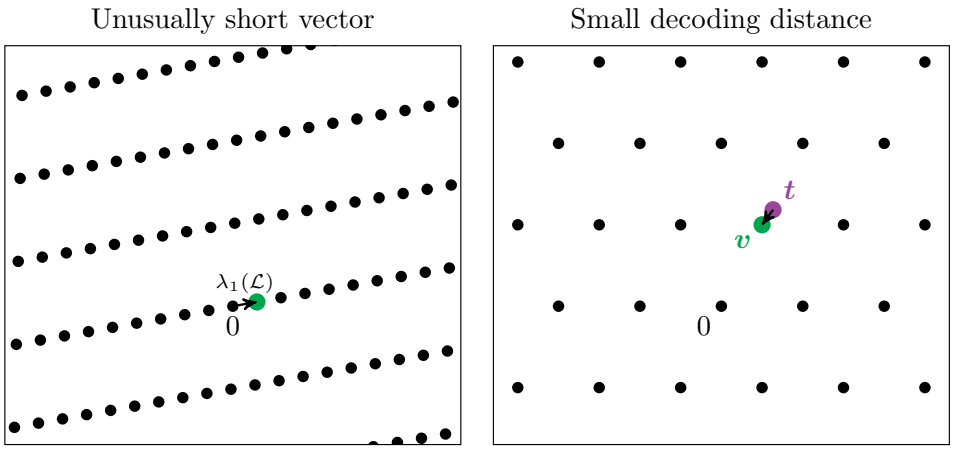


Figure 1.8: On the left a lattice with an unusually short vector (or alternatively a dense rank 1 sublattice), on the right a BDD instance with small decoding radius ρ.

with $\beta \leq n/2 + o(n)$ is sufficient to break them. As a result the lattice dimensions used have to be at least twice as large as what you would expect from the $2^{0.292n+o(n)}$ complexity to solve SVP, leading to schemes with suboptimal performance.

It is not always immediately clear what SVP dimension β is needed to break a scheme. A large task for cryptanalysts is to understand better how the basis reduction algorithm precisely discover the hidden structure. From this understanding we can then derive both asymptotic, but more importantly, concrete estimates for β. In particular, special care should be taken when parameters are taken beyond their usual range, as this might enable new attacks, or the usual attacks might perform better than predicted due to unforeseen weaknesses in the lattice geometry.

The Lattice Isomorphism Problem

Current lattice-based cryptography has several weaknesses that allow to break a scheme based on an n-dimensional lattice by only solving SVP in dimension $\beta \leq n/2 + o(n)$. This stems either from an unbalanced geometry of the lattice (or its dual), from a poor decoding dis-

tance, or from giving away a somewhat good basis. With the currently used techniques of using a trapdoor basis, the decoding distance and unbalanced geometry of the lattice are inherently related. The only way to improve the decoding distance is by making the geometry of the lattice even more unbalanced, and vice versa. The optimal trade-off still falls victim to a basis reduction attack with $\beta = n/2 + o(n)$.

In contrast there do exist wonderful lattices that have efficient decoding algorithms that do not rely on any good basis, but only on the remarkable structure of the lattice in question. Such lattices can offer a much better balance between their decoding distance and their geometry, which might allow to move beyond this $\beta \leq n/2 + o(n)$ barrier.

Now suppose we have such an efficiently decodable lattice with a balanced geometry. Given that (in general) there is no randomness in the generation of this lattice we must assume the lattice and its decoding algorithm is public knowledge. Giving away a bad basis publicly is therefore meaningless, everyone already knows how to decode the lattice. We need some way to hide the remarkable structure of the lattice, without destroying it (as we need it for decoding purposes). The only way to not influence the geometry of a lattice at all is by applying an isometry. This brings us to the lattice isomorphism problem.

Two lattices $\mathcal{L}, \mathcal{L}' \subset \mathbb{R}^n$ are called *isomorphic* or *isometric* if there exists an orthonormal transformation $\mathbf{O} \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}' = \mathbf{O} \cdot \mathcal{L} = \{\mathbf{O} \cdot \mathbf{v} : \mathbf{v} \in \mathcal{L}\}$. See Figure 1.9 for an example of two isomorphic lattices. The Lattice Isomorphism Problem (LIP) asks to compute such a transformation $\mathbf{O}$ given the two lattices $\mathcal{L}, \mathcal{L}'$. The best asymptotic and practical algorithms that solve LIP all require to first compute a shortest vector, even when applied to very simple lattices such as $\mathbb{Z}^n$. LIP therefore gives us a solid foundation for lattice-based cryptography.

We sketch in Figure 1.10 how one would build an encryption scheme build on LIP. First let $\mathcal{L}$ be a lattice in which we can decode efficiently up to some radius $\rho < \lambda_1(\mathcal{L})/2$. We generate some random orthonormal matrix $\mathbf{O} \in \mathcal{O}_n(\mathbb{R})$ and give away a bad basis of the rotated lattice $\mathbf{O} \cdot \mathcal{L}$. The *secret key* is the transformation $\mathbf{O}$, while the *public key* is (some description of) the lattice $\mathbf{O} \cdot \mathcal{L}$. Now to encrypt anyone can take a message $\mathbf{m} \in \mathbf{O} \cdot \mathcal{L}$, and add a small uniformly random error $\|\mathbf{e}\| = \rho$ to the message: $\mathbf{c} := \mathbf{m} + \mathbf{e}$. Without knowing the underlying

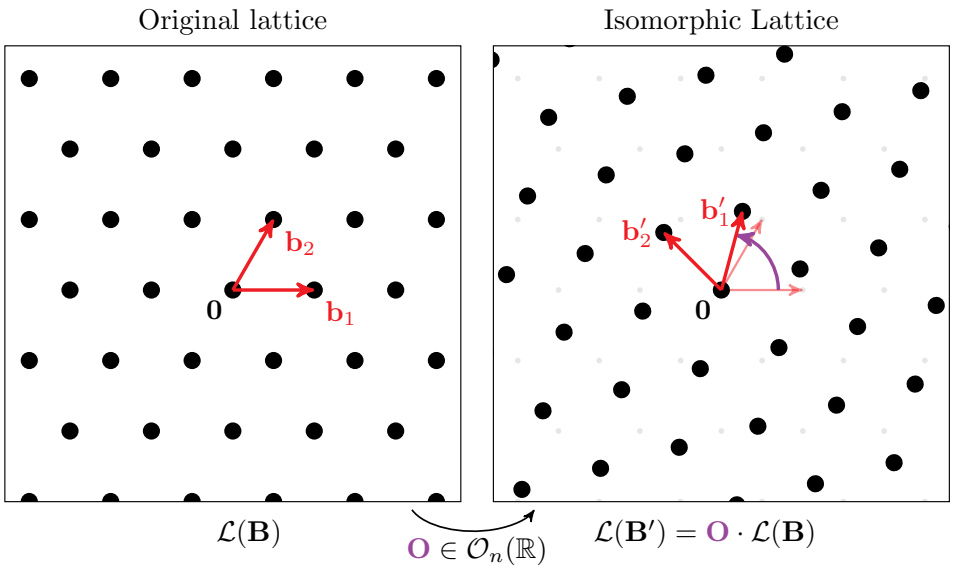


Figure 1.9: The Lattice Isomorphism Problem.

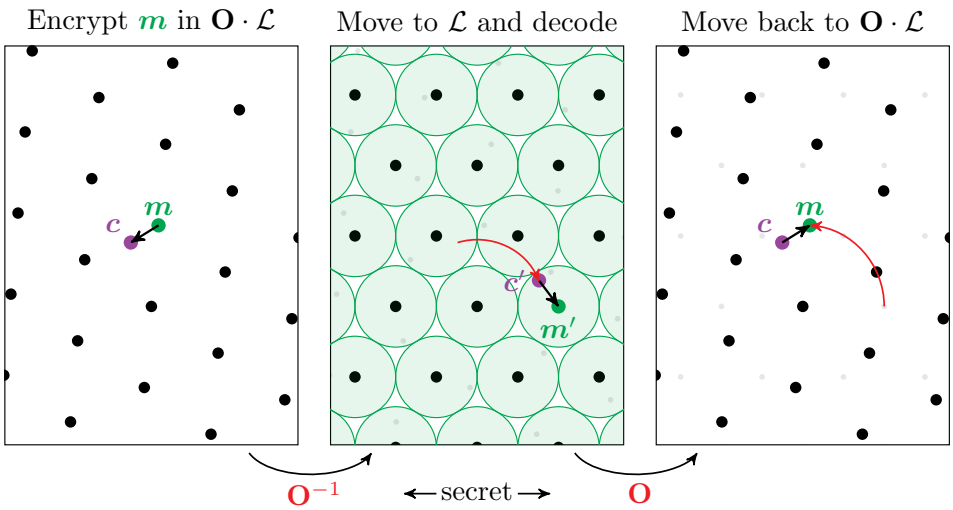


Figure 1.10: Sketch of an encryption scheme based on the lattice isomorphism problem.

structure of $\mathbf{O} \cdot \mathcal{L}$ it is hard to recover the message. However applying the secret transformation we obtain a target $\mathbf{c}' := \mathbf{O}^{-1} \cdot \mathbf{c}$ at distance ρ from the decodable lattice $\mathcal{L}$. We decode here to obtain a lattice vector $\mathbf{m}' \in \mathcal{L}$ and a small error $\|\mathbf{e}'\| \leq \rho$ such that $\mathbf{c}' = \mathbf{m}' + \mathbf{e}'$, which we then map back to the public lattice $\mathbf{O} \cdot \mathcal{L}$. By the uniqueness of the decoding we get that $\mathbf{m} = \mathbf{O} \cdot \mathbf{m}'$, and thus the original message is recovered.

Care has to be taken when formalizing the sketched encryption scheme. Firstly the particular basis chosen to represent the public lattice $\mathbf{O} \cdot \mathcal{L}$ should not leak any information about the original lattice $\mathcal{L}$. This can be solved by defining an appropriate average-case basis distribution, along with a worst-case to average-case reduction. In short, if one can recover $\mathbf{O}$ using the particular average-case description of $\mathbf{O} \cdot \mathcal{L}$, then one can recover $\mathbf{O}$ using any description of $\mathbf{O} \cdot \mathcal{L}$. Secondly the orthonormal transformation $\mathbf{O} \in \mathcal{O}_n(\mathbb{R})$, and the lattice $\mathbf{O} \cdot \mathcal{L}$ is defined over the reals and would in practice have to be approximated, say with floating point numbers, leading to loads of technical difficulties. This problem can be sidestepped by working directly with the Gram matrix $\mathbf{B}^\top \mathbf{B}$ instead of a basis $\mathbf{B}$. Given that $(\mathbf{OB})^\top (\mathbf{OB}) = \mathbf{B}^\top \mathbf{B}$ the orthonormal transformation moves out of the picture.

Outline and Contributions

After this introductory chapter, this thesis proceeds with the preliminaries in Chapter 2. Here the basic theory needed in the later parts is introduced. The remainder of the thesis is split into three parts.

Part II - Short and Close Lattice Vectors

Part II covers the Shortest and Closest Vector Problem. Chapter 3 treats the state-of-the-art on lattice sieving. In Chapter 4 we improve on this by implementing the current best lattice sieving algorithms on both CPUs and Graphical Processing Units (GPUs). With these implementations we improve the TU Darmstadt SVP records by 25 dimensions. This corresponds to a gain of about two orders of magnitude over previous records both in terms of wall-clock time and

energy efficiency. These contributions were published at Eurocrypt 2021 [DSW21].

In Chapter 5 we properly model and analyse the behaviour of the randomized iterative slicer algorithm. This is a CVP algorithm that takes as preprocessed input a large list of short lattice vectors. We present the first tight results on the time-memory trade-off for this algorithm. In the full work, published at PKC 2020 [DLW20b], we also show how to significantly reduce the memory usage of this algorithm when using nearest neighbour techniques.

Part III - Basis Reduction

Part III is all about basis reduction. Chapter 6 gives an overview of the state-of-the-art basis reduction algorithms, their heuristic behaviour, and the latest improvements.

Chapter 7 contributes to the state-of-the-art by giving a heuristically tight and concrete estimate for which parameters the BKZ algorithm solves the NTRU problem, backed by many experiments. This work, published at Asiacrypt 2021 [DW21], gives the first concrete and the best asymptotic estimates for the security of NTRU instances with large moduli, and, contrary to prior works, explains *how* the particular structure of the NTRU lattice allows to recover the secret key in this regime.

Chapter 8 introduces the notion of basis reduction to the world of binary codes (with the Hamming metric). The notion of orthogonality in the Euclidean space is ported to a corresponding notion of *orthopodality* for the Hamming metric. An adaptation of Gram-Schmidt Orthogonalisation and the LLL basis reduction algorithm from lattices to binary codes follows directly. In particular the proofs follow analogues to the lattice case. These contributions were published in IEEE Transactions of Information Theory in 2022 [DDW22].

Part IV - The Lattice Isomorphism Problem, Remarkable Lattices and Cryptography

Part IV covers the Lattice Isomorphism Problem (LIP). Chapter 9 starts with an introduction of LIP, explains how the problem is more naturally stated using quadratic forms, and discusses the best asymp-

totic and practical algorithms for solving it. In Section 9.6 a canonical function for lattice isomorphism is introduced together with an algorithm to compute it. This contribution was published at ANTS XIV in 2020 [DHVW20]. Subsequently, in Section 9.7 further practical improvements to this algorithm are discussed, which found application in ongoing work on perfect form enumeration, with the eventual goal of proving what is the densest lattice packing in dimension 9.

The thesis finishes with its main novel contribution in Chapter 10, where LIP is introduced as a new foundation for lattice-based cryptography. This is motivated by the cryptanalysis in previous parts. In particular, and contrary to currently used methods, this new foundation allows the use of remarkable lattices with great geometric and algorithmic properties. The chapter starts by establishing an average-case version of LIP, along with a worst-case to average-case reduction. Then, on this solid base, an identification, key encapsulation and signature scheme is constructed. We continue by discussing cryptanalysis and instantiations. These contributions were published at Eurocrypt 2022 [DW22].

Section 10.8 contains a short summary of the signature scheme Hawk, published at Asiacrypt 2022 [DPPW22]. This scheme, based on the LIP foundation applied to the simple lattice $\mathbb{Z}^n$, improves significantly on Falcon. The lattice $\mathbb{Z}^n$ is geometrically similar to the NTRU lattice used by Falcon. However, the simplicity of $\mathbb{Z}^n$ allows to remove the main disadvantage of Falcon: the use of floating point numbers in signing. Hawk-512 is $4\times$ to $15\times$ faster in signing than Falcon-512 for the AVX2 and reference implementation respectively, while maintaining similar security levels and signature and key sizes.