Lattice cryptography: isomorphisms & dual attacks

Pulles, L.N.

Lattice Cryptography:

Isomorphisms & Dual Attacks

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr. S. de Rijcke,
volgens besluit van het college voor promoties
te verdedigen op woensdag 23 september 2026
klokke 16:00 uur

door

Ludo Niels Pulles
geboren te Wageningen
in 1997

Promotores:

Prof.dr. L. Ducas	(CWI & Universiteit Leiden)
Prof.dr. R.J.F. Cramer	(CWI & Universiteit Leiden)

Promotiecomissie:

Prof.dr.ir. G.L.A. Derks	
Prof.dr. S. Fehr	(CWI & Universiteit Leiden)
Prof.dr. C. Ling	(Imperial College London)
Prof.dr. P.Q. Nguyen	(Inria Paris & École normale supérieure – PSL)
Dr. T. Debris-Alazard	(Inria Saclay & École polytechnique)

The author of this thesis carried out his research at Centrum Wiskunde & Informatica (CWI) in the Cryptology group. The PhD position was funded by ERC Starting Grant project ARTICULATE (no. 947821).

CWI
Centrum Wiskunde & Informatica

Universiteit Leiden

Lattice Cryptography:

Isomorphisms & Dual Attacks

Ludo Niels Pulles

ISBN 978-90-6196-440-7
Book cover by Puck Swildens
Printed & bound by Ridderprint

Contents

List of Algorithms

List of Tables

List of Figures

List of Publications

This is a comprehensive list of all the publications and preprints that the author has contributed to during their PhD programme. Each publication here lists authors alphabetically ordered. Each author has contributed equally to each publication.

Four chapters from this thesis are based on the following published papers, ordered chronologically.

[DPPvW22a] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 65–94. Springer, Cham, December 2022. DOI: 10.1007/978-3-031-22972-5_3.

[DP23c] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 37–69. Springer, Cham, August 2023. DOI: 10.1007/978-3-031-38548-3_2.

[vGP25] Daniël M.H. van Gent and Ludo N. Pulles. HAWK: Having automorphisms weakens key. *IACR Communications in Cryptology*, 2(2), 2025. DOI: 10.62056/a3qjp2w9p.

[DP26] Léo Ducas and Ludo N. Pulles. Accurate score prediction for Dual-Sieve attacks. *Journal of Cryptology*, 39(8), 2026. DOI: 10.1007/s00145-025-09560-7.

This thesis is also based on the following specification document submitted to NIST for the standardisation of additional signature schemes.

[BBD+26] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. Technical report, National Institute of Standards and Technology, 2026. URL: https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures.

In addition, the author has published the following papers that are not part of this thesis.

[PT24] Ludo N. Pulles and Mehdi Tibouchi. Cryptanalysis of EagleSign. In Clemente Galdi and Duong Hieu Phan, editors, *SCN 24, Part II*, volume 14974 of *LNCS*, pages 165–186. Springer, Cham, September 2024. DOI: 10.1007/978-3-031-71073-5_8.

[DPS25] Léo Ducas, Ludo N. Pulles, and Marc Stevens. Towards a modern LLL implementation. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part III*, volume 16247 of *LNCS*, pages 65–99. Springer, Singapore, December 2025. DOI: 10.1007/978-981-95-5099-9_3.

[KKN+26] Alexander Karenin, Elena Kirshanova, Julian Nowakowski, Eamonn W. Postlethwaite, Ludo N. Pulles, Fernando Virdia, and Paul Vié. Cool + cruel = dual, and new benchmarks for sparse LWE. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part IV*, volume 16544 of *LNCS*, pages 304–333. Springer, Cham, May 2026. DOI: 10.1007/978-3-032-25327-9_11.

Finally, the author authored the following preprint that is not part of this thesis.

[PV25] Ludo N. Pulles and Paul Vié. Accelerating the primal hybrid attack against sparse LWE using GPUs. Cryptology ePrint Archive, Paper 2025/1990, 2025. URL: https://eprint.iacr.org/2025/1990.

Part I
Introduction & Preliminaries

Chapter 1

Introduction

> *In a few decades [...] transfer of information will probably be much faster and much cheaper by "electronic mail" than by conventional postal systems. [...]*
>
> *It is hardly surprising that in recent years a number of mathematicians have asked themselves: is it possible to devise a cipher that can be rapidly encoded and decoded by computer, can be used repeatedly without changing the key and is unbreakable by sophisticated cryptanalysis? The surprising answer is yes.*
>
> — Martin Gardner [Gar77]

Today's society is unimaginable without cryptography, and our personal digital presence is inextricably linked by cryptographic protocols: messaging, online banking or filing taxes. The world has changed significantly within a century since the development of the first computer, which aided the breaking of the Enigma machine's encryption during the second world war.

For a very long time, if two people wanted to communicate while keeping its content unknown to others, one could *encrypt* messages using a *secret key*, send this *ciphertext* to the other, and the other could *decrypt* these ciphertexts using that same secret key. Worldwide usage of this encryption method was virtually impossible as the common secret key had to be confidentially communicated beforehand, for example by meeting in person. Encryption was impractical!

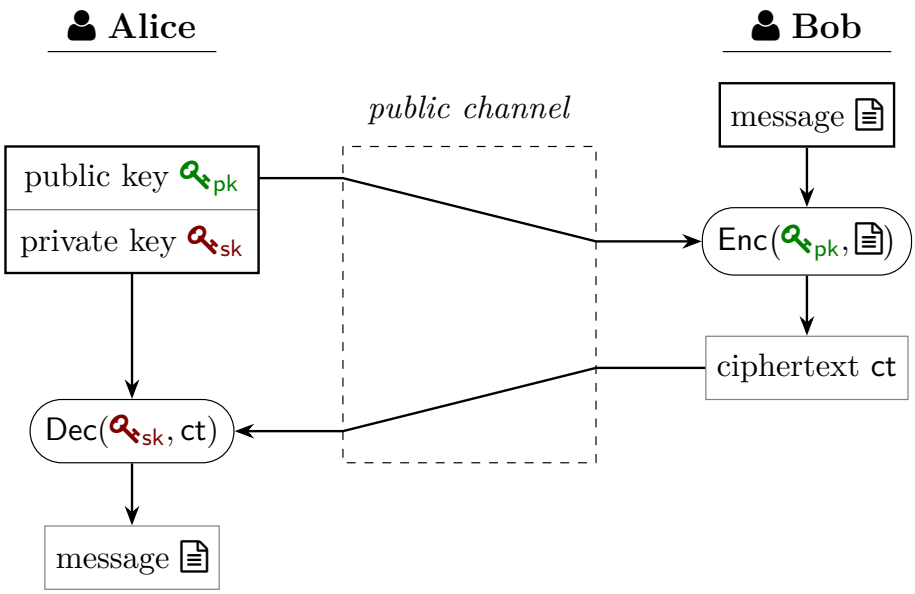


Figure 1.1: Public-key encryption, in which Bob tries to confidentially communicate a message to Alice. Over a public channel, Alice sends her public key to Bob, and Bob sends a ciphertext back.

Rivest–Shamir–Adleman

Fifty years ago, it suddenly became much easier to use cryptography with the discovery of *public-key cryptography*. In an article, Gardner [Gar77] challenges the reader to break the *public-key encryption scheme* RSA, named after its inventors Rivest, Shamir and Adleman [RSA78], who also offered $100 to anyone solving the challenge.

Public-key encryption, visualised in Figure 1.1, is a way of sending a message confidentially that does not require both parties to know a shared secret. The key difference with symmetric cryptography is that encryption now uses a *public key* known to everyone, while decryption requires knowledge of the associated *private key* only known by one individual.

Suppose Alice wants to receive messages confidentially from anyone using RSA encryption. First, Alice randomly generates two large prime numbers p and q, and picks a relatively small integer e. Then, she computes the *RSA modulus* $N = p \cdot q$, and constructs an integer d such that $d \cdot e \equiv 1 \pmod{(p-1)(q-1)}$ holds.

While making the public key $\mathsf{pk} = (N, e)$ known to everyone, she makes sure that the private key $\mathsf{sk} = (p, q, d)$ is kept secret. Now, the encryption of a message $m \in \{0, 1, \ldots, N-1\}$ is simply $c \equiv m^e \pmod{N}$ which can be publicly computed, and passed publicly to Alice. Only Alice, who knows d, can decrypt the ciphertext using $m \equiv c^d \pmod{N}$, which follows from elementary number theory. In order to break encryption, it is sufficient to find a prime divisor of N. Although it is generally not necessary to recover the private key in order to break encryption, factoring N still remains the fastest known method for breaking RSA encryption.

If the number N is sampled uniformly from $\{1, 2, \ldots, n\}$, finding a prime divisor of N is easy in at least 50% probability, that is when N is even, but is considered hard in the worst case [Knu97, 4.5.4]. The number N is generated in RSA specifically as a product of two primes, both of similar size, such that private key recovery requires solving a factorisation instance that is *hard in the average case*. A locksmith should deliver secure locks every time, and similarly, a good cryptosystem should generate a public/private key pair such that the private key is never easily derived from the public key. Hence, RSA generates N as a product of two primes.

In Gardner's article, the challenge was, given a ciphertext and public key (N, e), to recover a message, which was encoded as an integer modulo N by mapping the space character to 00 and letters A–Z to 01–26, respectively. The RSA modulus N was said to be a 129-digit, obtained by multiplying a 64-digit prime p and a 65-digit prime q. Rivest estimated for the article that, 'if the best algorithm known and the fastest of today's computers were used,' solving this challenge would take $40 \cdot 10^{12}$ years. According to [AGLL94], the fastest algorithm at that time was the heuristic Pollard's ρ algorithm [Pol75] which outputs a divisor of N using $2\sqrt{p}$ modular operations in practice, although $\mathcal{O}(p/\log\log p)$ is the best known provable upper bound [Hea17].

> *Rivest and his associates have no proof that at some future time no one will discover a fast algorithm for factoring composites as large as the N [red.] they used or will break their cipher by some other scheme they have not thought of. They consider both possibilities extremely remote.*
>
> — Martin Gardner [Gar77]

1

For decades many researchers tried to find a fast algorithm for the factorisation problem, but nobody found one that could *efficiently* factorise N, i.e., there is no factorisation algorithm known with a runtime that grows polynomial in its input size, $\lg N$. Still, better *superpolynomial* algorithms were discovered, such as the quadratic sieve [Pom85], and the number field sieve (NFS) [LL93].

Gardner's challenge was ultimately solved in 1994 by a fast implementation of the quadratic sieve [AGLL94]. In their article, the authors recommend stopping use of RSA with a 512-bit modulus N (155 digits), and move to a 1024-bit modulus. Note this challenge can be solved nowadays within an hour distributed over 8 CPUs [McH15].

In 2009, a 768-bit (232 digits) RSA modulus was factored with the NFS, 2 years after the RSA Laboratories withdrew the $50,000 prize. Subsequently, the recommendation was to use a 2048-bit modulus.

Putting things into perspective, RSA was a big success, and has already seen wide usage for a very long time. Despite progress into integer factorisation, the doublings of the necessary key-sizes have been manageable, so far...

The quantum threat

In 1994, Shor publishes a *quantum algorithm* [Sho94] that could factorise a number N using $\mathsf{poly}(\lg N)$ operations on *qubits*, the quantum analogue of the normal bits inside a classical computer. Although no quantum computer existed at the time, Shor's algorithm threatened the security of RSA. In the same paper, Shor proposes another quantum algorithm that solves the discrete logarithm problem, which threatens other cryptosystems that use elliptic curves.

So far, there does not exist a cryptographically relevant quantum computer (CRQC), i.e., a quantum computer with sufficiently many qubits that are sufficiently fault-tolerant to factorise 2048-bit RSA. Physicists are working hard at lowering quantum gate error rates [SLM+25], and increasing the number of qubits. On the other hand, Shor's algorithm has seen recent improvements [GE21, Reg25, CFS25, Gid25], lowering the required number of qubits further.

With these advances, the time until the first CRQC appears becomes less and less, and with it the need to migrate to quantum-resistant cryptography increases. The current state of quantum computers is summarised in Figure 1.2.

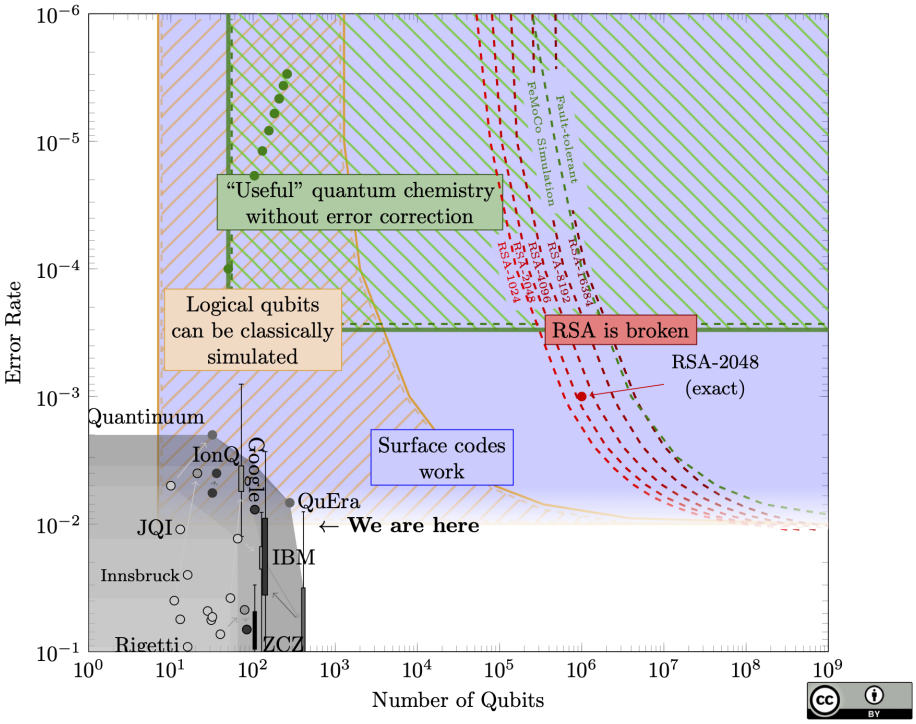


Figure 1.2: Landscape of quantum computing capabilities in terms of fault tolerance and number of qubits in a log scale. Physically realised quantum computers (grey) versus the requirements of Gidney's algorithm [Gid25] to break RSA (red). Created by Sam Jaques [Jaq25].

Note that some products are shipped with immutable cryptography, like vehicles or smart cards. Moreover, an eavesdropper can simply store communication today, and decrypt it when a CRQC is available, which may uncover still relevant sensitive industrial/medical data, or even state secrets. This is the so-called '*store-now/decrypt-later*' threat. Hence, all sensitive data must be migrated to post-quantum cryptography long before a CRQC exists.

Post-quantum cryptography migration

The imminent threat of quantum computers capable of breaking encryption creates the desire to transition towards new cryptographic systems that are secure against *both* quantum and classical computers. Such so-

called *post-quantum cryptography* must run on (classical) computers, and needs to rely on hardness assumptions that are different from RSA and discrete log. This led the National Institute of Standards and Technology (NIST) in the US to invite researchers from all over the world to submit post-quantum cryptography candidates with the purpose of establishing a standard that could be used for free by governments, industries and people worldwide to migrate before a CRQC exists [Nat16].

In 2022, NIST announced ML-KEM will be the new standard for a *key encapsulation mechanism* (KEM) [AAC+22]. A KEM enables two parties to establish a shared secret key over a public communication channel, so they can use (more efficient) symmetric encryption to communicate securely. Hence, a KEM is used to construct a public-key encryption scheme that is efficient regardless of the message size. As of 2026, Cloudflare observes [Wes25] that *at least half* of human web traffic uses the post-quantum secure KEM 'X25519MLKEM768', which is a hybrid of X25519, an elliptic curve-based KEM that is susceptible to Shor's algorithm, and ML-KEM. An attacker needs to break both X25519 *and* ML-KEM in order to obtain the 'X25519MLKEM768' private key. This hybridisation provides security that is best of both worlds at the cost of running and communicating both KEMs.

On the governmental side, US governmental systems need to be quantum-safe by 2035, while the EU high-risk systems need to be quantum-safe by 2030 and the rest by 2035. Australia sets more aggressive goals by demanding migration by 2030 to ML-KEM-1024, which offers more security than ML-KEM-768.

While the urgent migration of public-key encryption is already on the right trajectory, there is an important second migration needed:

> *The second migration, to post-quantum signatures (certificates), is not as urgent: we will need to have this sorted by the time the quantum computer arrives. However, it will be a bigger challenge.*
>
> — Westerbaan and Valenta [WV24]

Migration of digital signatures

Often in internet communication, one needs to know they are talking to the right person, or one needs to assert that some document really originates from someone. This can be achieved using a *digital signature*

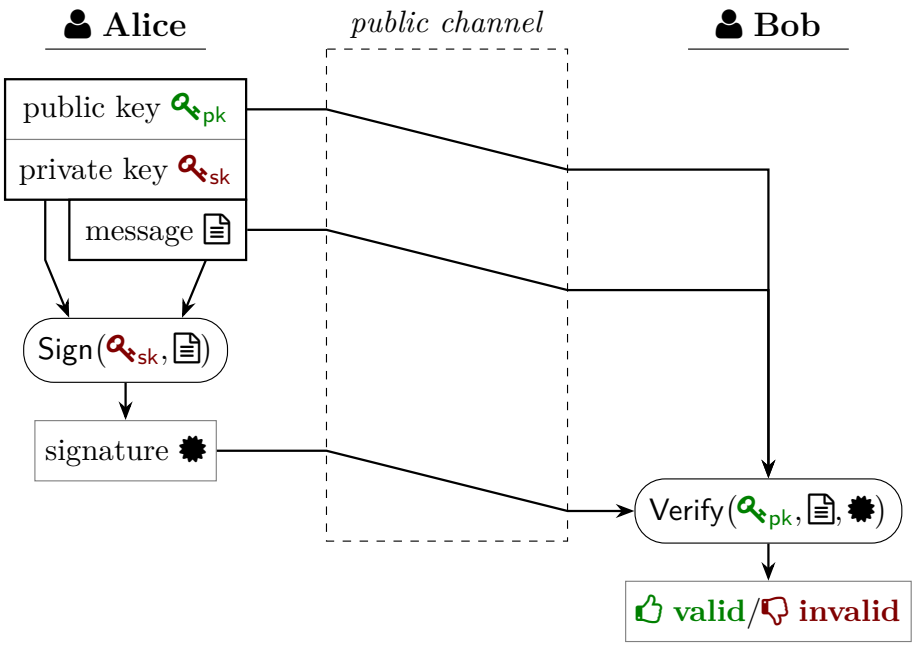


Figure 1.3: Digital signature scheme, in which Bob wants to know whether Alice sent him a particular message.

scheme or *signature scheme*, depicted in Figure 1.3.

Say Alice wants to send a message m to other people. Before the age of computers, Alice would add a wax seal to such a message, convincing people she used the seal die on that message. A digital signature scheme may be considered the digital analogue to the wax seal that offers more security as the seal is now specific to the message. Namely, Alice generates a key pair consisting of a public key pk and private key sk, and then publishes pk. A signature σ is the result of an algorithmic routine that takes as input her private key sk and the message m. Then, anyone acquiring the message and signature, can run a procedure called Verify that takes as input (pk, m, σ) and decides whether the signature is *valid*. Formally, we say a signature scheme is *existentially unforgeable under chosen message attacks* (EUF-CMA secure) if an attacker, without the private key sk, cannot produce within a certain time a valid signature on a message $\mathsf{m}^\star$ of their liking, even if they can persuade Alice to sign a certain number of arbitrary messages $\mathsf{m}_i \neq \mathsf{m}^\star$. Here the words 'certain' depend on the desired level of security.

1

Rivest–Shamir–Adleman [RSA78] proposed a signature scheme that is surprisingly similar to RSA encryption (p. 4). Namely, using the same public/private keys, a message m is signed using $\sigma \equiv m^d \pmod N$, and a signature σ is valid when $m \equiv \sigma^e \pmod N$ holds. This is essentially the idea of RSA signatures, although it is not EUF-CMA secure: if an attacker learns a valid signature σ for message m, then they can forge a signature 2σ for the artificial message $m' \equiv 2^e m \pmod N$. By slightly modifying the scheme, e.g., hashing the message [Cor00], one can make it EUF-CMA secure. Moreover, it is not secure to reuse the key pair from RSA encryption for RSA signatures.

In contrast to encryption, which needs to be quantum-safe as soon as possible due to store-now/decrypt-later, digital signatures cannot be forged until a CRQC exists. Still, these signatures need to be implemented in critical parts of the internet, especially because it takes multiple years before X.509 certificates, used in secure web-browsing, are replaced by new (post-quantum) signatures.

In 2022, NIST chose [AAC+22] to standardise three signature schemes: ML-DSA, FN-DSA and SLH-DSA. These three have different characteristics in terms of hardness assumption, runtimes, communication sizes, and implementation complexity. It therefore depends on the application which of these three is the most favourable. An SLH-DSA signature requires more than 7 kB, which is too large for some internet applications. The first two have smaller signature sizes of 2420 B and 666 B, respectively.

Based on signature sizes, one would guess that NIST would recommend using FN-DSA instead of ML-DSA, but the opposite is true: ML-DSA is recommended in general! FN-DSA signatures need floating-point numbers to generate signatures, but some CPUs do not support these. Namely, CPUs of some small chips, such as smart cards, do not support double-precision floating-point numbers. Although one can carefully emulate floating-points with 64-bit integers, this severely slows down signing. The implementation difficulties of FN-DSA are also evident from the fact that there is no FN-DSA standard yet, as opposed to ML-KEM [Nat24a], ML-DSA [Nat24b] and SLH-DSA [Nat24c].

When the third round of NIST's PQC standardisation process finished, there were no more signature schemes under consideration [AAC+22]. Among the standardised signature schemes, only SLH-DSA is not based on structured lattices, but this one has relatively large signature sizes.

As such, NIST opened a call for additional digital signature proposals to be considered in the PQC standardisation process [Nat22]. While NIST is primarily interested in diversifying the post-quantum signature standards by schemes that do not rely on structured lattices, NIST is also interested in lattice-based proposals that significantly outperform ML-DSA and FN-DSA. This standardisation process, also known as the PQC signature 'on ramp', considers nine candidates in its ongoing third round [ABC+26].

1.1 Motivation

Lattices are the central object of study in this thesis, as three of the four schemes standardised by NIST are based on lattices: ML-KEM, ML-DSA and FN-DSA.

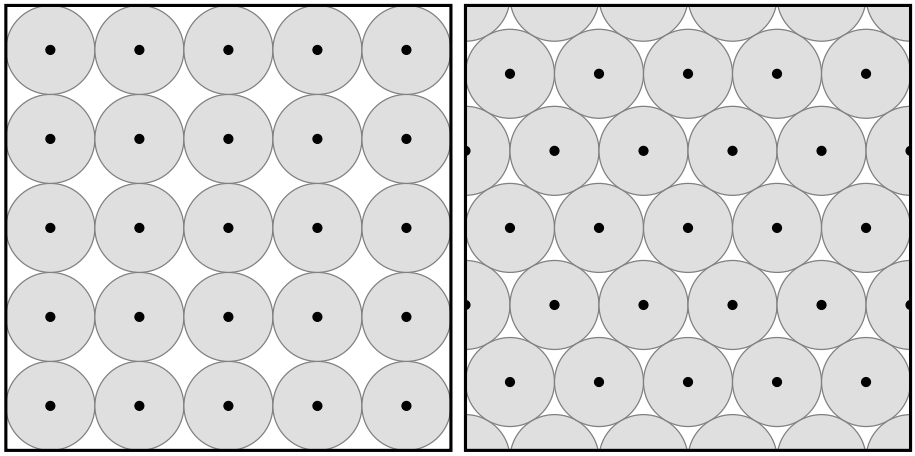
We will perform *cryptanalysis* on lattices. Cryptanalysis is the process of designing, analysing and optimising algorithms that can break a scheme, and determining the most efficient one. The schemes are supposed to offer security, so it will take too long to run the best attack in an experiment. Instead, we can only estimate the runtime of the best attack by understanding it theoretically, and by extrapolating runtimes required to break weaker instances. For accurate runtime estimates of lattice-based cryptosystems, it is essential to have a thorough understanding of lattices.

Lattices

A lattice is a discrete, additive subgroup of n-dimensional Euclidean space, $\mathbb{R}^n$. Figure 1.4 contains two examples of a lattice. Here, *discrete* means that there is a minimal distance between any two lattice points, i.e., we can draw a ball of some positive radius around every lattice point such that none of these balls intersect. Moreover, *additive subgroup* means that the collection of points is closed under addition and negation, i.e., if $\mathbf{x}$ and $\mathbf{y}$ are both lattice points, then so are $-\mathbf{x}$ and $\mathbf{x}+\mathbf{y}$. Note that the zero vector, $\mathbf{0}$, must also lie in a lattice.

For any computational task with a lattice, we use a basis, which consists of n lattice points such that any lattice point can be expressed as an integral combination of the basis vectors.

Example 1.1. Let $n \in \mathbb{Z}_{\geq 1}$. The elementary unit vectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$, where



(a) $\mathbb{Z}^2$ (b) Hexagonal lattice

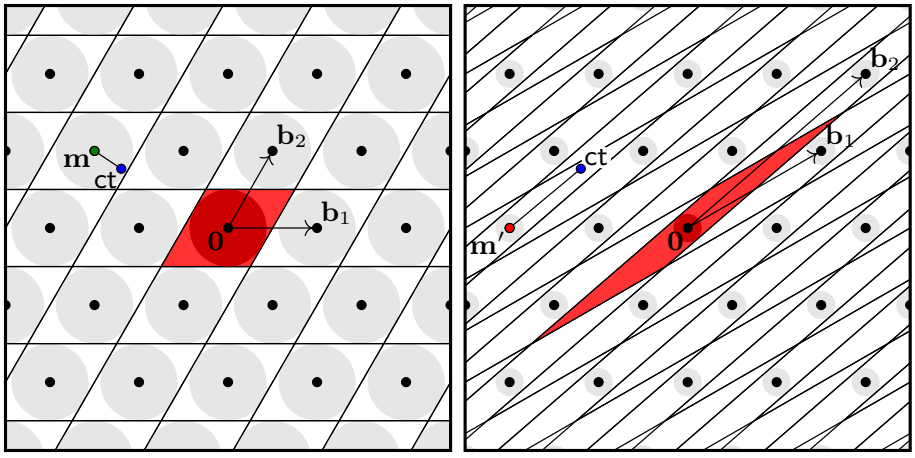
Figure 1.4: Two examples of a lattice, together with nonoverlapping balls centred at each lattice point.

$\mathbf{e}_i \in \mathbb{Z}^n$ has 1 as its ith coefficient and zeros elsewhere, form a basis for the integer lattice $\mathbb{Z}^n$.

While a 1-dimensional lattice has only two bases, any higher dimensional lattice has an infinite number of choices for a basis. For the remainder of this chapter, let us assume the dimension is at least 2. Informally, we say a basis is 'good' if it consists only of short vectors, and say a basis is 'bad' if it does not. For any lattice, there exist arbitrarily 'bad' bases.

Every basis gives a tiling of the space as can be seen in Figure 1.5. The radius of the largest ball that fits in a tile is larger for a good basis than for a bad basis. We may use lattices for public-key cryptography using a good basis as private key, and a bad basis as public key. Namely, we can associate the message space with lattice points, expressed in terms of the public, bad basis. Then, given a lattice point $\mathbf{m}$, we can generate noise $\mathbf{e}$ of a length r such that $r_a > r > r_b$, where r_a, r_b are the radii of the grey circles in (a) and (b) of Figure 1.5, respectively, after which the ciphertext is $\mathsf{ct} = \mathbf{m} + \mathbf{e}$. This way, anyone cannot decode the ciphertext to the correct message using the tiling, unless they have the good basis.

In Figure 1.6, it is shown how a lattice-based signature scheme can be built. While a KEM is usually used once by two parties to establish a



(a) Good basis (b) Bad basis

Figure 1.5: Tiling of space based on a good, and a bad basis for the hexagonal lattice, respectively. The encryption $\mathsf{ct} = \mathbf{m} + \mathbf{e}$ of message $\mathbf{m}$, obtained by adding a short error $\mathbf{e}$, can only be successfully decrypted using the tiling of a good basis.

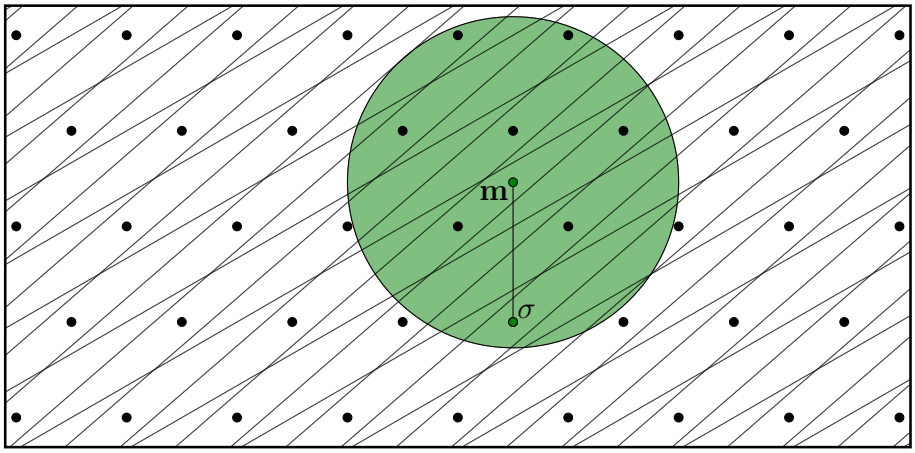


Figure 1.6: High-level idea of a lattice-based signature scheme. A message is mapped into space (a non-lattice point), and a signature is a lattice point in the vicinity of the message.

shared key for (symmetric) encryption, a signature scheme could be used
1 millions of times with the same key. As a consequence, signatures should not leak too much information about the private key. The early lattice-based signature schemes GGH and NTRU, for example, were insecure because the private key could be recovered if many valid signatures were known [NR09]. This makes it more of a delicate matter to build a signature scheme [GPV08].

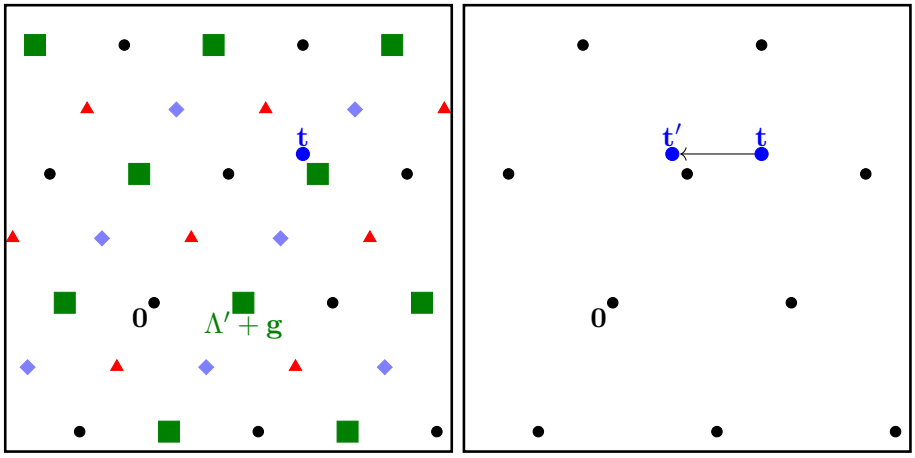
Lattice basis reduction

The process of improving a basis with shorter basis vectors is called *lattice basis reduction*. The *shortest vector problem* (SVP) asks, given an arbitrary basis, to find a shortest lattice point. It was shown [Ajt98] that SVP is NP-hard for randomised reductions, i.e., there is a probabilistic Turing machine that, in polynomial time, reduces any NP-problem to an SVP instance. Note that SVP is trivial when the input basis contains the shortest vector. Yet, even for the average case there is no polynomial-time algorithm known solving SVP, where we omit the average case input distribution as it is technically too involved for a gentle introduction. In contrast to factorisation of a random integer (p. 5), which was trivial with 50% probability, SVP is hard on average with high probability.

On a more algorithmic side, there exists a polynomial-time algorithm [LLL82] that finds a lattice point of length at most 2^n times larger than the smallest norm for a lattice of dimension n. This so-called LLL algorithm improves a basis and prevents its basis vectors from becoming arbitrarily long. In small dimensions, LLL performs well, and the approximation factor is rather small, making LLL useful in many applications [NV10].

Ultimately, secure lattice-based cryptosystems can only be broken by stronger basis reduction, in particular, using the so-called *block Korkine–Zolotarev* (BKZ) reduction [Sch87]. This algorithm requires exponential time in a parameter β, called *block size*, which controls the dimension of projected sublattices. BKZ improves the basis by repeatedly solving SVP locally in these projected sublattices.

To determine security of a KEM, we would like to know, given a ciphertext, how easily one finds the closest lattice point, which is called the *bounded distance decoding problem* (BDD). Similarly for a signature scheme, we would like to know, given a (specific) message, how easily one finds a nearby lattice point, which is called the *(approximate) closest*



(a) Finding nearest coset (b) After sparsification

Figure 1.7: Solving BDD by finding the nearest coset $\Lambda' + \mathbf{g}$ to the target $\mathbf{t}$. In (a), out of the four given cosets of Λ', the green coset is nearest to $\mathbf{t}$. After applying a shift of $-\mathbf{g}$ to the coset, we get (b), which is another BDD instance with lattice Λ' and target $\mathbf{t}' = \mathbf{t} - \mathbf{g}$.

vector problem (CVP). There are two ways of solving BDD: the *primal attack* and the *dual attack*.

Primal attack

The primal attack constructs a lattice such that the BDD solution can be easily computed from the shortest vector of this lattice. Thus, by running BKZ, one can solve BDD. In these lattice constructions, the shortest vector is smaller than that of a random lattice of same dimension. The ratio between these two quantities can then be used to determine the required block size [ADPS16].

Dual attack

The dual attack, visualised in Figure 1.7, partitions the lattice Λ into multiple *cosets* of a sparser sublattice Λ', and then determines which coset the input is closest to. By repeatedly doing this, the lattice becomes sparser during each step, and at a certain point, the lattice is so sparse that the nearest lattice point is easy to recover, e.g., using LLL.

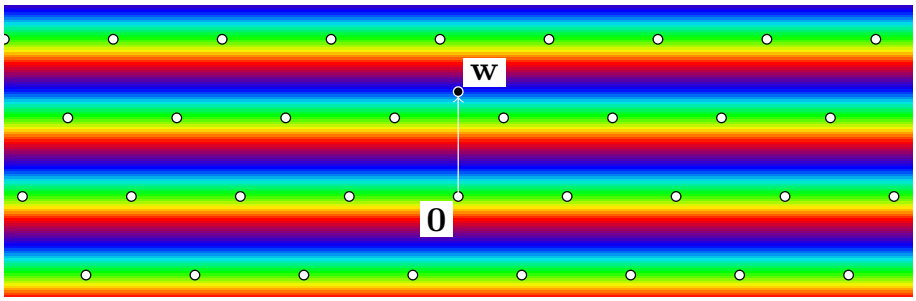


Figure 1.8: Visualisation of a dual vector, using the correspondence between dual vectors and characters. The green colour indicates the top of the wave function, which is at every lattice point.

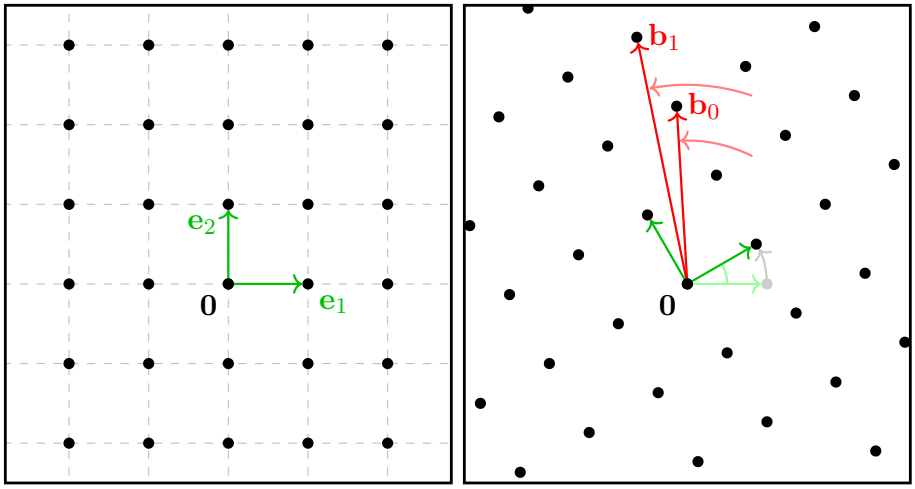
To determine the closest coset to a target, we use many *dual vectors*, i.e., vectors from the so-called *dual lattice*. A dual vector corresponds to something called a *character*, which can be thought of as a travelling wave in n-dimensions, such that the wave is at its top in every lattice point, visualised in Figure 1.8. More formally, a character is a function (with certain properties) from n-dimensional Euclidean space to the unit circle that maps every lattice point to $(1, 0)$. A character needs to be continuous, and therefore a point that lies near the lattice must evaluate to something close to $(1, 0)$. The converse is, however, not necessarily true. By combining many characters, one gains more confidence whether a point is really near the lattice. Namely, we can assign a *score* to a target $\mathbf{t}$, by considering the sum of the evaluations of characters at $\mathbf{t}$. Since the y-coordinate cancels out with that of a dual vector's negation, we may consider the score as a *real* value. Now, given a list of targets, in which only one is close to the lattice, very naturally, the target with the highest score is most likely the closest one to the lattice.

Lattice Isomorphism Problem

The integer lattice, $\mathbb{Z}^n$, cannot be used directly as a cryptosystem because it is publicly known that the decoding function,

$$\lceil \cdot \rfloor \colon \mathbb{R}^n \to \mathbb{Z}^n, \quad (x_1, \ldots, x_n) \mapsto (\lceil x_1 \rfloor, \ldots, \lceil x_n \rfloor),$$

efficiently solves CVP and BDD by rounding each coefficient to its nearest integer. What if we would be able to use such a simple lattice in a cryptosystem?



(a) Good basis for Λ (b) Rotated bad basis for $\mathbf{O}\Lambda$

Figure 1.9: Lattice Isomorphism Problem

Such a cryptosystem would be fast and simple because of this efficient, easily implementable decoding function. Moreover, generating signatures is easy because each coefficient can be sampled independently, and distributions can be precomputed. By building a signature scheme with $\mathbb{Z}^n$, one can generate signature without using floating-points, contrary to the intricate signing in Falcon which requires floating-points because it uses NTRU lattices [HPS98]. How could we use this simple lattice?

The answer lies in the Lattice Isomorphism Problem. Lattices Λ_1 and Λ_2 are said to be *isomorphic* if there is an orthogonal transformation, i.e., a length-preserving map that maps Λ_1 to Λ_2 one-to-one, in which case we call the transformation the *isomorphism*. Such a map can be a combination of a rotation and a reflection through the origin. The *Lattice Isomorphism Problem* (LIP) asks, given two isomorphic lattices, to find such an isomorphism. An illustration is found in Figure 1.9. All best known ways to solve LIP require finding short vectors in each lattice using lattice basis reduction [PS97, HR14, DHVvW20]. Hence, we believe LIP is, on average, as least as hard as solving SVP.

Example 1.2. In the remarkable case of $\mathbb{Z}^n$, of which the shortest vectors form an orthonormal basis, solving $\mathbb{Z}$LIP, LIP for the lattice $\mathbb{Z}^n$, is computationally as hard as finding all its shortest vectors. It is sufficient to find all shortest vectors of a rotation of $\mathbb{Z}^n$ to recover the orthogonal

transformation in LIP, and this is also necessary, as all shortest vectors
1 are encoded in the transformation matrix.

Remark 1.3. To create an instance of LIP, the basis $\mathbf{B}_2$ of Λ_2 must be generated by *both* applying the orthogonal transformation to $\mathbf{B}_1$ *and* applying a random basis transformation to it. Without the basis transformation, one can easily compute the orthogonal transformation as $\mathbf{B}_2\mathbf{B}_1^{-1}$.

In the current formulation, a LIP-based signature scheme certainly must use floating-points to rotate a basis, which would make a robust implementation more tedious than what FN-DSA offers. However, one can avoid all these difficulties by working with bases modulo orthogonal transformations, and things become simpler. Given a basis $\mathbf{B}$, the *Gram matrix* $\mathbf{G} = \mathbf{B}^\mathsf{T}\mathbf{B}$ encodes all the pairwise inner products between basis vectors. This Gram matrix is invariant under orthogonal transformations as these transformations are length-preserving. Moreover, if $\mathbf{B}$ has integer entries, $\mathbf{G}$ is also integral. Hence, we prefer to describe an instance of LIP, not using bases of two lattices but using the Gram matrices of these bases. A Gram matrix is an instance of what is formally called a (positive-definite) *quadratic form*.

Remark 1.4. Quadratic forms were already studied, e.g., in Disquisitiones arithmeticæ (1801) by Gauß, long before lattices came into fashion.

Given a quadratic form $\mathbf{Q}$, one can find a particular lattice basis with Gram matrix $\mathbf{Q}$. However, most lattice algorithms have a quadratic form equivalent, so an actual basis is often not needed when manipulating a quadratic form.

Additional structure

Lattice-based cryptosystems offer security when using a dimension of at least a thousand. A basis for such a lattice involves a million entries, so the communication cost is on the order of megabytes. To reduce the communication and computation costs, one can enrich the lattice with extra algebraic structure.

Ring of integers of a number field are versatile objects that come naturally with a lattice structure by their canonical embeddings into the complex numbers. The additional structure allows us to reduce communication costs to the order of kilobytes. Moreover, we often use

cyclotomic number fields as the so-called *fast Fourier transform* (FFT) reduces computation costs to a complexity that is almost linear. However, on the other hand, the question is:

Question 1.5. Does the additional structure lead to improved attacks?

Initially, researchers considered enriching the lattice constructions from *Learning with Errors* (LWE) [Reg09] with a ring structure, leading to Ring-LWE [SSTX09, LPR13], in which security is now based on the hardness of *ideal lattice* problems, instead of unstructured lattice problems. A quantum algorithm was discovered for finding relatively short vectors in a principal ideal lattice [CGS14], and was generalised to general ideal lattices [CDW17]. This algorithm did not directly impact Ring-LWE as:

(1) Ring-LWE reduces quantumly to an ideal lattice problem [SSTX09] that in the worst case asymptotically has a *smaller* approximation factor than above algorithm, and
(2) Ring-LWE could be computationally harder since there is no reduction known going the other way.

Nevertheless, if one could significantly lower the approximation factors in the algorithm [CGS14], it could leave security of Ring-LWE unsupported.

> *Despite these two serious obstacles to attack Ring-LWE based schemes by the algebraic approach developed in [CGS14, BS16, CDPR16] and in this paper, it seems a reasonable precaution to start considering weaker structured lattice assumptions, such as Module-LWE [LS15] (i.e., an "unusually-Short Vector Problem" in a module of larger rank over a smaller ring), which provides an intermediate problem between Ring-LWE and general LWE.*
>
> — Cramer *et al.* [CDW17]

Since then, module structures have received a lot of cryptanalytic attention and have shown to be fruitful, as ML-KEM and ML-DSA use module structures. Similarly, FN-DSA also uses a module structure that comes from the NTRU lattice construction [HPS98] instead of LWE. In Part III, we will investigate how to incorporate a module structure into LIP.

1.2 Overview

This thesis is divided into three parts. Part I consists of this introductory chapter and Chapter 2, which covers preliminaries that are necessary for the remaining two parts of the thesis. Part II is cryptanalytic work on the dual attack. Part III constructs and analyses a signature scheme based on a module variant of the Lattice Isomorphism Problem.

Part II — Dual-Sieve Attack

Part II is related to the *Dual-Sieve attack*, which is a dual attack that uses lattice sieving to acquire many dual vectors. It is difficult to determine the runtime of the Dual-Sieve attack, because the lattice sieve is a probabilistic algorithm that receives a randomly generated lattice. Therefore, one usually requires a heuristic assumption on the output distribution of a lattice sieve, to analyse the Dual-Sieve attack.

Chapter 3 investigates the so-called independent score heuristic. Using this heuristic, prior works [GJ21, MAT22] claimed the Dual-Sieve attack can break Kyber512 (ML-KEM-512) in an estimated runtime that is lower than NIST's security requirement. In this chapter, we show that this heuristic leads to three contradictions in certain scenarios. One of these contradictions, depicted in Figure 3.3, is concrete and is observed in extensive experiments. This work was initially published in the proceedings of CRYPTO 2023 [DP23c].

Chapter 4 introduces an alternative heuristic, which can be used to simplify the analysis of the Dual-Sieve attack. This new heuristic is verified in practice using multiple experiments. Showing accurate predictions, this work contributes to a better attack cost prediction for state-of-the-art Dual-Sieve attacks. It is left as future work to determine the time of breaking Kyber512 using the Dual-Sieve attack. This work was originally uploaded to the cryptology ePrint archive [DP23b].

The two works, on which Chapter 3 and Chapter 4 are based, were combined into one paper that is published in the Journal of Cryptology [DP26].

Part III — Lattice Isomorphism Problem

Part III is related to the signature scheme Hawk. This scheme was submitted to NIST's standardisation process of additional signature

schemes [BBD+23], and is the only lattice-based candidate that advanced to the third round [BBD+26]. Security of Hawk is based on the hardness of the Lattice Isomorphism Problem for the integer lattice ($\mathbb{Z}$LIP) having a module structure (smLIP).

Chapter 5 introduces the signature scheme Hawk. First, a theoretically simple version of Hawk is explained. Then, Hawk is explained in detail by applying some optimisations to the simplified variant. This scheme is then cryptanalysed in theory, and we experiment with smaller instances of the scheme. Based on this cryptanalysis, parameter sets are given in Table 5.3, which should provide two levels of security. The novel design of Hawk avoids floating-points, and offers small signature and public key sizes, similar to Falcon. Moreover, Table 5.1 shows great performance on high-end and low-end devices. This chapter is based on the initial work published in the proceedings of ASIACRYPT 2022 [DPPvW22a], and is based on the specification document submitted to NIST [BBD+26].

Chapter 6 looks into the so-called one more shortest vector problem (omSVP), which is the hardness problem on which Hawk is based. In this chapter, we want to characterise the symmetries in the definition of omSVP such that there is no trivial algorithm solving omSVP. Specifically, assuming the private key in Hawk cannot be recovered easily, it is shown that the formulation of omSVP covers all symmetries that are easily computed. This work is published in Communications in Cryptology [vGP25].

Chapter 2

Preliminaries

The sets of complex, natural, rational and real numbers, and integers, are denoted by $\mathbb{C}, \mathbb{N}, \mathbb{Q}, \mathbb{R}, \mathbb{Z}$, respectively. The natural numbers, $\mathbb{N}$ include 0. The logarithm in base $2, e$ and 10, are denoted by $\lg, \ln$ and $\log$, respectively. We may write $[n] = \{1, 2, \ldots, n\}$. Boldfaced uppercase and lowercase letters denote matrices and column vectors, respectively. More notations can be found in the back matter of this thesis in the List of Symbols.

2.1 Elementary theory

For later use, we shortly explain some terminology, and things related to geometry and probability theory.

A square matrix $\mathbf{A}$ is called *unitriangular* if it is upper triangular, i.e., $\mathbf{A}_{ij} = 0$ for $1 \leq j < i \leq n$, and its diagonal consists of only ones.

2.1.1 Terminology

In this thesis, we will encounter heuristics. A *heuristic* is a simplification of a mathematical analysis that is not fully precise or rationalised but nevertheless decent enough of an approximation. It will be made clear whenever a heuristic is used.

Moreover, any result that use one or more heuristics will be called a *heuristic claim*. A *heuristic justification* then shows how the heuristic

claim is derived from the used heuristic(s), or motivates why it is generally believed to be true.

For instance, in Chapters 3 and 4, we derive various heuristic claims regarding random lattices from Heuristic 2.41 and/or Heuristic 2.42. We use heuristics because it would be very difficult to unconditionally prove a more precise version of these claims using a formally defined notion of random lattices.

2.1.2 Geometric objects

In this work, we consider Euclidean space $\mathbb{R}^n$ with the standard ℓ^2-norm induced by the dot product,

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\mathsf{T} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i \ .$$

The ℓ^2-norm, or simply norm, of a point $\mathbf{x} \in \mathbb{R}^n$ is $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

More generally, one may consider for any number $p \in \mathbb{Z}_{\geq 1}$ the ℓ^p-norm of a point $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} \mathbf{x}_i^p \right)^{1/p} \ ,$$

and the ℓ^∞-norm of $\mathbf{x}$ is given by $\|\mathbf{x}\|_\infty = \max\{|\mathbf{x}_1|, \dots |\mathbf{x}_n|\}$.

The *Minkowski sum* of a subset $S \subseteq \mathbb{R}^n$ and vector $\mathbf{t} \in \mathbb{R}^n$ is denoted by $S + \mathbf{t} = \{\mathbf{s} + \mathbf{t} \in \mathbb{R}^n \mid \mathbf{s} \in S\}$, and the *Minkowski sum* of subsets $S, T \subseteq \mathbb{R}^n$ is denoted by $S + T = \cup_{\mathbf{t} \in T}(S + \mathbf{t})$. Given any $c \in \mathbb{R}$ and subset $S \subseteq \mathbb{R}^n$, we write $cS = \{c\mathbf{s} \in \mathbb{R}^n \mid \mathbf{s} \in S\}$.

For $n \in \mathbb{Z}_{\geq 2}$, the $(n-1)$-dimensional sphere is denoted by $\mathcal{S}^{n-1}$ and consists of all points $\mathbf{x} \in \mathbb{R}^n$ of norm 1. The *unit circle* $\mathcal{S}^1$ can be seen as a subgroup of $\mathbb{C}^*$ by considering the map $\mathbb{R}^2 \to \mathbb{C}, (x, y) \mapsto x + iy$. The n-dimensional closed ball is denoted by $\mathcal{B}^n$ and consists of all points $\mathbf{x}$ of norm $\mathbf{x} \leq 1$. For any $r \in \mathbb{R}_{>0}$, we have

$$\mathrm{Vol}_n(r\mathcal{B}^n) = \frac{r^n \pi^{n/2}}{\Gamma(\frac{n}{2} + 1)} \geq (2r/\sqrt{n})^n \ , \tag{2.1}$$

where Γ denotes the *Gamma function*, which satisfies $\Gamma(n+1) = n!$ for $n \in \mathbb{N}$. The lower bound follows from $[-r/\sqrt{n}, r/\sqrt{n}]^n \subseteq r\mathcal{B}^n$.

2.1.3 Probability theory

The probability of an event E happening is denoted by $\Pr[E]$. For a random variable X, we denote its *expectation value* by $\mathbb{E}[X]$; its *variance* by $\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$; its *standard deviation* (std. dev.) by $\sigma_X = \sqrt{\mathbb{V}[X]}$.

For a probability distribution $\mathcal{D}$ on $\mathbb{R}$, we denote its support by $\operatorname{Supp}\mathcal{D}$, the *cumulative distribution function* (CDF) by $x \mapsto \Pr_{X\leftarrow\mathcal{D}}[X \leq x]$, and the *survival function* (SF) by $\Pr_{X\leftarrow\mathcal{D}}[X \geq x] = 1 - \Pr_{X\leftarrow\mathcal{D}}[X < x]$. The *probability density function* (PDF) at $x \in \mathbb{R}$ is the derivative of $\Pr_{X\leftarrow\mathcal{D}}[X \leq x]$ with respect to x.

The *uniform distribution on a set* X is denoted by $\mathrm{U}(X)$.

The n-dimensional *Gaussian mass with parameter* $\sigma \in \mathbb{R}_{>0}$ is denoted by

$$\rho_\sigma \colon \mathbb{R}^n \to \mathbb{R}_{>0}, \quad \mathbf{x} \mapsto \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right) .$$

The *Gaussian, or normal distribution, with centre* $\mu \in \mathbb{R}$ *and parameter* $\sigma \in \mathbb{R}_{>0}$, is denoted by $\mathcal{N}(\mu, \sigma^2)$, and its PDF is given by $\frac{1}{\sigma\sqrt{2\pi}}\,\rho_\sigma(x-\mu)$. Note the Gaussian $\mathcal{N}(\mu, \sigma^2)$ has expected value μ and standard deviation σ. The *standard Gaussian* is $\mathcal{N}(0, 1)$. Given $n \geq 1$, *the* n*-dimensional centred Gaussian with parameter* $\sigma \in \mathbb{R}_{>0}$ is denoted by $\mathcal{N}_n(0, \sigma)$ and is the joint distribution of points $\mathbf{x} \in \mathbb{R}^n$ where each coefficient is independently sampled according to $\mathcal{N}(0, \sigma)$.

The *error function* is given by $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathrm{d}t$. The CDF of the Gaussian $\mathcal{N}(\mu, \sigma^2)$ is expressible in terms of the error function as follows:

$$\Pr_{X\leftarrow\mathcal{N}(\mu,\sigma^2)}[X \leq x] = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \qquad (x \in \mathbb{R}). \tag{2.2}$$

Concretely, we have

$$\Pr_{X\leftarrow\mathcal{N}(\mu,\sigma^2)}[\mu - k\sigma \leq X \leq \mu + k\sigma] = \operatorname{erf}\left(\frac{k}{\sqrt{2}}\right) \approx \begin{cases} 68\% & k = 1, \\ 95\% & k = 2, \\ 99.7\% & k = 3. \end{cases}$$

The *complementary error function* is given by $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$, and has the following asymptotic decay rate.

Lemma 2.1 ([AS64, 7.1.24])**.** *For all $x > 0$,*

$$\operatorname{erfc}(x) \leq \frac{1}{\sqrt{\pi}x} \cdot e^{-x^2}.$$

Theorem 2.2 (central limit)**.** *Let $X_1, X_2, \ldots$ be* independent and identically distributed *(i.i.d.) random variables, each having mean μ and variance σ^2. As n tends to infinity, the random variable*

$$\sqrt{n} \cdot \left(\frac{X_1 + \cdots + X_n}{n} - \mu \right) ,$$

converges in distribution towards $\mathcal{N}(0, \sigma^2)$.

In concrete settings, we heuristically expect that above random variable is closely distributed as Gaussian for large n.

Heuristic 2.3 (central limit heuristic)**.** *Given some large number n and i.i.d. random variables $X_1, \ldots, X_n$, each having mean μ and variance σ^2, the random variable $(X_1 + \cdots + X_n)/n$ has distribution $\mathcal{N}(\mu, \sigma^2/n)$.*

Statistical distance and Rényi divergence

Definition 2.4. Let P, Q be two probability distributions on S. The *statistical distance, or total variation distance, between P and Q*, is

$$\Delta(P, Q) = \sup_{T \subseteq S} |P(T) - Q(T)| \ .$$

If S is discrete, we have $\Delta(P, Q) = \frac{1}{2} \sum_{x \in S} |P(x) - Q(x)|$.

The Rényi divergence [Rén61, vEH14] is used in security proofs to provide tighter reductions compared to a proof using statistical distance [Pre17, BLR+18].

Definition 2.5. Let P, Q be two probability distributions such that their supports satisfy $\operatorname{Supp}(P) \subseteq \operatorname{Supp}(Q)$. The *Rényi divergence of P from Q at order $a \in (1, \infty)$*, is given by

$$\mathrm{R}_a(P \parallel Q) = \left(\sum_{x \in \operatorname{Supp}(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{1/(a-1)} . \tag{2.3}$$

In the literature, one may find the Rényi divergence being defined as the logarithm of R_a in Equation (2.3).

In contrast to statistical distance or 'max-log distance' [MW17], Rényi divergence does not provide a metric, as it is not symmetric in its arguments and does not satisfy the triangle inequality.

If the Rényi divergence is finite, which it will be for our applications, we think of it as a value $1 + \delta$ for $\delta \geq 0$; the smaller δ is the closer the two distributions. Note that for any fixed P, Q on which the Rényi divergence is finite, $\mathrm{R}_a(P \parallel Q)$ is nondecreasing as a function of a, i.e. a larger order a defines a stricter notion of closeness.

Rényi divergence satisfies the following three properties.

Proposition 2.6. *Let $a \in (1, \infty)$ be an order. Two probability distributions P, Q satisfy:*

- *(data processing inequality) for any function f we have*

$$\mathrm{R}_a\Big(P^f \,\Big\|\, Q^f\Big) \leq \mathrm{R}_a(P \parallel Q) \quad ,$$

  in which P^f and Q^f denotes the distribution of $f(X)$ induced by sampling $X \leftarrow P$ and $X \leftarrow Q$, respectively.

- *(probability preservation) for any event $E \subseteq \mathrm{Supp}(P)$ we have*

$$Q(E) \geq P(E)^{a/(a-1)} / \mathrm{R}_a(P \parallel Q) \quad .$$

Moreover, (multiplicativity) for families of distributions $(P_i)_{i \in I}, (Q_i)_{i \in I}$, we have

$$\mathrm{R}_a\left(\prod_{i \in I} P_i \,\middle\|\, \prod_{i \in I} Q_i\right) = \prod_{i \in I} \mathrm{R}_a(P_i \parallel Q_i) \quad .$$

Often P is considered as an inexact realisation of some ideal distribution Q, and the event E as some adversary winning a search game; in this setting the probability preservation inequality allows us to bound from above the probability of an adversary winning the search game using distribution P. We also consider the order a as a parameter that we may optimise over.

2.1.4 Fourier transform

Definition 2.7. The *Fourier transform* of a function $f\colon \mathbb{R}^n \to \mathbb{R}$, for which $\int_{\mathbb{R}^n} f(\mathbf{x})^2 \mathrm{d}\mathbf{x}$ is a finite value, is the function $\widehat{f}\colon \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{f}(\mathbf{y}) = \int_{\mathbb{R}^n} e^{-2\pi i \langle \mathbf{x},\mathbf{y}\rangle} f(\mathbf{x}) \mathrm{d}\mathbf{x},$$

for all $\mathbf{y} \in \mathbb{R}^n$.

For example, the Fourier transform of ρ_σ is given by,

$$\widehat{\rho_\sigma}(\mathbf{y}) = \left(\sigma\sqrt{2\pi}\right)^n \rho_{1/(2\pi\sigma)}(\mathbf{y}).$$

The well-known Fourier inversion theorem states that, under certain convergence conditions, one may recover f from $\widehat{f}$ by:

$$f(\mathbf{x}) = \int_{\mathbb{R}^n} e^{2\pi i \langle \mathbf{x},\mathbf{y}\rangle} \widehat{f}(\mathbf{y}) \mathrm{d}\mathbf{y}.$$

If f is a *radial function*, i.e., there exists $g\colon \mathbb{R} \to \mathbb{R}$ such that $f(\mathbf{x}) = g(\|\mathbf{x}\|)$, then $\widehat{f}$ is also a radial function, and vice versa [SW71, Thm. 3.3].

2.2 Lattice

The main subject of this thesis is the theory of Euclidean lattices. For a thorough introduction, see [Sie89, Cas96].

Definition 2.8. A *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^n$.

Here, discrete means that the induced topology on Λ is the discrete topology, i.e., there is an open ball $r\mathcal{B}^n$ $(r > 0)$ around the origin such that $r\mathcal{B}^n \cap \Lambda = \{\mathbf{0}\}$ holds. In particular, there is a minimal distance between lattice points. For example, $\mathbb{Z}^n$ is a lattice. Figure 1.4 depicts two lattices in two-dimensional space. Important properties of lattices are rank, volume, successive minima and the covering radius, which we will cover below in this sequence.

Let us use $\mathrm{span}(S)$ to denote the $\mathbb{R}$-*linear span of a set* $S \subset \mathbb{R}^n$.

Definition 2.9 (rank)**.** The *rank* of a lattice $\Lambda \subset \mathbb{R}^n$ is denoted by $\mathrm{rk}(\Lambda)$, and equals the dimension of $\mathrm{span}(\Lambda)$. We say Λ is *full-rank* if $\mathrm{span}(\Lambda) = \mathbb{R}^n$, or equivalently the dimension of $\mathrm{span}(\Lambda)$ equals n.

Later, Proposition 2.59 will show that any lattice can be taken without loss of generality to be of full-rank after possibly applying an orthogonal transformation to it.

Definition 2.10 (volume)**.** The *volume*, or rather covolume, of a rank-k lattice $\Lambda \subseteq \mathbb{R}^n$ is $\det(\Lambda) = \mathrm{Vol}_k(\mathrm{span}(\Lambda)/\Lambda)$. A lattice Λ is called *unit-volume* if $\det(\Lambda) = 1$.

We may normalise a rank-k lattice Λ to have unit-volume, by considering the lattice

$$\frac{1}{\det(\Lambda)^{1/k}} \cdot \Lambda \ ,$$

in which each point in Λ is multiplied by $1/\det(\Lambda)^{1/k}$.

The volume is a measure of global sparsity of lattice points. Equivalently, $1/\det(\Lambda)$ measures the density of lattice points in Euclidean space, as the following proposition shows.

Proposition 2.11. *Let $\Lambda \subset \mathbb{R}^n$ be a full-rank lattice, and $V \subset \mathbb{R}^n$ be a measurable set. Then,*

$$\mathop{\mathbb{E}}_{\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)} \left[|(V + \mathbf{t}) \cap \Lambda| \right] = \frac{\mathrm{Vol}_n(V)}{\det(\Lambda)} \ .$$

Definition 2.12 (successive minima)**.** For $i \in [\mathrm{rk}(\Lambda)]$, the *$i$th successive minimum of* Λ is given by

$$\lambda_i(\Lambda) = \inf \{ r \in \mathbb{R}_{>0} \mid \dim \mathrm{span}(\Lambda \cap r\mathcal{B}^n) \geq i \} \ ,$$

or equivalently it is the smallest $r > 0$ such that there are at least i points in Λ of norm $\leq r$ that are $\mathbb{R}$-linearly independent. In particular, the *first minimum*, $\lambda_1(\Lambda)$, is the length of a shortest nonzero vector of Λ.

Note that we have $0 < \lambda_1(\Lambda) \leq \lambda_2(\Lambda) \leq \cdots \leq \lambda_k(\Lambda) < \infty$. For example, $\Lambda = \mathbb{Z}^n$ satisfies $\lambda_1(\Lambda) = \cdots = \lambda_n(\Lambda) = 1$.

Example 2.13. Given $n \geq 4$, the lattice $\Lambda = \mathbb{Z}^n + \mathbb{Z} \cdot (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})^\mathsf{T}$ has the same successive minima as $\mathbb{Z}^n$ and contains the sublattice $\mathbb{Z}^n$ of index $[\Lambda : \mathbb{Z}^n] = 2$.

Example 2.14. For $k \in \mathbb{Z}_{\geq 1}$, the *root lattice of type A* is given by

$$A_k = \left\{ \mathbf{x} \in \mathbb{Z}^{k+1} \,\middle|\, \sum_i \mathbf{x}_i = 0 \right\} \ .$$

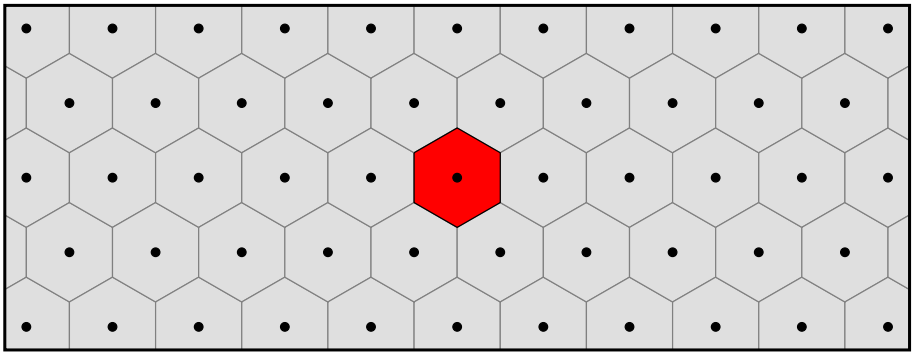


Figure 2.1: Voronoi diagram of the hexagonal lattice. The Voronoi cell, $\mathcal{V}(\Lambda)$, is highlighted in red.

The lattice A_k has rank k, volume $\sqrt{k+1}$, successive minima $\lambda_1(A_k) = \cdots = \lambda_k(A_k) = \sqrt{2}$, and $k(k+1)$ shortest vectors of this length [CS98].

Minkowski [Min1896] showed in 1896 that the first minimum can be bounded using $\det(\Lambda)$.

Theorem 2.15 (Minkowski's convex body theorem)**.** *Let $\Lambda \subset \mathbb{R}^n$ be a full-rank lattice and let $S \subseteq \mathbb{R}^n$ be of volume $\mathrm{Vol}_n(S) > 2^n \det(\Lambda)$, such that $-S \subseteq S$, and for all $\mathbf{x}, \mathbf{y} \in S$ and $\lambda \in [0,1]$ we have $\lambda\mathbf{x}+(1-\lambda)\mathbf{y} \in S$. Then, $S \cap \Lambda$ contains a nonzero lattice point.*

Proof. See Cassels [Cas96, III]. □

Minkowski's first theorem easily follows.

Theorem 2.16 (Minkowski's first)**.** *A full-rank lattice $\Lambda \subset \mathbb{R}^n$ satisfies*

$$\lambda_1(\Lambda) \leq 2 \cdot \frac{\det(\Lambda)^{1/n}}{\mathrm{Vol}_n(\mathcal{B}^n)^{1/n}} \ .$$

Proof. Consider $S = r\mathcal{B}^n$ for $r > 2 \cdot (\det(\Lambda)/\mathrm{Vol}_n(\mathcal{B}^n))^{1/n}$. As we have $\mathrm{Vol}_n(S) > 2^n \det(\Lambda)$, there exists $\mathbf{v} \in S \cap \Lambda \setminus \{\mathbf{0}\}$ so $\lambda_1(\Lambda) \leq r$. □

By placing balls of radius $\frac{1}{2}\lambda_1(\Lambda)$ centred at each lattice point we get a so-called lattice packing. A unit-volume lattice maximising its first minimum yields the densest lattice packing through this construction.

Table 2.1: Known Hermite constants

n	1	2	3	4	5	6	7	8	9	24
γ_n	1	$\sqrt{4/3}$	$\sqrt[3]{2}$	$\sqrt{2}$	$\sqrt[5]{8}$	$\sqrt[6]{\frac{64}{3}}$	$\sqrt[7]{64}$	2	2	4

Definition 2.17. *Hermite's constant* is given by

$$\gamma_n = \sup_{\Lambda \in \mathcal{X}_n} \lambda_1(\Lambda)^2 \ ,$$

where $\mathcal{X}_n$ is the family of rank-n unit-volume lattices.

By combining Theorem 2.16 and Equation (2.1), we obtain

$$\sqrt{\gamma_n} \leq 2\,\mathrm{GH}(n) \leq \sqrt{n}.$$

The values of γ_n are known for $n \leq 8$ [Bli29, Bli35], for $n = 24$ [CK09] and for $n = 9$ [DvW25], and can be found in Table 2.1. Hermite's constant in dimension 9 was determined using extensive machine-aided computation. A remarkable fact is that in dimension 8 and 24 the densest sphere packing is one in which the sphere centres form a lattice [Via17, CKM+17].

Definition 2.18 (Voronoi cell)**.** The *Voronoi cell of* Λ is denoted by $\mathcal{V}(\Lambda)$ and consists of all points that are not closer to any lattice point other than to the origin:

$$\mathcal{V}(\Lambda) = \{\mathbf{x} \in \mathrm{span}(\Lambda) \mid \forall \mathbf{v} \in \Lambda\colon \ \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{v}\|\} \ .$$

Note that we have $\det(\Lambda) = \mathrm{Vol}_k(\mathcal{V}(\Lambda))$ for any rank-k lattice Λ. Figure 2.1 shows that the Voronoi cell provides a tiling of Euclidean space, also known as a Voronoi diagram, named after Voronoï [Vor1908a, Vor1908b].

Definition 2.19 (covering radius)**.** The *covering radius* of a lattice Λ is denoted by $\mu(\Lambda)$ and equals the furthest distance from any point to Λ:

$$\mu(\Lambda) = \sup_{\mathbf{x} \in \mathrm{span}(\Lambda)} \left\{ \min_{\mathbf{v} \in \Lambda} \{\|\mathbf{x} - \mathbf{v}\|\} \right\} \ .$$

The function $d(\mathbf{x}, \Lambda) = \min_{\mathbf{v} \in \Lambda} \|\mathbf{x} - \mathbf{v}\|$ is Λ-periodic, i.e., for all $\mathbf{v} \in \Lambda$ we have $d(\mathbf{x}, \Lambda) = d(\mathbf{x} + \mathbf{v}, \Lambda)$. Hence, one may take this supremum over all $\mathbf{x} \in \mathcal{V}(\Lambda)$ instead. As the origin is the closest lattice point to any point in $\mathcal{V}(\Lambda)$, the covering radius equals

$$\mu(\Lambda) = \max\{\|\mathbf{x}\| \mid \mathbf{x} \in \mathcal{V}(\Lambda)\} \ .$$

2.2.1 Lattice basis

In order to algorithmically work with a lattice Λ, we describe a lattice by a so-called generating set, or a so-called basis.

Definition 2.20 (generating set and basis)**.** A *generating set* for a lattice $\Lambda \subset \mathbb{R}^n$ is a finite list of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \Lambda$ such that for all $v \in \Lambda$ there exist $c_1, \ldots, c_m \in \mathbb{Z}$ such that $\mathbf{v} = c_1\mathbf{b}_1 + \cdots + c_m\mathbf{b}_m$ holds.

A *basis* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m] \in \mathbb{R}^{n \times m}$ is a matrix such that $\mathbf{b}_1, \ldots, \mathbf{b}_m$ are $\mathbb{R}$-linearly independent. We say $\mathbf{B}$ is *a basis for* Λ whenever $\mathbf{b}_1, \ldots, \mathbf{b}_m$ is a generating set for Λ. Given a basis $\mathbf{B} \in \mathbb{R}^{n \times m}$, *the lattice generated by* $\mathbf{B}$ is denoted by

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{y} \in \mathbb{R}^n \mid \exists \mathbf{x} \in \mathbb{Z}^m \colon \mathbf{y} = \mathbf{B} \cdot \mathbf{x}\}.$$

A generating set for Λ requires at least $\mathrm{rk}(\Lambda)$ many vectors, and a basis for Λ contains exactly $k = \mathrm{rk}(\Lambda)$ column vectors. It is easily verifiable that $\mathcal{L}(\mathbf{B})$ is a lattice, and $\mathbf{B}$ is a basis for $\mathcal{L}(\mathbf{B})$. The set of square bases $n = m$ corresponds to $\mathrm{GL}_n(\mathbb{R})$. The following proposition shows that every lattice Λ has a basis.

Proposition 2.21 ([Kan83, Prop. 1.1])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice, and $\mathbf{v} \in \Lambda$ be a nonzero lattice point. If $x\mathbf{v} \notin \Lambda$ for all $x \in (0, 1)$, then there exists a basis $\mathbf{B}$ for Λ, such that $\mathbf{B}$ contains $\mathbf{v}$.*

Corollary 2.22. *Every lattice Λ can be described by a basis $\mathbf{B}$ such that $\mathbf{b}_1$ is a shortest vector.*

Proof. Let $\mathbf{v} \in \Lambda$ be a shortest vector of Λ. Suppose $x\mathbf{v} \in \Lambda$ would hold for some $x \in (0, 1)$. In this case, we would have $\|x\mathbf{v}\| = x\|\mathbf{v}\| < \|\mathbf{v}\|$, which would contradict that $\mathbf{v}$ is a shortest vector of Λ. Thus, we may use Proposition 2.21, which gives a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ for Λ such that $\mathbf{b}_i = \mathbf{v}$ for some $i \in [k]$. By swapping $\mathbf{b}_1$ and $\mathbf{b}_i$ if $i > 1$, we obtain the corollary. □

Given a lattice Λ, there is always a basis $\mathbf{B}$ for a *sublattice* of Λ such that for all $i \in [\mathrm{rk}(\Lambda)]$, we have $\mathbf{b}_i \in \Lambda$ and $\|\mathbf{b}_i\| = \lambda_i(\Lambda)$, but this is not always a basis for the lattice Λ itself. For example, given the lattice Λ in Example 2.13, such a basis is $\mathbf{B} = \mathbf{I}_n(\mathbb{Z})$, which is a basis for the sublattice $\mathbb{Z}^n \subset \Lambda$ whenever $n > 4$ but not for Λ itself.

Lemma 2.23 ([MG02, Lem. 7.1]). *There exists a polynomial-time algorithm that, on input a basis* $\mathbf{B} \in \mathbb{R}^{n\times k}$ *of a lattice* Λ *and* k *linearly independent lattice vectors* $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{s}_1\| \leq \|\mathbf{s}_2\| \leq \cdots \leq \|\mathbf{s}_k\|$ *holds, outputs a basis* $\mathbf{R}$ *for* Λ *such that* $\|\mathbf{r}_i\| \leq \|\mathbf{s}_i\| \cdot \max\{1, \sqrt{i}/2\}$ *holds for all* $i = 1, \ldots, k$.

Corollary 2.24 ([MG02, Cor. 7.2]). *Given a lattice* $\Lambda \subset \mathbb{R}^n$*, there exists a basis* $\mathbf{B}$ *for* Λ *such that* $\|\mathbf{b}_i\| \leq \lambda_i(\Lambda) \cdot \max\{1, \sqrt{i}/2\}$ *holds for all* $i = 1, \ldots, \mathrm{rk}(\Lambda)$.

There are some lists of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ that do not form a generating set. Namely, if such a list satisfies an $\mathbb{R}$-linear relation, it may not generate a discrete set. For example, the set $\mathbb{Z} + \sqrt{2}\mathbb{Z}$ is dense in $\mathbb{R}$ and hence not a lattice.

There is an efficient algorithm that, upon input a generating set for some lattice Λ, outputs a basis for Λ [MG02, Sec. 2.1].

Sublattices and basis transformations

If $\mathbf{B}$ is a basis for a rank-k lattice Λ, and $\mathbf{M} \in \mathbb{Z}^{k\times m}$ is a basis with $m \in [k]$, then $\mathbf{BM}$ is a basis for the sublattice $\mathcal{L}(\mathbf{BM})$, which is of rank m.

Two lattices $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{R}^{n\times k}$ generate the same lattice if and only if there exists a unimodular matrix $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{B}_2 = \mathbf{B}_1\mathbf{U}$ holds. This is an easy consequence of the above because a matrix $\mathbf{U}$ is invertible over the integers whenever $\det(\mathbf{U}) = \pm 1$.

Given a basis $\mathbf{B}$ for Λ, we have $\det(\Lambda) = \sqrt{\det(\mathbf{B}^\mathsf{T}\mathbf{B})}$, and if Λ is full-rank, $\det(\Lambda) = |\det \mathbf{B}|$. The quantity $\det(\mathbf{B}^\mathsf{T}\mathbf{B})$ is invariant under $\mathrm{GL}_n(\mathbb{Z})$. Namely, because unimodular matrices have determinant ± 1, we have $\det\Big((\mathbf{B}_1\mathbf{U})^\mathsf{T} \cdot \mathbf{B}_1\mathbf{U}\Big) = (\det \mathbf{U})^2 \cdot \det\Big(\mathbf{B}_1^\mathsf{T} \cdot \mathbf{B}_1\Big) = \det(\mathbf{B}_1^\mathsf{T}\mathbf{B}_1)$.

The group $\mathrm{GL}_k(\mathbb{Z})$ is of infinite cardinality when $k \geq 2$, and thus, any lattice of rank 2 and above, admits an infinite number of bases.

Orthogonal transformations

Given an orthogonal transformation $\mathbf{O} \in \mathrm{O}_n(\mathbb{R})$ and a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ for $\Lambda \subset \mathbb{R}^n$, then $\mathbf{OB}$ is a basis for the rotated lattice $\mathbf{O}\Lambda = \{\mathbf{Ov} : \mathbf{v} \in \Lambda\}$.

Remark 2.25. Technically, an orthogonal transformation is a combination of a rotation (having determinant 1) and a reflection. However, for ease

of writing, we may say 'rotation' or 'isometry' to mean an orthogonal transformation.

Definition 2.26 (isomorphic lattice)**.** A lattice Λ is *isomorphic* to another lattice Λ' if there exists an orthogonal transformation $\mathbf{O} \in \mathrm{O}_n(\mathbb{R})$ such that $\Lambda' = \mathbf{O} \cdot \Lambda$ holds.

2

Lattice isomorphism is an equivalence relation as $\mathrm{O}_n(\mathbb{R})$ is a group. We choose to call such a map an isomorphism as it preserves the inner product space, i.e., for all $\mathbf{x}, \mathbf{y} \in \Lambda$ we have

$$\langle \mathbf{Ox}, \mathbf{Oy} \rangle = \mathbf{x}^\mathsf{T}\mathbf{O}^\mathsf{T}\mathbf{Oy} = \mathbf{x}^\mathsf{T}\mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle \quad .$$

Recovering the orthogonal transformation is the Lattice Isomorphism Problem (LIP).

Problem 2.27 (Lattice Isomorphism Problem, LIP)**.** Let Λ_1, Λ_2 be two lattices that are isomorphic to one another. Given on input a basis $\mathbf{B}_1$ for Λ_1 and a basis $\mathbf{B}_2$ for Λ_2, find $\mathbf{O} \in \mathrm{O}_n(\mathbb{R})$ satisfying

$$\Lambda_2 = \mathbf{O} \cdot \Lambda_1 \quad .$$

Note that an orthogonal transformation $\mathbf{O}$ satisfies above equation if and only if there exists $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{B}_2 = \mathbf{OB}_1\mathbf{U}$.

Automorphisms

A lattice automorphism is an isomorphism from a lattice to itself.

Definition 2.28 (lattice automorphism)**.** Let $\Lambda \subset \mathbb{R}^n$ be a lattice. A *lattice automorphism of* Λ is an orthogonal matrix $\mathbf{O} \in \mathrm{O}_n(\mathbb{R})$ such that $\mathbf{O}\Lambda = \Lambda$ holds. The *group of automorphisms* of Λ is denoted by $\mathrm{O}(\Lambda)$ and consists of all lattice automorphisms of Λ.

Given a basis $\mathbf{B}$ for Λ, any automorphism provides a unimodular matrix $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{OB} = \mathbf{BU}$, since $\mathbf{OB}$ is another basis for Λ.

Any lattice automorphism group contains the 'trivial automorphisms' $\mathbf{I}_n(\mathbb{R})$ and $-\mathbf{I}_n(\mathbb{R})$.

Remark 2.29. Suppose Λ_1 is isomorphic to the lattice $\Lambda_2 = \mathbf{O}_{12}\Lambda_1$ for some $\mathbf{O}_{12} \in \mathrm{O}_n(\mathbb{R})$. First, there is a group isomorphism, $\mathrm{O}(\Lambda_1) \cong \mathrm{O}(\Lambda_2)$,

where a group isomorphism is given by $\mathbf{O}_1 \mapsto \mathbf{O}_{12}\mathbf{O}_1\mathbf{O}_{12}^{\mathsf{T}}$. Second, there is a bijection

$$\mathrm{O}(\Lambda_1) \to \{\mathbf{O} \in \mathrm{O}_n(\mathbb{R}) \mid \mathbf{O}\Lambda_1 = \Lambda_2\}\,, \quad \mathbf{O}_1 \mapsto \mathbf{O}_{12}\mathbf{O}_1 \ ,$$

where the image is the set of solutions to Problem 2.27. Hence, a solution to LIP is only unique up to right multiplication by an automorphism of Λ_1, and, similarly, unique up to left multiplication by an automorphism of Λ_2.

Example 2.30. The automorphisms of the *integer lattice*, $\mathbb{Z}^n$, are precisely the *signed permutations*: a permutation of the coordinates followed by a possible sign change on each coordinate independently. Abstractly we have

$$\mathrm{O}(\mathbb{Z}^n) \cong \{\pm 1\}^n \rtimes S_n \ ,$$

and the group contains $2^n \cdot n!$ elements.

Normal forms

There are two normal forms of interest: Hermite normal form and Smith normal form.

The Hermite normal form is useful for sublattices of $\mathbb{Z}^n$, and by scaling, to any lattice $\Lambda \subset \mathbb{Q}^n$.

Definition 2.31 (HNF)**.** A nonzero matrix $\mathbf{A} \in \mathbb{Z}^{n\times k}$ is said to be *in column-style Hermite normal form* (HNF) if there exist so-called 'pivots' $(p_i)_{i=1}^r$ for some $r \le k$ such that $1 \le p_1 < p_2 < \cdots < p_r \le n$ and:

(1) if $j > r$ then $\mathbf{A}_{ij} = 0$
(2) for all $j \le r$ and $i < p_j$, we have $\mathbf{A}_{ij} = 0$,
(3) $\forall j \in [r]\colon \mathbf{A}_{p_j,j} > 0$, and
(4) $\forall k < j \le r\colon 0 \le \mathbf{A}_{p_j,k} < \mathbf{A}_{p_j,j}$.

The Hermite normal form can be generalised from $\mathbb{Z}$ to a principal ideal domain [SL96].

Example 2.32. A square full-rank matrix $\mathbf{A}$ is in HNF, if $r = k$, $\forall i \in$

$[k]\colon p_i = i$, and it is of the form

$$\begin{pmatrix} \mathbf{A}_{11} & 0 & \cdots & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \mathbf{A}_{k1} & \mathbf{A}_{k2} & \cdots & \mathbf{A}_{kk} \end{pmatrix},$$

Proposition 2.33. *For any matrix $\mathbf{A} \in \mathbb{Z}^{n\times k}$, there exists a matrix $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{AU}$ is in HNF. If $\mathbf{A}$ is full-rank, then $\mathbf{U}$ is unique.*

Corollary 2.34. *Every lattice $\Lambda \subseteq \mathbb{Z}^n$ has a unique basis $\mathbf{B} \in \mathbb{Z}^{n\times k}$ in HNF.*

By reducing a matrix $\mathbf{A}$ using the action of $\mathrm{GL}_n(\mathbb{Z})$ on both left and right of $\mathbf{A}$, we obtain the Smith normal form.

Definition 2.35 (SNF)**.** A matrix $\mathbf{A} \in \mathbb{Z}^{n\times k}$ is in *Smith normal form* (SNF) if $\mathbf{A}_{11} \mid \mathbf{A}_{22} \mid \cdots \mid \mathbf{A}_{rr}$ for some $r \le k$ is a divisor chain of positive integers, and $\mathbf{A}$ is zero except for $\mathbf{A}_{ii}$ with $i \le r$,

Proposition 2.36. *For any matrix $\mathbf{A} \in \mathbb{Z}^{n\times k}$, there exist $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ and $\mathbf{V} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{UAV}$ is in SNF.*

2.2.2 Random lattice

To use lattices in cryptography, we ideally want to generate lattices using randomness. There are uncountably many full-rank lattices of volume 1 in any dimension $n \ge 2$.

As was shown by Siegel [Sie45], there is a natural Haar measure μ_n on the collection of full-rank, unit-volume lattices, which may be identified by the set $\mathcal{X}_n = \mathrm{SL}_n(\mathbb{R})/\mathrm{SL}_n(\mathbb{Z})$. Hence, there exists a probability distribution to sample random lattices according to the so-called invariant distribution μ_n.

Although such a distribution may have elegant theoretical properties, it is practically desirable to have a large enough distribution that is efficiently sampleable from. Moreover, integer or rational lattices may be easier to work with. In this direction, Goldstein and Mayer give an algorithm for generating dimension-n unit-volume lattices [GM03], which is Algorithm 2.1 below. We will explain shortly after Theorem 2.37 that

Algorithm 2.1. SampleRandom: sampler for Goldstein–Mayer lattices

Input: dimension n
Output: unit-volume lattice in dimension n

1: sample a large integer N
2: sample uniformly from the finite collection of sublattices $\Lambda \subseteq \mathbb{Z}^n$ with volume N
3: **return** $N^{-1/n} \cdot \Lambda$

Step 2 of Algorithm 2.1 is easily implemented if one only generates prime numbers in Step 1, while it may be hard for composite N.

For sufficiently large N, the output of Algorithm 2.1 is closely distributed to Siegel's definition of random lattices.

Theorem 2.37 ([GM03, Thm. 2.1]). *Let $A \subseteq \mathcal{X}_n$ be any measurable set such that its boundary has zero measure with respect to μ_n, and let $\mathcal{L}_{n,N}$ be the collection of full-rank sublattices of $\mathbb{Z}^n$ of volume N. Then,*

$$\lim_{N\to\infty} \frac{1}{|\mathcal{L}_{n,N}|} \cdot \sum_{\Lambda\in\mathcal{L}_{n,N}} \mathbf{1}_A(N^{-1/n}\Lambda) = \mu(A) \ .$$

For prime q, by Corollary 2.34, the collection $\mathcal{L}_{n,q}$ consists of lattices generated by a basis of the form

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ x_1 & \cdots & x_{i-1} & q & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} ,$$

for some $i \in [n]$ and $x_1, \ldots, x_{i-1} \in \{0, 1, \ldots, q-1\}$.

For a particular $i \in [n]$, there are q^{i-1} possible options for $x_1, \ldots, x_{i-1}$. Hence, for large q, most lattices in $\mathcal{L}_{n,q}$ are generated by a basis of the

form

$$\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ x_1 & \cdots & x_{n-1} & q \end{pmatrix} . \tag{2.4}$$

2

In this thesis, we are interested in q-ary lattices.

Definition 2.38 (q-ary lattice)**.** Let $q \geq 2$ be a prime number. A *q-ary lattice* is of the form

$$\mathcal{L}_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \exists \mathbf{y} \in \mathbb{Z}^n \colon \mathbf{x} \equiv \mathbf{A}\mathbf{y} \pmod q\} \ ,$$

where $n, m \in \mathbb{Z}_{\geq 1}$ and $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{m \times n}$.

The distribution of *random q-ary lattices* of dimensions (m, n) is given by $\mathcal{L}_q(\mathbf{A})$ where $\mathbf{A} \leftarrow \mathrm{U}((\mathbb{Z}/q\mathbb{Z})^{m \times n})$.

Given $n \leq m$, a sample $\mathbf{A} \leftarrow \mathrm{U}((\mathbb{Z}/q\mathbb{Z})^{m \times n})$ is full-rank with probability approximately $1/q$, in which case we have $[\mathcal{L}_q(\mathbf{A}) : q\mathbb{Z}^m] = q^n$ and $[\mathbb{Z}^m : \mathcal{L}_q(\mathbf{A})] = q^{m-n}$. Intuitively, every column of $\mathbf{A}$ increases the density of $\mathcal{L}_q(\mathbf{A})$ by a factor q.

Alternatively, one may consider the so-called *small-secret embedding* [BG14], i.e.,

$$\mathcal{L}'_q(\mathbf{A}) = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \in \mathbb{Z}^{m+n} \;\middle|\; \mathbf{x}_1 \equiv \mathbf{A} \cdot \mathbf{x}_2 \pmod q \right\} = \mathcal{L}_q\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{I}_n \end{bmatrix} \right) .$$

A q-ary lattice of the form $\mathcal{L}_q(\mathbf{A})$ is equivalent to a small-secret embedding if $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{m \times n}$ is full-rank. Namely, in this case, by permuting the rows of $\mathbf{A}$, we may obtain a matrix $\mathbf{A}' = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}$ such that the square matrix $\mathbf{A}_2$ is invertible, which in turn shows:

$$\mathcal{L}'_q\left(\mathbf{A}_1\mathbf{A}_2^{-1}\right) = \mathcal{L}_q(\mathbf{A}') \cong \mathcal{L}_q(\mathbf{A}) \ .$$

The class of *orthogonal q-ary lattices* of the form

$$\mathcal{L}_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x} \cdot \mathbf{A} \equiv \mathbf{0} \pmod q\} \ , \tag{2.5}$$

are related to q-ary lattices by the so-called notion of duality, which will be explained in Section 2.3.

Ajtai proves that this class exhibits a *worst case to average case reduction* for some lattice problems [Ajt96, Ajt99]. In other words, if there exists a polynomial time algorithm $\mathcal{A}$ that solves a lattice problem upon input a random lattice sampled from this class, then there is a polynomial time algorithm $\mathcal{B}$ that solves that lattice problem upon input any lattice of said distribution.

For more information on random lattices, see [AEN19].

Gaussian heuristic

Siegel's mean value theorem [Sie45] implies that for any bounded, compact set $S \subseteq \mathbb{R}^n$, we have

$$\mathop{\mathbb{E}}_{\Lambda \sim \mu_n} [|S \cap \Lambda \setminus \{\mathbf{0}\}|] = \mathrm{Vol}_n(S) \ . \tag{2.6}$$

Definition 2.39. For $n \in \mathbb{Z}_{\geq 1}$, let us denote the radius of the unit-volume ball by

$$\mathrm{GH}(n) = \mathrm{Vol}_n(\mathcal{B}^n)^{-1/n} \ .$$

By considering $S = \mathrm{GH}(n)\mathcal{B}^n$, we expect one nonzero lattice point in S (and its negation) by Equation (2.6). In addition, $\lambda_1(\Lambda)$ is very concentrated around $\mathrm{GH}(n)$ [Rog56, Söd11].

Theorem 2.40 ([LN25, Thm. 6])**.** *A Haar-random lattice $\Lambda \leftarrow \mu_n$ satisfies*

$$\left| \frac{\lambda_1(\Lambda)}{\mathrm{GH}(n)} - 1 \right| \leq \frac{\log \log n}{n} \ ,$$

with probability at least $1 - o(1)$ as $n \to \infty$.

These concentration results allow us to approximate $\lambda_1(\Lambda)$ for many lattices encountered in cryptography (although not all), even when these lattices are not entirely random. Given a lattice Λ and a bounded set $S \subset \mathbb{R}^n$, the *Gaussian heuristic* states

$$|\Lambda \cap S| \approx \mathrm{Vol}_n(S) / \det(\Lambda) \ ,$$

which is a heuristic version of Equation (2.6). The Gaussian heuristic leads to the following two heuristics regarding the length of short vectors.

Heuristic 2.41. *A random lattice $\Lambda \subset \mathbb{R}^n$, has $\lambda_1(\Lambda) = \mathrm{GH}(n) \cdot \det(\Lambda)^{1/n}$.*

Note Theorem 2.16 states $\lambda_1(\Lambda) \leq 2 \cdot \mathrm{GH}(n) \cdot \det(\Lambda)^{1/n}$. Moreover, applying Stirling's approximation on Equation (2.1) leads to

$$\mathrm{GH}(n) \sim \sqrt{n/(2\pi e)} \qquad (\text{as } n \to \infty).$$

2

It is known that $\mathrm{GH}(n) \approx \sqrt{n/(2\pi e)}$ for $n \geq 50$ [Che13].

Heuristic 2.42. *Given a random lattice $\Lambda \subset \mathbb{R}^n$ and some $r > 1$, we have*

$$\left|\left\{\mathbf{v} \in \Lambda \;\middle|\; \|\mathbf{v}\| \leq r \cdot \mathrm{GH}(n)\det(\Lambda)^{1/n}\right\}\right| \approx r^n.$$

In particular, the ith shortest lattice point $\mathbf{v}$ has a length of approximately $\|\mathbf{v}\| \approx \mathrm{GH}(n)\det(\Lambda)^{1/n} i^{1/n}$.

There exist constructions of lattices where the Gaussian heuristic does not hold. However, for our applications, i.e., random q-ary lattices with sufficiently large q or projected sublattices considered during lattice reduction, Heuristic 2.41 accurately describes $\lambda_1(\Lambda)$.

2.2.3 Hard problems

In this section, we state the lattice problems that are deemed to be computationally hard to solve. The fundamental hard problem in lattice-based cryptography is that of finding the shortest vector.

Our problem statements say input and/or output contain lattices and/or lattice points. Computationally, these are actually described respectively by a rational basis and/or a rational coefficient vector expressing a point as a linear combination of basis vectors of certain precision.

Problem 2.43 (shortest vector, SVP)**.** Given upon input a lattice $\Lambda \subset \mathbb{R}^n$, output a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

In practice, the lattice is specified by a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$, and the goal is to find coefficients $\mathbf{c}_1, \ldots, \mathbf{c}_k$ such that $\mathbf{v} = \sum_{i=1}^{k} c_i \mathbf{b}_i$ is a shortest vector.

Problem 2.44 (closest vector, CVP)**.** Given upon input a lattice Λ and a vector $\mathbf{t} \in \mathbb{R}^n$, find a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \|\mathbf{w} - \mathbf{t}\|$ for all $\mathbf{w} \in \Lambda$.

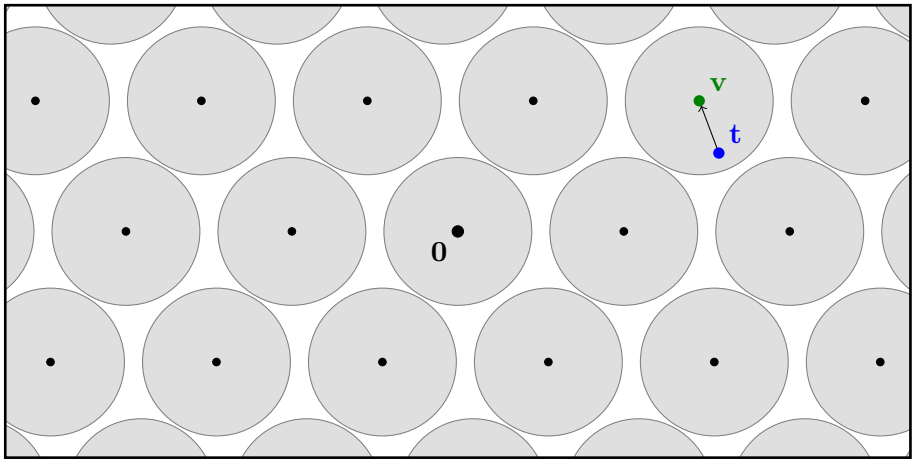


Figure 2.2: Search bounded distance decoding with $\alpha = 0.49$

Van Emde Boas [vEB81] proved NP-completeness of the decisional variants of two computational problems: (1) SVP with respect to ℓ^∞-norm, and (2) CVP, and conjectured SVP with respect to ℓ^2-norm is NP-complete as well. Remark that he phrased (1) and (2) the 'closest vector' and 'nearest vector' problem, respectively, which may be confusing with modern terminology. Ajtai [Ajt98] showed that SVP with respect to ℓ^2-norm is NP-hard for randomised reductions.

Bounded distance decoding

The *bounded distance decoding problem* (BDD), visualised in Figure 2.2, is a variant of CVP in which the distance from the target $\mathbf{t}$ to the lattice is guaranteed to be bounded in norm. The following computational problems are considered hard for specific parameters.

Problem 2.45 (search-BDD$_\alpha$, lattice form)**.** Given upon input a parameter $\alpha > 0$, a lattice $\Lambda \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$, such that $\mathbf{t} = \mathbf{v} + \mathbf{e}$ holds for some $\mathbf{v} \in \Lambda$ and $\|\mathbf{e}\| \leq \alpha\lambda_1(\Lambda)$, output $\mathbf{v}$.

By considering $\mathbf{t}$ modulo the lattice and demanding $\mathbf{t} - \mathbf{v}$ as a result, we get the syndrome form.

Problem 2.46 (search-BDD$_\alpha$, syndrome form)**.** Given upon input a parameter $\alpha > 0$, a lattice $\Lambda \subset \mathbb{R}^n$ and a target $\bar{\mathbf{t}} = \mathbf{e} + \Lambda \in \mathbb{R}^n/\Lambda$, such that $\|\mathbf{e}\| \leq \alpha\lambda_1(\Lambda)$ holds, output $\mathbf{e}$.

Concretely, in the syndrome form of search-BDD, one is given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ of Λ, and a target $\mathbf{t} = \mathbf{B} \cdot \mathbf{c}$ that is expressed in terms of $\mathbf{B}$ using coefficients $\mathbf{c} \in \mathbb{R}^k$ in the interval $[0, 1)$.

Note it is difficult to compute $\lambda_1(\Lambda)$ for an arbitrary lattice by the hardness of SVP, so it is hard to verify a candidate $\mathbf{v} \in \Lambda$, i.e., to check $\|\mathbf{t} - \mathbf{v}\| \leq \alpha\lambda_1(\Lambda)$. For random lattices, however, this is possible as we have $\lambda_1(\Lambda) \approx \mathrm{GH}(\Lambda)$ by Heuristic 2.41.

Remark 2.47. When search-BDD is instantiated with $\alpha < \frac{1}{2}$, it is guaranteed that there is a unique lattice point close enough to $\mathbf{t}$. Indeed, suppose $\mathbf{v}_1 \in \Lambda$ and $\mathbf{v}_2 \in \Lambda$ satisfy $\|\mathbf{v}_i - \mathbf{t}\| < \lambda_1(\Lambda)/2$ $(i = 1, 2)$. By the triangle inequality, we have $\|\mathbf{v}_1 - \mathbf{v}_2\| \leq \|\mathbf{v}_1 - \mathbf{t}\| + \|\mathbf{t} - \mathbf{v}_2\| < \lambda_1(\Lambda)$, so $\mathbf{v}_1 = \mathbf{v}_2$.

The following heuristic shows that there is still a unique lattice point close enough with high probability in average case search-BDD$_\alpha$ instances using $\alpha < 1$, a random lattice and $\mathbf{e} \leftarrow \mathrm{U}(\alpha\,\mathrm{GH}(n)\mathcal{B}^n)$.

Heuristic Claim 2.48. *Let $\Lambda \subset \mathbb{R}^n$ be a random unit-volume lattice, $\alpha \in (0, 1)$, and $R = \alpha\,\mathrm{GH}(n)$. The probability that a target $\mathbf{t} \leftarrow \mathrm{U}(R\mathcal{B}^n)$ is at a distance of at most R from some* nonzero *lattice point $\mathbf{v} \in \Lambda$ is at most $O(n\sqrt{n})\alpha^n$.*

This claim can be justified using Heuristic 2.42, and an upper bound by Micciancio and Voulgaris on spherical domes [MV10, Lem. 4.1].

Heuristic Justification. Note that only lattice points $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$ are relevant with $\|\mathbf{v}\| \leq 2R = 2\alpha\,\mathrm{GH}(n)$ by the triangle inequality. For such a $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$, we are interested in $\mathrm{Vol}_n\left(R\mathcal{B}^n \cap (\mathbf{v} + R\mathcal{B}^n)\right)$, which is twice the volume of the spherical dome $\{\mathbf{t} \in R\mathcal{B}^n \,|\, \langle \mathbf{t}, \mathbf{v}\rangle \geq \frac{1}{2}\|\mathbf{v}\|^2\}$. Set $\alpha = \|\mathbf{v}\|/2R$. This spherical dome is contained in a cylinder with base $R\sqrt{1-\alpha^2} \cdot \mathcal{B}^{n-1}$ and height $R(1-\alpha)$, which has volume at most $R^n(1-\alpha^2)^{n/2}\,\mathrm{Vol}_{n-1}(\mathcal{B}^{n-1})$. One can show that $\mathrm{Vol}_{n-1}(\mathcal{B}^{n-1}) \leq \frac{\sqrt{en}}{2}\,\mathrm{Vol}_n(\mathcal{B}^n)$ holds, which implies

$$\mathrm{Vol}_n(R\mathcal{B}^n \cap (\mathbf{v} + R\mathcal{B}^n)) \leq O(\sqrt{n})\alpha^n\left(1-\alpha^2\right)^{n/2}.$$

Heuristic 2.42 predicts approximately ℓ^n lattice points in a ball of radius $\ell\,\mathrm{GH}(n)$. By using this estimate for $\ell \in (1, 2\alpha)$, the volume of all

the spherical domes is roughly

$$\int_1^{2\alpha} n\ell^{n-1} O(\sqrt{n})\alpha^n \left(1 - \frac{\ell^2}{4\alpha^2}\right)^{n/2} \mathrm{d}\ell \leq O(n\sqrt{n})\alpha^n \int_1^{2\alpha} \left(\ell^2 - \frac{\ell^4}{4\alpha^2}\right)^{n/2} \mathrm{d}\ell.$$

The integrand reaches its maximum α^n at $\ell = \sqrt{2}\alpha$, so the above equation can be bounded from above by $O(n\sqrt{n})\alpha^{2n}(2\alpha - 1)$. For the desired probability, we consider the ratio of volumes, which is at most $O(n\sqrt{n})\alpha^{2n}/\operatorname{Vol}_n(R\mathcal{B}^n) = O(n\sqrt{n})\alpha^n$. △

Alternatively, there is also a decisional version of BDD.

Problem 2.49 (decision-BDD)**.** Given upon input a parameter $\alpha > 0$, a lattice $\Lambda \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$, output 'yes' if there exists $\mathbf{v} \in \Lambda$ such that $\|\mathbf{t} - \mathbf{v}\| \leq \alpha\lambda_1(\Lambda)$, and otherwise 'no'.

In this thesis, we are interested in average case variants of search-BDD and decision-BDD. Here, one should output 'yes' on input a target $\mathbf{t}$ drawn from a *BDD distribution* χ on $\mathbb{R}^n$ such that almost all samples have a norm around $\alpha\lambda_1(\Lambda)$ for some $\alpha > 0$. At the same time, one should output 'no' on input a target $\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$ drawn from the uniform distribution. Since the BDD and uniform distribution may overlap, average case BDD cannot always be solved. Rather, a successful algorithm both rarely outputs 'yes' when $\mathbf{t}$ is sampled uniformly (*false positive*), and rarely outputs 'no' when $\mathbf{t}$ is sampled from χ (*false negative*). The statistical distance between the BDD and uniform distribution is large if α is small (e.g., $\alpha = \frac{1}{2}$) and n is large, in which case a successful algorithm may exist, albeit not necessarily efficient.

Problem 2.50 (average case search-BDD$_\alpha$)**.** Let $\alpha > 0$ and let χ be a distribution on $\mathbb{R}^n$ such that $\|\mathbf{t}\| \approx \alpha\lambda_1(\Lambda)$ is likely if $\mathbf{t} \leftarrow \chi$. Given upon input α, a lattice $\Lambda \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$ such that $\mathbf{t} = \mathbf{v} + \mathbf{e}$ holds for some $\mathbf{v} \in \Lambda$, and $\mathbf{e} \leftarrow \chi$, output $\mathbf{v}$.

Problem 2.51 (average case decision-BDD$_\alpha$)**.** Let $\alpha > 0$ and let χ be a distribution on $\mathbb{R}^n$ such that $\|\mathbf{t}\| \approx \alpha\lambda_1(\Lambda)$ is likely if $\mathbf{t} \leftarrow \chi$. Given upon input α, a lattice $\Lambda \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$ sampled either from $\chi \pmod{\Lambda}$ or $\mathrm{U}(\mathbb{R}^n/\Lambda)$, output 'yes' in the former case and 'no' in the latter case.

Usually, in search-BDD and decision-BDD, the error distribution χ is a uniform distribution on a sphere or ball of some radius R, or a (discrete, see Section 2.4) Gaussian with some parameter σ. The hardness of search-BDD and decision-BDD then depends on the radius R or parameter σ, i.e., the bigger R or σ is, the larger the average error norm is and thus the harder the problem instances are.

2

Unique shortest vector

In certain lattices, the shortest vector is of much smaller norm than other vectors that are not parallel to it, in which case SVP is not harder than for a general lattice.

Problem 2.52 (unique shortest vector, $\mathrm{UniqueSVP}_\gamma$)**.** Given upon input $\gamma \in \mathbb{R}_{\geq 1}$ and a lattice Λ with the promise $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, output a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

Lyubashevsky and Micciancio [LM09] show two polynomial-time reductions: (1) from $\mathrm{BDD}_{1/2\gamma}$ to $\mathrm{UniqueSVP}_\gamma$ given any $\gamma \geq 1$, and (2) from $\mathrm{UniqueSVP}_\gamma$ to $\mathrm{BDD}_{1/\gamma}$ given any γ polynomial in n. Hence, given $\alpha < \frac{1}{2}$, BDD_α is essentially as hard as $\mathrm{UniqueSVP}_\gamma$ for some $\gamma \in [\frac{1}{2\alpha}, \frac{1}{\alpha}]$.

Learning with errors

It is common in cryptography to base security on the hardness of the learning with errors problem, which was introduced by Regev in 2005 [Reg09]. Learning with errors is basically the BDD problem using q-ary lattices.

Problem 2.53 (learning with errors, LWE)**.** Let $\mathbf{A} \leftarrow \mathrm{U}((\mathbb{Z}/q\mathbb{Z})^{m\times n})$ be a random q-ary matrix, χ_s be a secret distribution on $(\mathbb{Z}/q\mathbb{Z})^n$, and χ_e an error distribution on $\mathbb{Z}^m$. Given upon input the pair $(\mathbf{A}, \mathbf{b})$ where $\mathbf{b} \equiv \mathbf{A}\mathbf{s} + \mathbf{e} \pmod q$ is computed after sampling $\mathbf{s} \leftarrow \chi_\mathrm{s}$ and $\mathbf{e} \leftarrow \chi_\mathrm{e}$, output the vector $\mathbf{s}$.

A standard choice for the error distribution is the discrete Gaussian (see Section 2.4) $\chi_\mathrm{e} = \mathcal{D}_{\mathbb{Z}^m,\sigma}$ with parameter $\sigma > 0$. In a worst-case analysis of LWE, one determines, given any $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$, the difficulty of recovering $\mathbf{s}$. On the other hand, in cryptographic constructions, such as ML-KEM and ML-DSA, the secret $\mathbf{s}$ always has small norm, for example by taking $\chi_\mathrm{s} = \chi_\mathrm{e}$ and/or restricting secret coefficients to the set $\{-3, -2, \ldots, 2, 3\}$, which is the case for ML-KEM and ML-DSA.

2.2.4 Projection

First, we recall some linear algebra and discuss Gram–Schmidt orthogonalisation and QR decomposition. These give rise to projected sublattices, which are used in basis reduction to map a high dimensional lattice to a lower dimensional one.

Definition 2.54 (orthogonality)**.** A vector $\mathbf{v}$ is *orthogonal to a vector* $\mathbf{w} \in \mathbb{R}^n$ if $\langle \mathbf{v}, \mathbf{w} \rangle = 0$, and *orthogonal to a subspace* $V \subseteq \mathbb{R}^n$ if $\forall \mathbf{w} \in V\colon \langle \mathbf{v}, \mathbf{w} \rangle = 0$.

Given a subspace $V \subseteq \mathbb{R}^n$, the *orthogonal subspace* $V^\perp$ consists of all $\mathbf{v} \in \mathbb{R}^n$ orthogonal to V. Moreover, $\pi_V\colon \mathbb{R}^n \to V$ is the projection onto V, and $\pi_V^\perp = \pi_{V^\perp}$ is the projection orthogonally away from V.

Note for all $\mathbf{x} \in \mathbb{R}^n$, we have $\mathbf{x} = \pi_V(\mathbf{x}) + \pi_V^\perp(\mathbf{x})$. Moreover, as $\pi_V(\mathbf{x})$ is orthogonal to $\pi_V^\perp(\mathbf{x})$, by Pythagoras' theorem, we have

$$\|\mathbf{x}\|^2 = \|\pi_V(\mathbf{x})\|^2 + \|\pi_V^\perp(\mathbf{x})\|^2 \ , \tag{2.7}$$

so in particular $\|\pi_V(\mathbf{x})\| \leq \|\mathbf{x}\|$, and $\|\pi_V^\perp(\mathbf{x})\| \leq \|\mathbf{x}\|$.

Given a basis, we are interested in the subspaces generated by subarrays of a basis $\mathbf{B}$, and the orthogonal subspaces of these.

Gram–Schmidt orthogonalisation

Definition 2.55 (basis projection)**.** Given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ and $\ell \in [k]$, the projection orthogonally away from $[\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1}]$ is denoted by

$$\pi_{\mathbf{B},\ell} = \pi^\perp_{\mathrm{span}(\mathbf{b}_1,\ldots,\mathbf{b}_{\ell-1})} \ ,$$

where we may omit $\mathbf{B}$ if it is clear from context. In particular, we have $\pi_1(\mathbf{x}) = \mathbf{x}$.

Definition 2.56 (Gram–Schmidt)**.** Given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$, the ith *Gram–Schmidt vector* is $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$ for $i \in [k]$. The *Gram–Schmidt basis* is given by $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_k^*]$, and the *Gram–Schmidt coefficients* μ_{ij} where $i, j \in [k]$ are given by

$$\mu_{ij} = \langle \mathbf{b}_i^*, \mathbf{b}_j \rangle / \|\mathbf{b}_i^*\|^2 \ .$$

Note $\mathbf{B}^*$ is an orthogonal basis. Moreover, note $\mu_{ij} = 0$ when $j < i$ and $\mu_{ii} = 1$. In other words, writing $\mathbf{M} = (\mu_{ij})_{i,j=1}^k$ shows that $\mathbf{M}$ is

unitriangular. We have $\mathbf{b}_i = \mu_{1i}\mathbf{b}_1^* + \cdots + \mu_{i-1,i}\mathbf{b}_{i-1}^* + \mathbf{b}_i^*$, or in matrix form,

$$\mathbf{B} = \mathbf{B}^* \cdot \mathbf{M} \ .$$

By the above, we get $\det \Lambda = |\det \mathbf{B}| = |\det(\mathbf{B}^*)| = \prod_{i=1}^{n} \|\mathbf{b}_i^*\|$.

2

The *Gram–Schmidt orthogonalisation (GSO)* of $\mathbf{B}$ is the pair $(\mathbf{d}, \mathbf{M})$ where $\mathbf{d} \in \mathbb{R}^k$ satisfies $\mathbf{d}_i = \|\mathbf{b}_i^*\|$ for $i \in [k]$. An approximation of the GSO can be computed, assuming no numerical issues arise, by setting $\mathbf{b}_1^* = \mathbf{b}_1$, and then iteratively computing $\mathbf{b}_j^* = \mathbf{b}_j - \sum_{i<j} \mu_{ij}\mathbf{b}_i^*$ for $j = 2, \ldots, n$ using floating-point numbers with certain precision.

For a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ of a lattice Λ and $1 \le i \le j \le k$, we may consider the *projected sublattice* $\mathbf{B}_{[i,j]} = [\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j)]$. Note $\mathbf{B}_{[i,k]}$ is a basis for the lattice $\pi_i(\Lambda)$.

Lemma 2.57. *Given a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ of Λ, we have*

$$\lambda_1(\Lambda) \ge \min\{\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_k^*\|\} \ .$$

Proof. Suppose $\mathbf{v} \in \Lambda$ is nonzero, and write $\mathbf{v} = \sum_{i=1}^{k} c_i\mathbf{b}_i$ with $c_i \in \mathbb{Z}$. Let $i \in [k]$ be maximal such that $c_i \neq 0$ holds, and note such i exists as $\mathbf{v}$ is nonzero. Using Equation (2.7) and $\pi_i(\mathbf{v}) = c_i \cdot \pi_i(\mathbf{b}_i)$, we obtain

$$\|\mathbf{v}\| \ge \|\pi_i(\mathbf{v})\| = |c_i| \cdot \|\mathbf{b}_i^*\| \ge \|\mathbf{b}_i^*\| \ . \qquad \square$$

QR decomposition

The QR decomposition [Hig02, Chapter 19] is related to Gram–Schmidt orthogonalisation.

Definition 2.58 (QR decomposition)**.** The *QR decomposition of a basis* $\mathbf{B} \in \mathbb{R}^{n\times k}$ is the pair $(\mathbf{Q}, \mathbf{R})$ with $\mathbf{Q} \in \mathbb{R}^{n\times k}$ and $\mathbf{R} \in \mathbb{R}^{k\times k}$ such that $\mathbf{R}$ is an upper triangular matrix with all diagonal entries in $\mathbb{R}_{>0}$, $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}_k(\mathbb{R})$ and $\mathbf{B} = \mathbf{Q} \cdot \mathbf{R}$. Here $\mathbf{R}$ denotes the *R-factor* of $\mathbf{B}$.

There are four remarks to make.

(1) A QR decomposition is unique per basis.
(2) The matrix $\mathbf{Q}$ can be extended by some $\mathbf{Q}' \in \mathbb{R}^{n\times(n-k)}$ to an orthogonal transformation $[\mathbf{Q} \mid \mathbf{Q}'] \in \mathrm{O}_n(\mathbb{R})$.
(3) Software libraries such as LAPACK and NumPy may return an R-factor, in which some diagonal entries are negative.

(4) The QR decomposition $(\mathbf{Q}, \mathbf{R})$ is related to the GSO $(\mathbf{d}, \mathbf{M})$ by

$$\mathbf{Q} = \left[\frac{\mathbf{b}_1^*}{\|\mathbf{b}_1^*\|} \quad \cdots \quad \frac{\mathbf{b}_k^*}{\|\mathbf{b}_k^*\|} \right], \quad \text{and} \quad \mathbf{R} = \operatorname{diag}(\mathbf{d}) \cdot \mathbf{M} \ ,$$

$$\text{so } \mathbf{R}_{ii} = \|\mathbf{b}_i^*\| \text{ and } \mathbf{R}_{ij} = \langle \mathbf{b}_i^*, \mathbf{b}_j \rangle / \|\mathbf{b}_i^*\| = \mathbf{d}_i \mu_{ij} \text{ for } i, j \in [k].$$

The QR decomposition and GSO can both be computed using at most $\mathcal{O}(nk^2)$ floating-point operations. The QR decomposition, or only the R-factor, can be computed using Householder reflections [Hig02, Chapter 19]. Householder reflections are more numerically stable than the GSO, which is useful when computing floating-point approximations of inner products $\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle$, like will be done in Section 2.5.

Proposition 2.59. *Let $\Lambda \subseteq \mathbb{R}^n$ be a rank-k lattice. Then, there exists a full-rank lattice $\Lambda' \subseteq \mathbb{R}^k$ such that Λ is isomorphic to*

$$\left\{ (v_1, \ldots, v_n)^\mathsf{T} \in \mathbb{R}^n \;\middle|\; (v_1, \ldots, v_k)^\mathsf{T} \in \Lambda', \text{ and } v_{k+1} = \cdots = v_n = 0 \right\} \ .$$

Moreover, given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ of Λ, then its R-factor is a basis for Λ' satisfying the above.

Proof. The second remark above shows there exists $\mathbf{Q}' \in \mathrm{O}_n(\mathbb{R})$ such that $\mathbf{B} = \mathbf{Q}' \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$ holds. This is equivalent to ${\mathbf{Q}'}^\mathsf{T} \mathbf{B} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$ and the claim follows. □

2.2.5 Lattice isomorphism

Recalling paragraph § Orthogonal transformations (p. 33), if Λ, Λ' are generated by $\mathbf{B}, \mathbf{B}'$ respectively, then they are isomorphic if there exists an orthogonal transformation $\mathbf{O} \in \mathrm{O}_n(\mathbb{R})$ and a unimodular matrix $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{O} \cdot \mathbf{B} \cdot \mathbf{U} = \mathbf{B}'$. We can remove the real valued orthogonal transformation by moving to quadratic forms.

Definition 2.60 (quadratic form)**.** A *quadratic form* is a positive definite real symmetric matrix $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$. Given two quadratic forms $\mathbf{Q}, \mathbf{Q}' \in \mathcal{S}_n^{>0}(\mathbb{R})$, an *isomorphism from $\mathbf{Q}$ to $\mathbf{Q}'$* is a unimodular $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{U}^\mathsf{T} \cdot \mathbf{Q} \cdot \mathbf{U} = \mathbf{Q}'$ holds. We say $\mathbf{Q}$ and $\mathbf{Q}'$ are *isomorphic* if there exists an isomorphism from $\mathbf{Q}$ to $\mathbf{Q}'$.

For any lattice basis $\mathbf{B}$ the Gram matrix $\mathbf{B}^\mathsf{T}\mathbf{B}$, consisting of all pairwise inner products, is a quadratic form. Conversely, the *Cholesky decomposition* of a quadratic form $\mathbf{Q}$ is the unique basis $\mathbf{R}$ that is an upper triangular matrix with strictly positive diagonal such that $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{Q}$ holds. Note in this section, $\mathbf{Q}$ denotes a quadratic form and is not related to QR decomposition.

We have that two bases generate isomorphic lattices if and only if their Gram matrices are isomorphic. This allows us to restate Problem 2.27.

Problem 2.61 (LIP, restated)**.** Given upon input two isomorphic quadratic forms $\mathbf{Q}, \mathbf{Q}' \in \mathcal{S}_n^{>0}(\mathbb{R})$, output any isomorphism from $\mathbf{Q}$ to $\mathbf{Q}'$.

Analogously to Definition 2.28, we define the *orthogonal group* of a quadratic form $\mathbf{Q}$ as follows.

Definition 2.62. Let $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$ be a quadratic form. An *automorphism of* $\mathbf{Q}$ is an isomorphism from $\mathbf{Q}$ to $\mathbf{Q}$. The *group of automorphisms of* $\mathbf{Q}$ is denoted by $\mathrm{O}(\mathbf{Q})$ and consists of all automorphisms of $\mathbf{Q}$.

ℤLIP is a problem that asks, given a lattice isomorphic to $\mathbb{Z}^n$, to output any isomorphism. Using quadratic forms, we get the following elegant problem description.

Problem 2.63 (ℤLIP)**.** Given upon input a quadratic form $\mathbf{Q} \in \mathrm{GL}_n(\mathbb{Z})$ for which there exists $\mathbf{B} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{Q} = \mathbf{B}^\mathsf{T}\mathbf{B}$ holds, output any such $\mathbf{B}$.

If one knows such a $\mathbf{B}$, it becomes easy to compute $\mathrm{O}(\mathbf{Q})$ since we know $\mathrm{O}(\mathbb{Z}^n)$. Without $\mathbf{B}$, we may compute the trivial automorphisms $\pm\mathbf{I}_n$. It is believed to be difficult in general, however, to construct any other automorphism of $\mathbf{Q}$ without solving ℤLIP.

Definition 2.64. The *inner product with respect to a quadratic form* $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$ is given by,

$$\langle \cdot, \cdot \rangle_\mathbf{Q} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^\mathsf{T} \cdot \mathbf{Q} \cdot \mathbf{y} \ .$$

The *norm with respect to* $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$ is defined as $\|\mathbf{x}\|_\mathbf{Q} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_\mathbf{Q}}$.

Note that for a basis $\mathbf{B}$ and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we have

$$\langle \mathbf{Bx}, \mathbf{By} \rangle = \mathbf{x}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{By} = \langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{B}^\mathsf{T}\mathbf{B}} \ ,$$

and thus the geometry of $\Lambda(\mathbf{B})$ is fully described by $\mathbf{Q} = \mathbf{B}^\mathsf{T}\mathbf{B}$. Moving from (bases of) lattices to quadratic forms can be viewed as forgetting about the specific embedding of the lattice in $\mathbb{R}^n$, while maintaining all geometric information. Throughout this thesis we will talk about lattices and quadratic forms interchangeably.

2.2.6 Module lattice and Hermitian form

A number field $\mathbb{K}$ is an algebraic extension of $\mathbb{Q}$ of finite degree $n = [\mathbb{K} : \mathbb{Q}]$. We write $\mathcal{O}_\mathbb{K}$ for the ring of integers of a general number field. In this work, we consider the cyclotomic field $\mathbb{K} = \mathbb{Q}(\zeta_{2n}) = \mathbb{Q}\left(e^{-2\pi i/2n}\right) \cong \mathbb{Q}[X]/(X^n+1)$ where $n \geq 2$ is a power of two. This is a CM field and has conductor $2n$. Many of the facts below are not true for general number fields.

The ring of integers $R = \mathcal{O}_\mathbb{K} \cong \mathbb{Z}[X]/(X^n+1)$, or any ideal of $\mathbb{K}$, obtains a rank-n lattice structure by considering its image under the so-called *canonical embedding*, or Minkowski embedding:

$$\sigma\colon \mathbb{K} \to \mathbb{C}^n,\ x \mapsto (\sigma_1(x), \ldots, \sigma_n(x)) \ .$$

Here, $\sigma_1, \sigma_2, \ldots, \sigma_n$ are the n canonical embeddings of $\mathbb{K}$ into $\mathbb{C}$, ordered such that $\sigma_{i+n/2} = \overline{\sigma_i}$ for $i \in [n/2]$ (for $n \geq 2$ there are no real embeddings). The subset $\{(x_1, \ldots, x_n) \in \mathbb{C}^n\colon \forall i \in [n/2], x_{i+n/2} = \overline{x_i}\} \subset \mathbb{C}^n$ is isomorphic as an inner product space to $\mathbb{R}^n$ [LPR10, Sec. 2.1]. We implicitly use this isomorphism and write $\sigma\colon \mathbb{K} \to \mathbb{R}^n$. We also have the *coefficient embedding*,

$$\textsc{Vec}\colon \mathbb{K} \to \mathbb{Q}^n, \quad a_0 + a_1 X + \cdots + a_{n-1}X^{n-1} \mapsto \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \ ,$$

which is an additive group isomorphism. Moreover, $\textsc{Vec}(R) = \mathbb{Z}^n$.

The algebraic norm and trace are given by $\mathrm{N}(x) = \prod_{i=1}^n \sigma_i(x)$ and $\mathrm{Tr}(x) = \sum_{i=1}^n \sigma_i(x)$ for $x \in \mathbb{K}$, respectively. Since the σ_i are ring homomorphisms the algebraic norm is multiplicative and the trace is additive. If $x \in R$ then $\mathrm{N}(x), \mathrm{Tr}(x) \in \mathbb{Z}$.

The embeddings enable us to view $\mathbb{K}$ as an inner product space over $\mathbb{Q}$ by defining $\langle \cdot, \cdot \rangle : \mathbb{K} \times \mathbb{K} \to \mathbb{Q}$ as

$$\langle f, g \rangle = \frac{1}{n} \cdot \sum_{i=1}^{n} \overline{\sigma_i(f)} \cdot \sigma_i(g). \tag{2.8}$$

2

We renormalise by $\frac{1}{n}$ as there is an isometry, up to a scaling factor of n, from the canonical embedding to the coefficient embedding, i.e., we have $\langle f, g \rangle = \langle \textsc{Vec}(f), \textsc{Vec}(g) \rangle$ with the right hand inner product over $\mathbb{R}^n$. This gives a geometric norm on $\mathbb{K}$ as $\| \cdot \| : \mathbb{K} \to \mathbb{Q},\ f \mapsto \sqrt{\langle f, f \rangle}$, which agrees with the Euclidean norm of $\textsc{Vec}(f)$. In particular, given any $f = f_0 + f_1 X + \cdots + f_{n-1} X^{n-1} \in \mathbb{K}$, note $\langle 1, f \rangle = f_0$.

Because $\mathbb{K}$ is a CM field, it comes with a unique involutive automorphism $(\,\cdot\,)^\star : \mathbb{K} \to \mathbb{K}$ that acts as complex conjugation on its embeddings, i.e., we have $\sigma_i(x^\star) = \overline{\sigma_i(x)}$ for all $x \in \mathbb{K}$ and $i \in [n]$. We call $(\,\cdot\,)^\star$ *complex conjugation*, and we say $f \in \mathbb{K}$ is *self-adjoint* if $f^\star = f$. For all $f = f_0 + f_1 X + \cdots + f_{n-1} X^{n-1} \in \mathbb{K}$, we have

$$f^\star = f_0 - f_{n-1} X - \cdots - f_1 X^{n-1} \ .$$

Moreover, Equation (2.8) becomes,

$$\langle f, g \rangle = \frac{1}{n} \sum_{i=1}^{n} \sigma_i(f^\star g) = \frac{1}{n} \operatorname{Tr}(f^\star g) \ .$$

Module lattice

For any $\ell \in \mathbb{Z}_{\geq 1}$, we define $\mathbb{K}^\ell = \underbrace{\mathbb{K} \oplus \cdots \oplus \mathbb{K}}_{\ell \text{ times}}$, and similarly for the R-module R^ℓ of rank ℓ. Extend $\textsc{Vec} \colon \mathbb{K}^\ell \to \mathbb{Q}^{n\ell}$ in the natural way. We extend the inner product and norm to vectors $\mathbf{f}, \mathbf{g} \in \mathbb{K}^\ell$ by

$$\langle \mathbf{f}, \mathbf{g} \rangle = \sum_{i=1}^{\ell} \langle f_i, g_i \rangle \ , \qquad \text{and} \qquad \|\mathbf{f}\| = \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle} \ .$$

We write $\textsc{Rot}(f) = \big(\textsc{Vec}(f), \textsc{Vec}(Xf), \ldots, \textsc{Vec}(X^{n-1} f)\big) \in \mathbb{Q}^{n \times n}$ for $f \in \mathbb{K}$, which is a basis for the lattice $\sigma(f)$ given by the (possibly fractional) ideal (f). Note that $\textsc{Rot}$ is a ring homomorphism, satisfying

$\textsc{Vec}(xy) = \textsc{Rot}(x)\,\textsc{Vec}(y)$ and $\textsc{Rot}(xy) = \textsc{Rot}(x)\,\textsc{Rot}(y)$ for $x, y \in \mathbb{K}$. We extend Rot to matrices $\mathbf{B} \in \mathbb{K}^{k\times\ell}$ in the natural way

$$\textsc{Rot}(\mathbf{B}) = \begin{pmatrix} \textsc{Rot}(\mathbf{B}_{11}) & \cdots & \textsc{Rot}(\mathbf{B}_{1\ell}) \\ \vdots & \ddots & \vdots \\ \textsc{Rot}(\mathbf{B}_{k1}) & \cdots & \textsc{Rot}(\mathbf{B}_{k\ell}) \end{pmatrix}.$$

We now define a module lattice. Since R is the ring of integers of a number field, it is a Dedekind domain and the notion of rank is well-defined for R-modules.

Definition 2.65. Let $M \subset \mathbb{K}^k$ be an R-module of rank $\ell \leq k$. Then, we have the map,

$$\boldsymbol{\sigma} = (\sigma, \ldots, \sigma)\colon \mathbb{K}^k \to \mathbb{R}^{nk}, \qquad (x_1, \ldots, x_k) \mapsto (\sigma(x_1), \ldots, \sigma(x_k)) \ ,$$

and we say the image $\boldsymbol{\sigma}(M)$ is a *module lattice*.

Any module lattice $\boldsymbol{\sigma}(M)$ is a rank-$n\ell$ lattice in $\mathbb{R}^{nk}$. We may refer to 'the module lattice M' to mean $\boldsymbol{\sigma}(M)$. If $\mathbf{B} \in \mathbb{K}^{k\times\ell}$ is a basis for an R-module M then $\textsc{Rot}(\mathbf{B}) \in \mathbb{Q}^{nk\times n\ell}$ is a basis for the module lattice $\boldsymbol{\sigma}(M)$. A matrix $\mathbf{B}$ is an R-basis for the module lattice R^ℓ whenever $\mathbf{B} \in \mathrm{GL}_\ell(R)$.

Definition 2.66. Given $k, \ell \in \mathbb{Z}_{\geq 1}$, the *Hermitian adjoint* of a matrix $\mathbf{B} \in \mathbb{K}^{k\times\ell}$ is denoted by $\mathbf{B}^\star$ and is obtained by applying complex conjugation coefficientwise to $\mathbf{B}^\mathsf{T}$. The Hermitian adjoint of a vector $\mathbf{f} \in \mathbb{K}^\ell$, is the row vector $\mathbf{f}^\star$, by interpreting the column vector as an $\ell \times 1$ matrix.

Hermitian form

Analogously to defining an R-basis, we may consider a structured variant of a quadratic form: the Hermitian form.

Definition 2.67. For $\ell \geq 1$, the *set of Hermitian forms* $\mathcal{H}_\ell^{>0}(\mathbb{K})$ consists of all $\mathbf{Q} \in \mathbb{K}^{\ell\times\ell}$ such that $\mathbf{Q}^\star = \mathbf{Q}$ and $\mathrm{Tr}(\mathbf{v}^\star\mathbf{Q}\mathbf{v}) > 0$ for all $\mathbf{v} \in \mathbb{K}^\ell \setminus \{\mathbf{0}\}$.

An *isomorphism* of Hermitian forms $\mathbf{Q}$ and $\mathbf{Q}'$ is a matrix $\mathbf{U} \in \mathrm{GL}_\ell(R)$ such that $\mathbf{U}^\star\mathbf{Q}\mathbf{U} = \mathbf{Q}'$ holds, and we analogously write $\mathrm{O}(\mathbf{Q})$ for the *group of automorphisms* of $\mathbf{Q}$.

Equivalently, $\mathbf{Q}$ is a Hermitian form whenever $\textsc{Rot}(\mathbf{Q})$ is a quadratic form. Given a basis $\mathbf{B} \in \mathbb{K}^{\ell \times \ell}$, its Gram matrix $\mathbf{B}^\star\mathbf{B}$ is a Hermitian form.

Note that $\textsc{Rot}$ maps $\mathrm{O}(\mathbf{Q})$ into $\mathrm{O}(\textsc{Rot}(\mathbf{Q}))$. The automorphism group of $\mathbf{I}_2(R)$ is equal to

$$\mathrm{O}(\mathbf{I}_2(R)) = \left\{ \begin{pmatrix} X^a & 0 \\ 0 & X^b \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^c \;\middle|\; 0 \le a, b < n,\ c \in \{0,1\} \right\} ,$$

and has size $8n^2$. This is very little compared to the sheer number of automorphisms of $\textsc{Vec}(R^2) = \mathbb{Z}^{2n}$, which is $2^{2n}(2n)!$ by Example 2.30. Similar to Definition 2.64, a Hermitian form induces an inner product and norm.

Definition 2.68. The *inner product with respect to a Hermitian form* $\mathbf{Q} \in \mathcal{H}_\ell^{>0}(\mathbb{K})$ is given by,

$$\langle \mathbf{f}, \mathbf{g} \rangle_{\mathbf{Q}} = \frac{1}{n} \cdot \mathrm{Tr}(\mathbf{f}^\star \mathbf{Q} \mathbf{g}) \ , \qquad \text{and} \qquad \|\mathbf{f}\|_{\mathbf{Q}} = \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle_{\mathbf{Q}}} \ ,$$

for $\mathbf{f}, \mathbf{g} \in \mathbb{K}^\ell$.

Once again, observe that for any basis $\mathbf{B}$, we have $\langle \mathbf{Bf}, \mathbf{Bg} \rangle = \langle \mathbf{f}, \mathbf{g} \rangle_{\mathbf{B}^\star\mathbf{B}}$ and $\|\mathbf{Bf}\| = \|\mathbf{f}\|_{\mathbf{B}^\star\mathbf{B}}$.

The following problem is the module analogue of ZLIP.

Problem 2.69 (smLIP)**.** Given upon input a matrix $\mathbf{Q} \in \mathrm{GL}_\ell(R)$ for which there exist $\mathbf{B} \in \mathrm{GL}_\ell(R)$ such that $\mathbf{Q} = \mathbf{B}^\star\mathbf{B}$, output any such $\mathbf{B}$.

Note that smLIP is not harder than ZLIP in dimension $n\ell$.

2.3 Duality

There are two ways to define the dual of a lattice. The first one is inherited from groups, while the second one is geometric and specific to lattices. In the context of Fourier transforms, it is useful to consider both definitions, and relate them. Note that characters are only taken for full-rank lattices.

Definition 2.70 (group-theoretic dual)**.** For a locally compact group G (e.g. a finite group or a torus $\mathbb{R}^n/\Lambda$ of a full-rank lattice Λ), the *group of characters on* G, is denoted by

$$\widehat{G} = \left\{ \chi \colon G \to \mathcal{S}^1 \;\middle|\; \chi \text{ continuous group homomorphism} \right\} .$$

Note that when G is a finite abelian group, any homomorphism $\chi\colon G \to \mathcal{S}^1$ is continuous.

Definition 2.71 (dual lattice)**.** The *dual of a lattice* $\Lambda \subset \mathbb{R}^n$ is denoted by $\Lambda^\vee$ and consists of all vectors $\mathbf{x} \in \operatorname{span}(\Lambda)$ for which $\langle \mathbf{x}, \Lambda \rangle \subseteq \mathbb{Z}$ holds.

We refer to Λ as the *primal lattice* and $\Lambda^\vee$ as the *dual lattice*.

Example 2.72. We have $(\mathbb{Z}^n)^\vee = \mathbb{Z}^n$ and $\mathcal{L}_q(\mathbf{A})^\vee = \frac{1}{q}\,\mathcal{L}_q^\perp(\mathbf{A})$. Moreover, for any lattice Λ and nonzero $\alpha \in \mathbb{R}$, $(\alpha\Lambda)^\vee = \Lambda^\vee/\alpha$.

The following lemma shows that we may interchange these two notions of duality, i.e., any dual vector defines a character and vice versa.

Lemma 2.73. *The map from* $\Lambda^\vee$ *to* $\widehat{\mathbb{R}^n/\Lambda}$ *that sends a dual vector* $\mathbf{w}$ *to the character*

$$\chi_{\mathbf{w}}\colon \mathbf{t} \mapsto \exp(2\pi i \cdot \langle \mathbf{t}, \mathbf{w} \rangle), \tag{2.9}$$

is a group isomorphism.

Proof. We prove the map $\mathbf{w} \mapsto \chi_{\mathbf{w}}$ is a well-defined group homomorphism, injective and surjective.

Well-definedness: It follows directly from the definition of $\Lambda^\vee$ that $\chi_{\mathbf{w}}$ is a character given $\mathbf{w} \in \Lambda^\vee$. It is also easy to see $\mathbf{w} \mapsto \chi_{\mathbf{w}}$ is a group homomorphism.

Injectivity: Suppose $\chi_{\mathbf{w}}(\mathbf{t}) = 1$ holds for all $\mathbf{t} \in \mathbb{R}^n$. In particular, for $\mathbf{t} = c\mathbf{w}$ with $c \in \mathbb{R}_{>0}$, we have $\chi_{\mathbf{w}}(c\mathbf{w}) = 1$, which implies $\langle \mathbf{w}, c\mathbf{w} \rangle \in \mathbb{Z}$. Assume $\mathbf{w} \neq \mathbf{0}$ so in particular $\|\mathbf{w}\| > 0$. Then, for $c < 1/\|\mathbf{w}\|^2$, we get a contradiction as $\langle \mathbf{w}, c\mathbf{w} \rangle = c\|\mathbf{w}\|^2$ must be simultaneously a (strictly) positive integer and smaller than 1. We conclude $\mathbf{w} = \mathbf{0}$.

Surjectivity: Let $\chi \in \widehat{\mathbb{R}^n/\Lambda}$ be given. By continuity, there is an open ball $U \subseteq \mathbb{R}^n$ centred at $\mathbf{0}$ such that $\chi(U + \Lambda) \subseteq \{z \in \mathcal{S}^1 | \operatorname{Re}(z) > 0\}$ holds. On this ball, we can find a linear map $\varphi\colon U \to (-\frac{1}{4}, \frac{1}{4})$ such that $\chi(\mathbf{x} + \Lambda) = \exp(2\pi i \varphi(\mathbf{x}))$ holds for all $\mathbf{x} \in U$ as there is exactly one value possible for $\varphi(\mathbf{x})$. The character χ is completely determined by its values on U, i.e., for any $\mathbf{x} \in \mathbb{R}^n$ there exists $m \in \mathbb{Z}_{\geq 1}$ such that $\mathbf{x}/m \in U$ so $\chi(\mathbf{x}) = \chi(\mathbf{x}/m)^m$. We may now extend φ to a linear function $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ that satisfies $\chi(\mathbf{x} + \Lambda) = \exp(2\pi i \varphi(\mathbf{x}))$ for all $\mathbf{x} \in \mathbb{R}^n$. Now there is a $\mathbf{w} \in \mathbb{R}^n$ such that $\varphi = \langle \cdot, \mathbf{w} \rangle$, as any linear map is of this form. Note because $\chi(\Lambda) = 1$ holds, we have $\langle \mathbf{w}, \Lambda \rangle \subset \mathbb{Z}$ so $\mathbf{w} \in \Lambda^\vee$. Thus, $\chi = \chi_{\mathbf{w}}$. □

For a sublattice $\Lambda' \subset \Lambda$, the dual of the finite group Λ/Λ' has a natural connection to the dual lattices of Λ' and Λ by the following lemma.

Lemma 2.74. *For two full-rank lattices $\Lambda_1 \subset \Lambda_2 \subset \mathbb{R}^n$, there is a canonical group isomorphism of abelian groups,*

$$\left(\widehat{\mathbb{R}^n/\Lambda_1}\right)\Big/\left(\widehat{\mathbb{R}^n/\Lambda_2}\right) \to \widehat{\Lambda_2/\Lambda_1},$$

given by restriction of a character $\chi\colon \mathbb{R}^n/\Lambda_1 \to \mathcal{S}^1$ (modulo $\widehat{\mathbb{R}^n/\Lambda_2}$) to Λ_2/Λ_1.

Proof. We prove restriction is a well-defined group homomorphism, injective and surjective.

Well-definedness: Any Λ_1-periodic character χ can be multiplied by a Λ_2-periodic character in $\widehat{\mathbb{R}^n/\Lambda_2}$ as the function stays the same on Λ_2/Λ_1.

Injectivity: Any character $\chi \in \widehat{\mathbb{R}^n/\Lambda_1}$ that is 1 on Λ_2/Λ_1 is coming from a function $\mathbb{R}^n \to \mathcal{S}^1$ that is Λ_2-periodic, i.e., a character from $\widehat{\mathbb{R}^n/\Lambda_2}$.

Surjectivity: Left hand side has size

$$\left|\widehat{\mathbb{R}^n/\Lambda_1}\Big/\widehat{\mathbb{R}^n/\Lambda_2}\right| = |\Lambda_1^\vee/\Lambda_2^\vee| = |\Lambda_2/\Lambda_1|,$$

and right hand side has size $|\widehat{\Lambda_2/\Lambda_1}| = |\Lambda_2/\Lambda_1|$, where we use that a finite abelian group G is isomorphic to its dual. Since there is an injection from one set to another set of the same finite cardinality, it must be surjective as well. □

One can compute a Λ-periodic function by evaluating its Fourier transform at dual lattice points.

Theorem 2.75 (Poisson summation formula, [SW71, Chapter VII])**.** *Given a full-rank lattice Λ and a function $f\colon \mathbb{R}^n \to \mathbb{R}$ satisfying certain decaying conditions, for all $\mathbf{t} \in \mathbb{R}^n$ one has*

$$\sum_{\mathbf{v}\in\Lambda} f(\mathbf{v}+\mathbf{t}) = \frac{1}{\det(\Lambda)} \sum_{\mathbf{w}\in\Lambda^\vee} e^{2\pi i\langle \mathbf{t},\mathbf{w}\rangle} \widehat{f}(\mathbf{w}).$$

2.3.1 Dual basis

Given a basis $\mathbf{B}$ of a primal lattice Λ, one can compute a basis for the dual lattice $\Lambda^\vee$.

Definition 2.76 (dual basis)**.** Given a basis $\mathbf{B}$ of the primal lattice Λ, the *dual basis associated to* $\mathbf{B}$ is

$$\mathbf{B}^\vee = [\mathbf{b}_1^\vee, \dots, \mathbf{b}_k^\vee] = \mathbf{B} \cdot \left(\mathbf{B}^\mathsf{T} \cdot \mathbf{B}\right)^{-1} ,$$

and ${}^\vee\mathbf{B} = [\mathbf{b}_k^\vee, \dots, \mathbf{b}_1^\vee]$ is used to denote the *reversed dual basis*, which contains the basis vectors of $\mathbf{B}^\vee$ in reversed ordering.

There are three remarks to make here.

(1) The map $\mathbf{B} \mapsto \mathbf{B}^\vee$ is an involution, i.e., $\mathbf{B} = (\mathbf{B}^\vee)^\vee$. Similarly, we have $\mathbf{B} = {}^\vee({}^\vee\mathbf{B})$;
(2) The dual basis $\mathbf{D} = [\mathbf{b}_1^\vee, \dots, \mathbf{b}_k^\vee] = \mathbf{B}^\vee$ is the unique basis with vectors in $\mathrm{span}(\mathbf{B})$ that satisfies $\left\langle \mathbf{b}_i, \mathbf{b}_j^\vee \right\rangle = [\![i = j]\!]$, or equivalently, $\mathbf{D}^\mathsf{T}\mathbf{B} = \mathbf{I}_k(\mathbb{Z})$. Hence, $\mathbf{B}^\vee = \mathbf{B}^{-\mathsf{T}}$ if $\mathbf{B}$ is full-rank.
(3) Any point $\mathbf{x} = \sum_{i=1}^k c_i \mathbf{b}_i$ with $c_i \in \mathbb{R}$ satisfies $c_i = \langle \mathbf{x}, \mathbf{b}_i^\vee \rangle$, because $\mathbf{D}^\mathsf{T}\mathbf{x} = (\mathbf{D}^\mathsf{T}\mathbf{B})\mathbf{c} = \mathbf{c}$. Hence, a point $\mathbf{x} \in \mathrm{span}(\mathbf{B})$ satisfies $\mathbf{x} \in \mathrm{span}(\mathbf{b}_2, \dots, \mathbf{b}_k)$ if and only if $\langle \mathbf{x}, \mathbf{b}_1^\vee \rangle = 0$. Generalizing this further, leads to following proposition.

Proposition 2.77 ([Mic14, Thm. 5])**.** *Let $\mathbf{B} \in \mathbb{R}^{n \times k}$ be a basis and $1 \le \ell \le r \le k$. Then, the projected sublattice $({}^\vee\mathbf{B})_{[k+1-r,k+1-\ell]}$ is the reversed dual basis associated to the projected sublattice $\mathbf{B}_{[\ell,r]}$.*

Informally, sectioning in the primal basis corresponds to projecting in the reversed dual basis.

Proof. Iteratively and using (1), we may reduce to the simplest case $\ell = 2$ and $r = k$. The above shows $({}^\vee\mathbf{B})_{[1,k-1]}$ is the reversed dual of $\mathbf{B}_{[2,k]}$. □

This provides three corollaries: (1) $\det(\Lambda^\vee) = 1/\det(\Lambda)$, (2) for all $i \in [k]$ we have $\|\mathbf{b}_i^\vee\| = 1/\|\mathbf{b}_i^*\|$, and (3) $\mathbf{b}_k^*/\|\mathbf{b}_k^*\|^2 = \mathbf{b}_k^\vee$ is a dual vector.

2.3.2 Discrete Fourier transform

For any set S let $\mathbb{C}^S$ be the group of sequences $(x_s)_{s\in S}$ having complex coefficients x_s, where the group operation is given by pointwise addition. Given a finite group G, the *discrete Fourier transform* (DFT) of a sequence $(x_g)_{g\in G} \subset \mathbb{C}$ is the $\mathbb{C}$-linear map

$$\begin{aligned} \mathrm{DFT}_G\colon \qquad & \mathbb{C}^G \to \mathbb{C}^{\widehat{G}}, \\ & (x_g)_{g\in G} \mapsto \left(\sum_{g\in G} x_g \cdot \overline{\chi(g)}\right)_{\chi\in\widehat{G}}. \end{aligned} \tag{2.10}$$

The m-dimensional *fast Fourier transform* (FFT) is an algorithm that, upon input a group G, given as $n_1, \dots, n_m \in \mathbb{Z}_{\geq 2}$ such that $G \cong \bigoplus_{j=1}^m (\mathbb{Z}/n_j\mathbb{Z})$, and $(x_g)_{g\in G}$, outputs $\mathrm{DFT}_G\Big((x_g)_{g\in G}\Big)$ in time $O(|G|\log|G|)$. There are various FFTs known for any finite group G (even when an n_i is a large prime) [Rad68, DV90]. When the group G is not cyclic, the algorithm is often referred to as a multidimensional FFT. When $G \cong (\mathbb{Z}/2\mathbb{Z})^k$, the algorithm is a *Walsh–Hadamard transform* (WHT), which is more efficient in practice. For a finite group G, the inverse of DFT_G is given by

$$\mathrm{DFT}_G^{-1}\Big((y_\chi)_{\chi\in\widehat{G}}\Big) = \frac{1}{|G|}\cdot\left(\sum_{\chi\in\widehat{G}} y_\chi\chi(g)\right)_{g\in G}.$$

Identifying an element $g \in G$ with the evaluation map $\mathrm{ev}_g\colon \chi \mapsto \chi(g)$ gives the canonical isomorphism $G \cong \widehat{\left(\widehat{G}\right)}$, so an inverse DFT is basically a DFT up to some reordering.

2.4 Discrete Gaussian sampling

Given a lattice Λ and parameter $\sigma \in \mathbb{R}_{>0}$, we may consider a discrete analogue of the continuous Gaussian.

Definition 2.78. For any lattice $\Lambda \subset \mathbb{R}^n$ and vector $\mathbf{c} \in \mathbb{R}^n$, we denote the *discrete Gaussian distribution* on $\Lambda + \mathbf{c}$ with parameter σ by $\mathcal{D}_{\Lambda+\mathbf{c},\sigma}$, which assigns a probability

$$\frac{1}{\sum_{y\in\Lambda+\mathbf{c}} \rho_\sigma(\mathbf{y})} \cdot \rho_\sigma(\mathbf{x}),$$

to a point $\mathbf{x} \in \Lambda + \mathbf{c}$, and probability zero to all points in $\mathbb{R}^n \setminus (\Lambda + \mathbf{c})$. The discrete Gaussian distribution $\mathcal{D}_{\Lambda,\sigma}$ is shorthand for $\mathcal{D}_{\Lambda+\mathbf{0},\sigma}$.

Note that the discrete Gaussian is usually defined [MR07] with respect to a width $s = \sqrt{2\pi}\sigma$, while we mimic Falcon in this regard [PFH+22]. Moreover, σ is generally not the standard deviation of a discrete Gaussian, though it will be close for large enough σ, see Section 2.4.1. A discrete Gaussian has a similar tail bound to a continuous Gaussian.

Lemma 2.79 ([Ban93, Lem. 1.5(ii)])**.** *For any lattice $\Lambda \subset \mathbb{R}^n$, point $\mathbf{c} \in \mathbb{R}^n$ and $\tau \geq 1$, we have*

$$\Pr_{\mathbf{x} \sim \mathcal{D}_{\Lambda+\mathbf{c},\sigma}} [\|\mathbf{x}\| > \tau\sigma\sqrt{n}] \leq 2\frac{\rho_\sigma(\Lambda)}{\rho_\sigma(\Lambda + \mathbf{c})} \cdot \tau^n e^{-\frac{n}{2}(\tau^2-1)}.$$

We may also define a discrete Gaussian distribution with respect to a quadratic form instead of a lattice.

Definition 2.80. *Gaussian mass with respect to* $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$ is given by

$$\rho_{\mathbf{Q},\sigma} \colon \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \mapsto \exp\left(-\|\mathbf{x}\|_{\mathbf{Q}}^2/2\sigma^2\right) .$$

For any $\mathbf{c} \in \mathbb{R}^n$, the *discrete Gaussian distribution on* $\mathbb{Z}^n + \mathbf{c}$ *with respect to* $\mathbf{Q}$ *and parameter* σ is denoted by $\mathcal{D}_{\mathbf{Q},\mathbb{Z}^n+\mathbf{c},\sigma}$, which assigns a probability

$$\frac{1}{\sum_{\mathbf{y} \in \mathbb{Z}^n + \mathbf{c}} \rho_{\mathbf{Q},\sigma}(\mathbf{y})} \rho_{\mathbf{Q},\sigma}(\mathbf{x}) \quad ,$$

to a point $\mathbf{x} \in \mathbb{Z}^n + \mathbf{c}$, and zero otherwise. The distribution $\mathcal{D}_{\mathbf{Q},\sigma}$ is shorthand for $\mathcal{D}_{\mathbf{Q},\mathbb{Z}^n+\mathbf{0},\sigma}$.

If $\mathbf{Q} = \mathbf{B}^\mathsf{T} \cdot \mathbf{B}$, note we have

$$\mathbf{B} \cdot \mathcal{D}_{\mathbf{Q},\mathbb{Z}^n+\mathbf{c},\sigma}(\mathbf{x}) = \mathcal{D}_{\Lambda(\mathbf{B})+\mathbf{B}\cdot\mathbf{c},\sigma}(\mathbf{B} \cdot \mathbf{x}) \quad ,$$

as $\rho_{\mathbf{Q},\sigma}(\mathbf{x}) = \rho_\sigma(\mathbf{B} \cdot \mathbf{x})$.

When σ is large enough compared to the maximum length of a Gram–Schmidt basis vector, denoted by $\|\mathbf{B}_{\mathbf{Q}}^*\|$, we can efficiently sample a discrete Gaussian.

Lemma 2.81 ([BLP+13, Lem. 2.3], adapted)**.** *There is a probabilistic polynomial-time algorithm that on input a quadratic form $\mathbf{Q} \in \mathcal{S}_n^{>0}(\mathbb{R})$, $\mathbf{c} \in \mathbb{R}^n$ and parameter $\sigma \geq \|\mathbf{B}_{\mathbf{Q}}^*\| \cdot (1/\pi) \cdot \sqrt{\log(2n+4)/2}$ outputs a sample according to $\mathcal{D}_{\mathbf{Q},\mathbb{Z}^n+\mathbf{c},\sigma}$.*

Additionally, we extend the discrete Gaussian to Hermitian forms.

Definition 2.82. Given a Hermitian form $\mathbf{Q} \in \mathcal{H}_\ell^{>0}(\mathbb{K})$, the discrete Gaussian distribution with respect to $\mathbf{Q}$ is given by

$$\mathcal{D}_{\mathbf{Q},\sigma}(\mathbf{x}) = \mathcal{D}_{\mathrm{Rot}(\mathbf{Q}),\sigma}(\mathrm{Vec}(\mathbf{x})) \ ,$$

for any $\mathbf{x} \in \mathbb{K}^\ell$.

Due to our choice of $\mathbb{K}$ and the definition of our $\mathbf{Q}$-norm, this is equivalent to the natural definition that follows from $\rho_{\mathbf{Q},\sigma}$ when forgetting the module structure.

2.4.1 Smoothing parameter

Definition 2.83. For $\varepsilon > 0$ we define the *smoothing parameter* $\eta_\varepsilon(\Lambda)$ as the smallest $\sigma \in \mathbb{R}_{>0}$ satisfying $\rho_{1/(2\pi\sigma)}(\Lambda^\vee \setminus \{\mathbf{0}\}) \leq \varepsilon$.

Note that η_ε is usually defined with respect to a width $s = \sqrt{2\pi}\sigma$. Here its value is a factor $\sqrt{2\pi}$ smaller than usual [MR07, GPV08]. If $\sigma \geq \eta_\varepsilon(\Lambda)$ then $\mathcal{D}_{\Lambda+\mathbf{c},\sigma}$ exhibits several useful properties. For example, σ is close to the standard deviation of $\mathcal{D}_{\Lambda+\mathbf{c},\sigma}$, with the closeness parametrised by ε, see [MR07, Lem. 4.3], and cosets have similar weights. We may say σ is 'above smoothing' to refer to $\sigma \geq \eta_\varepsilon(\Lambda)$ for some implicit appropriate ε.

Lemma 2.84 ([MR07, Proof of Lem. 4.4])**.** *For any lattice $\Lambda \subset \mathbb{R}^n$, point $\mathbf{c} \in \mathbb{R}^n$, and $\varepsilon \in (0,1)$, $\sigma \geq \eta_\varepsilon(\Lambda)$, we have*

$$(1-\varepsilon) \cdot \frac{\left(\sigma\sqrt{2\pi}\right)^n}{\det(\Lambda)} \leq \rho_\sigma(\Lambda + \mathbf{c}) \leq (1+\varepsilon) \cdot \frac{\left(\sigma\sqrt{2\pi}\right)^n}{\det(\Lambda)},$$

To use above lemma for quadratic forms, note that for any basis $\mathbf{B}$ of Λ, we have $\rho_\sigma(\Lambda + \mathbf{c}) = \rho_{\mathbf{B}^\intercal\mathbf{B},\sigma}(\mathbb{Z}^n + \mathbf{B}^{-1}\mathbf{c})$.

2.5 Lattice basis reduction

There are an infinite number of bases for one lattice. It is hard to find a basis with short basis vectors, and this process is called lattice basis reduction. On the other hand, given a basis $\mathbf{B}$, one can easily

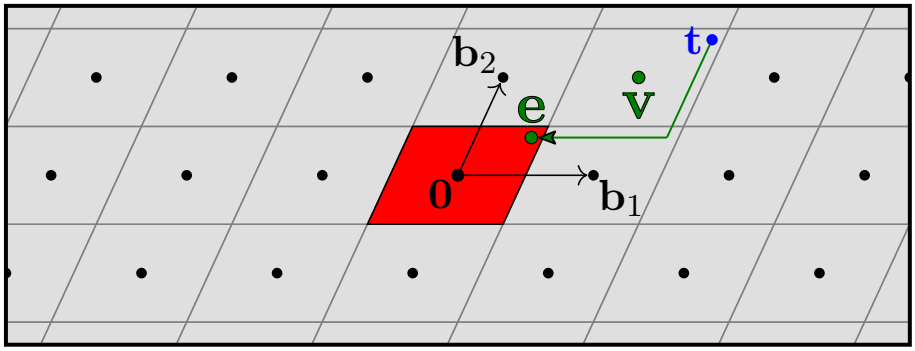


Figure 2.3: Fundamental domain of $\mathbf{B}$

obtain the basis $\mathbf{BU}$ with (generally) longer vectors, by generating some $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ arbitrarily.

We discuss reductions of a vector with respect to a basis, a basis itself and a rank-2 lattices. Then we discuss a polynomial-time basis reduction algorithm [LLL82], and an exponential-time algorithm that provides stronger basis reduction [SE94].

2.5.1 Babai's reduction algorithms

Here, we will look at the Rounding-Off algorithm and Babai's Nearest-Plane algorithm [Bab86], which both solve an approximation variant of CVP.

Definition 2.85 (approx-CVP$_\gamma$)**.** Given an approximation factor $\gamma \geq 1$, a lattice Λ and a vector $\mathbf{t} \in \mathbb{R}^n$, find a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \|\mathbf{w} - \mathbf{t}\|$ holds for all $\mathbf{w} \in \Lambda$.

Intuitively, this definition asks given a target $\mathbf{t}$ having lattice point $\mathbf{v}_0$ closest by, to find any nearby lattice point that is at most a factor γ further away from $\mathbf{t}$.

Rounding-Off

Given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ and a vector $\mathbf{t} = \sum_{i=1}^{k} c_i \mathbf{b}_i$ with $c_i \in \mathbb{R}$, rounding each c_i to the nearest integer, i.e., $n_i = \lceil c_i \rfloor$, provides a lattice point $\mathbf{v} = \sum_{i=1}^{k} n_i \mathbf{b}_i \in \Lambda$ that is somewhat near to $\mathbf{t}$.

Algorithm 2.2. RO$(\mathbf{B}, \mathbf{t})$: Rounding-Off algorithm

Input: basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ and target $\mathbf{t} \in \mathbb{R}^n$
Output: $(\mathbf{v}, \mathbf{e}) \in \Lambda \times \mathbb{R}^n$ such that $\mathbf{t} = \mathbf{v} + \mathbf{e}$ and $\pi_{\mathrm{span}(\mathbf{B})}(\mathbf{e}) \in \mathcal{P}(\mathbf{B})$

1: $\mathbf{c} \leftarrow \left\lceil (\mathbf{B}^\mathsf{T}\mathbf{B})^{-1} \cdot \mathbf{B}^\mathsf{T} \cdot \mathbf{t} \right\rfloor$ ▷ $\mathbf{c} \in \mathbb{Z}^k$
2: $\mathbf{v} \leftarrow \mathbf{B}\mathbf{c}$
3: **return** $(\mathbf{v}, \mathbf{t} - \mathbf{v})$

We will refer to this procedure, which can be found in Algorithm 2.2, the *Rounding-Off algorithm (RO)*. Note, this procedure may be considered "straightforward" [Bab86].

Remark 2.86. We assume that the rounding function,

$$\lceil \cdot \rfloor : \mathbb{R} \to \mathbb{Z}, x \mapsto \lceil x \rfloor \ ,$$

satisfies $x - \lceil x \rfloor \in (-1/2, 1/2]$ for all $x \in \mathbb{R}$, or equivalently, given $n \in \mathbb{Z}$, we have $\lceil x \rfloor = n$ whenever $x \in (n - \frac{1}{2}, n + \frac{1}{2}]$. Programming languages, however, round differently. For example, C's `lrint` function, and Python round a floating-point number $x \in \mathbb{Z} + \frac{1}{2}$ towards an even integer, whereas C's `round` function rounds away from zero.

Definition 2.87 (fundamental domain)**.** Given a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$, its *fundamental domain* is given by

$$\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{x} \in \mathrm{span}(\mathbf{B}) \;\middle|\; \exists \mathbf{y} \in \left(-\frac{1}{2}, \frac{1}{2}\right]^k : \mathbf{x} = \mathbf{B} \cdot \mathbf{y} \right\} \ .$$

The output vector $\mathbf{e}$ of the pair $(\mathbf{v}, \mathbf{e})$ lies in the fundamental domain of $\mathbf{B}$, which can be seen in Figure 2.3.

Nearest-Plane

Instead of determining all $c_1, \ldots, c_k$ in one go as done in Algorithm 2.2, we may as well choose to reduce $\mathbf{t}$ using an integer multiple of $\mathbf{b}_n$ that minimises $\langle \mathbf{t} - x\mathbf{b}_n, \mathbf{b}_n^* \rangle$, because any subsequent additions of $\mathbf{b}_{n-1}, \ldots, \mathbf{b}_1$ keep the inner product with $\mathbf{b}_n^*$ the same by definition of Gram–Schmidt orthogonalisation.

We will call this procedure, which can be found in Algorithm 2.3, *Babai's Nearest-Plane algorithm (NP)* [Bab86], although it was known before Babai's work [LLL82, Len83].

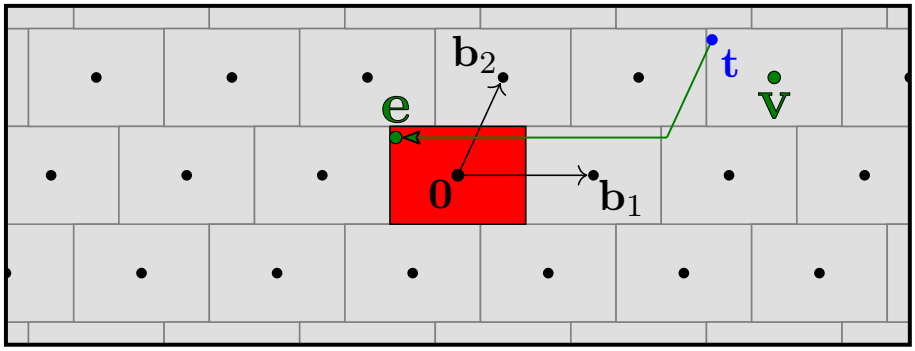


Figure 2.4: Babai's domain of $\mathbf{B}$

Algorithm 2.3. NP$(\mathbf{B}, \mathbf{t})$: Babai's Nearest-Plane algorithm

Input: basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ and target $\mathbf{t} \in \mathbb{R}^n$
Output: $(\mathbf{v}, \mathbf{e}) \in \Lambda \times \mathbb{R}^n$ such that $\mathbf{t} = \mathbf{v} + \mathbf{e}$ and $\pi_{\mathrm{span}(\mathbf{B})}(\mathbf{e}) \in \mathcal{P}(\mathbf{B}^*)$

$(\mathbf{v}, \mathbf{e}) \leftarrow (\mathbf{0}, \mathbf{t})$
for $i = k, k-1, \ldots, 1$ **do** ▷ Invariant: $\mathbf{t} = \mathbf{v} + \mathbf{e}$
  $c_i \leftarrow \lceil \langle \mathbf{b}_i^*, \mathbf{e} \rangle / \|\mathbf{b}_i^*\|^2 \rfloor$
  $\mathbf{v} \leftarrow \mathbf{v} + c_i \mathbf{b}_i$
  $\mathbf{e} \leftarrow \mathbf{e} - c_i \mathbf{b}_i$
return $(\mathbf{v}, \mathbf{e})$

Definition 2.88 (Babai's domain)**.** *Babai's domain* with respect to a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ is given by $\mathcal{P}(\mathbf{B}^*)$.

The output vector $\mathbf{e}$ of the pair $(\mathbf{v}, \mathbf{e})$ lies in Babai's domain, which can be seen in Figure 2.4.

Proposition 2.89. *Given a basis* $\mathbf{B} \in \mathbb{R}^{n \times k}$ *we have*

$$\max \left\{ \|\mathbf{e}\|^2 \;\middle|\; \mathbf{e} \in \mathcal{P}(\mathbf{B}^*) \right\} = \frac{1}{4} \sum_{i=1}^{n} \|\mathbf{b}_i^*\|^2 \ .$$

On the other hand, there is no such simple bound on the norm of elements in $\mathcal{P}(\mathbf{B})$. Namely, for every $L \in \mathbb{R}$ and unit-volume lattice in dimension 2 and higher, there is a basis such that $\mathcal{P}(\mathbf{B})$ has a vector of norm $\geq L$.

2.5.2 Size reduction

Here we discuss the reduction of a basis that makes the basis vectors nearly orthogonal using Rounding-Off or Babai's Nearest-Plane. We may reduce the length of a basis vector $\mathbf{b}_i$ using Babai's Nearest-Plane algorithm with respect to $[\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}]$. This reduces the size of the μ_{ij} coefficients and effectively makes pairs of basis vectors "nearly orthogonal", while $\mathbf{B}^*$ is kept the same.

Definition 2.90 (size-reduced)**.** A basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ is *size-reduced* if for all $1 \leq i < j \leq k$ we have $|\mu_{ij}| \leq \frac{1}{2}$.

Algorithm 2.4 transforms the basis $\mathbf{B}$ into a size-reduced basis. Moreover, the implicit basis transformation matrix $\mathbf{U}$ is unitriangular. Thus, a new $\mathbf{b}'_i$ is the sum of $\mathbf{b}_i$ and a $\mathbb{Z}$-linear combination of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

Algorithm 2.4. SizeReduce($\mathbf{B}$): size reduction of a basis

Input: basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{R}^{n \times k}$
Output: size-reduced basis $\mathbf{B}' = \mathbf{B}\mathbf{U}$ for some $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$

1: $\mathbf{b}'_1 \leftarrow \mathbf{b}_1$
2: **for** $i = 2, 3, \ldots, k$ **do**
3: $\quad (\_, \mathbf{b}'_i) \leftarrow \mathrm{NP}([\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}], \mathbf{b}_i)$
4: **return** $\mathbf{B}' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_k]$

Proposition 2.91. *Given a size-reduced basis* $\mathbf{B} \in \mathbb{R}^{n \times k}$*, for all* $\mathbf{e} \in \mathcal{P}(\mathbf{B})$ *we have*

$$\|\mathbf{e}\|^2 \leq \frac{k^2}{4} \sum_{i=1}^{k} \|\mathbf{b}_i^*\|^2 \ .$$

Proof. Let us write $\mathbf{b}_j = \mathbf{b}_j^* + \sum_{i<j} \mu_{ij} \mathbf{b}_i^*$ with $|\mu_{ji}| \leq \frac{1}{2}$ due to size-reduction, and $\mathbf{e} = \sum_{i=1}^{k} c_i \mathbf{b}_i$ with $c_i \in (-1/2, 1/2]$ by definition of $\mathcal{P}(\mathbf{B})$. Then,

$$\|\mathbf{e}\|^2 = \sum_{i=1}^{k} \|\mathbf{b}_i^*\|^2 \cdot \left(c_i + \sum_{j=i+1}^{k} c_j \mu_{ij} \right)^2 \ .$$

By the triangle inequality, for all $i \in [k]$ we have

$$\left| c_i + \sum_{j=i+1}^{k} c_j \mu_{ij} \right| \leq \frac{1}{2} + \frac{k-i}{4} \leq \frac{k}{2} \ ,$$

and the claim follows. □

Comparing Proposition 2.89 and Proposition 2.91, Nearest-Plane reduces vectors to a norm that may be a factor k smaller than that of Rounding-Off.

Proposition 2.92. *A size-reduced basis* $\mathbf{B} \in \mathbb{R}^{n\times k}$ *satisfies*

$$\forall i \in [k]\colon \qquad \|\mathbf{b}_i^*\|^2 \le \|\mathbf{b}_i\|^2 \le \|\mathbf{b}_i^*\|^2 + \frac{1}{4}\sum_{j=1}^{i-1}\|\mathbf{b}_j^*\|^2 \ .$$

Proof. The lower bound on $\|\mathbf{b}_i\|$ follows from Equation (2.7), and the upper bound follows from Proposition 2.89 using basis $\mathbf{B}_{[1,i-1]}$ and target $\mathbf{e} = \mathbf{b}_i - \mathbf{b}_i^*$. □

Thus, we need to reduce the Gram–Schmidt norms to obtain a basis with short basis vectors.

The condition number of a basis is a way to measure tolerance of numerical errors in algorithms utilising floating-points. Moreover, it enables one to bound the unimodular transformation to make a basis size-reduced [RH23, App. B.2].

Definition 2.93 (matrix norms)**.** Let $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_k] \in \mathbb{R}^{k\times k}$ be a full-rank matrix. Its *ℓ^∞-norm and Frobenius norm* are given by, respectively,

$$\|\mathbf{A}\|_\infty = \max_{i,j} |\mathbf{A}_{ij}|, \quad \text{and} \quad \|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{k}\|\mathbf{a}_i\|^2} = \sqrt{\sum_{i,j=1}^{k}\mathbf{A}_{ij}^2} \ .$$

Its *condition number* is denoted by $\kappa(\mathbf{A}) = \|\mathbf{A}\|_{\mathrm{F}} \cdot \|\mathbf{A}^{-1}\|_{\mathrm{F}}$. We say $\mathbf{A}$ is *well-conditioned* if $\kappa(\mathbf{A})$ is small, and ill-conditioned if it is large.

Note $\|\mathbf{A}\|_\infty \le \|\mathbf{A}\|_{\mathrm{F}} \le k \cdot \|\mathbf{A}\|_\infty$ holds for any matrix $\mathbf{A}$ so these norms are comparable in size. As the ℓ^2-norm of a vector is invariant under orthogonal transformations, the Frobenius norm is invariant as well, i.e., for all $\mathbf{O} \in \mathrm{O}_k(\mathbb{R})$ we have $\|\mathbf{OA}\|_{\mathrm{F}} = \|\mathbf{A}\|_{\mathrm{F}}$. Therefore, given a QR decomposition of a full-rank basis $\mathbf{B} = \mathbf{QR}$, we have $\kappa(\mathbf{B}) = \kappa(\mathbf{R})$.

Remark 2.94. Although a size-reduced basis has a small Frobenius norm, its condition number may be exponentially large in k. Namely, given

a size-reduced unitriangular $\mathbf{R} \in \mathbb{R}^{k\times k}$, its inverse has entries satisfying $|(\mathbf{R}^{-1})_{ij}| \leq \frac{1}{2}(3/2)^{k-2}$. This follows from an upper bound on the adjugant matrix. For example, for $k = 4$ we have

2

$$\left(\mathbf{R}^{-1}\right)_{14} = (-1)\cdot \det \begin{pmatrix} \mathbf{R}_{12} & \mathbf{R}_{13} & \mathbf{R}_{14} \\ 1 & \mathbf{R}_{23} & \mathbf{R}_{24} \\ 0 & 1 & \mathbf{R}_{34} \end{pmatrix} .$$

Developing this determinant and taking absolute values yields

$$\left|\left(\mathbf{R}^{-1}\right)_{14}\right| \leq |\mathbf{R}_{12}| \cdot \left|\det \begin{pmatrix} \mathbf{R}_{23} & \mathbf{R}_{24} \\ 1 & \mathbf{R}_{34} \end{pmatrix}\right| + 1 \cdot \left|\det \begin{pmatrix} \mathbf{R}_{13} & \mathbf{R}_{14} \\ 1 & \mathbf{R}_{34} \end{pmatrix}\right| .$$

The first determinant satisfies

$$\left|\det \begin{pmatrix} \mathbf{R}_{23} & \mathbf{R}_{24} \\ 1 & \mathbf{R}_{34} \end{pmatrix}\right| \leq |\mathbf{R}_{23}\mathbf{R}_{34}| + |\mathbf{R}_{24}| \leq \frac{1}{4} + \frac{1}{2} = \frac{3}{4} ,$$

and the second determinant has the same bound. Hence,

$$\left|\left(\mathbf{R}^{-1}\right)_{14}\right| \leq |\mathbf{R}_{12}| \cdot \frac{3}{4} + \frac{3}{4} \leq \left(\frac{1}{2} + 1\right) \cdot \frac{3}{4} = \frac{1}{2}\left(\frac{3}{2}\right)^2 .$$

The upper bound is attained for the following unitriangular matrix:

$$\mathbf{R} = \begin{pmatrix} 1 & -\frac{1}{2} & \cdots & -\frac{1}{2} \\ & 1 & \ddots & \vdots \\ & & \ddots & -\frac{1}{2} \\ & & & 1 \end{pmatrix} .$$

Its inverse is given by the unitriangular matrix having $(\mathbf{R}^{-1})_{ij} = \frac{1}{2}(3/2)^{j-i-1}$ for $1 \leq i < j \leq k$.

Seysen's reduction

Seysen investigated a way to simultaneously reduce the size of the basis and of its dual [Sey93]. Specifically, in a proof [Sey93, Prop. 5], a variant of size-reduction is defined, and an algorithm is provided to reduce any

basis into such one. In this section, we will explain this algorithm and show how to phrase Seysen's reduction method [Sey93, Prop. 5] as a variant of size-reduction using block matrices and Rounding-Off.

Recently, Seysen's reduction has regained notable attention [KEF21, RH23, DPS25] for use inside lattice reduction. First, we will give the definition and the algorithm that transform a basis to a Seysen-reduced basis. Then, we will show that Seysen-reduction makes a basis more so-called 'well-conditioned' than size-reduction, This allows basis reduction to succeed using lower precision floating-point numbers.

Definition 2.95 (Seysen-reduced)**.** Let $\mathbf{B} \in \mathbb{R}^{n \times k}$ be a basis. We say $\mathbf{B}$ is *Seysen-reduced* if either $k = 1$, or parsing its R-factor as $\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}$ with $\mathbf{R}_{11} \in \mathbb{R}^{\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor}$, satisfies these three properties: (1) $\mathbf{R}_{11}$ is Seysen-reduced; (2) $\mathbf{R}_{22}$ is Seysen-reduced, and (3) $\mathbf{R}_{12} \subset \mathcal{P}(\mathbf{R}_{11})$, i.e., each column of $\mathbf{R}_{12}$ is in $\mathcal{P}(\mathbf{R}_{11})$.

In addition, a basis $\mathbf{B}$ is *dual-Seysen-reduced* if either $k = 1$, or its R-factor $\mathbf{R}$, now parsed with $\mathbf{R}_{11} \in \mathbb{R}^{\lceil k/2 \rceil \times \lceil k/2 \rceil}$, i.e., larger for odd k, satisfies: (1) $\mathbf{R}_{11}$ is dual-Seysen-reduced; (2) $\mathbf{R}_{22}$ is dual-Seysen-reduced, and (3) $\mathbf{R}_{12} \subset -\mathcal{P}(\mathbf{R}_{11})$.

The reader should be warned that our definition of Seysen-reduction originates from the proof of Seysen [Sey93, Prop. 5], and is not related to what Seysen calls S-reduced nor S_2-reduced [Sey93, Sec. 5].

To provide a simpler description of our Seysen-reduction algorithm, let us extend Algorithm 2.2 to multiple targets, i.e., given $\ell \in \mathbb{Z}_{\geq 1}$ we write

$$([\mathbf{v}_1, \ldots, \mathbf{v}_\ell], [\mathbf{e}_1, \ldots, \mathbf{e}_\ell]) \leftarrow \mathrm{RO}(\mathbf{B},\ [\mathbf{t}_1, \ldots, \mathbf{t}_\ell])\ ,$$

where $(\mathbf{v}_i, \mathbf{e}_i) \leftarrow \mathrm{RO}(\mathbf{B},\ \mathbf{t}_i)$ holds for $i \in [\ell]$.

Algorithm 2.5 shows how to reduce an upper triangular matrix $\mathbf{R}$. The general case can be reduced to this case using QR decomposition.

If Line 6 of Algorithm 2.5 would call Babai's Nearest-Plane algorithm instead of Rounding-Off, then the output basis would be size-reduced, as $\mathbf{R}_{12} \subset \mathcal{P}(\mathbf{R}_{11}^*)$ implies $|\mu_{ij}| \leq \frac{1}{2}$ for all $i \leq k/2 < j \leq k$.

If a basis is Seysen-reduced, then its dual basis is dual-Seysen-reduced.

Lemma 2.96. *If a basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ is Seysen-reduced, then the reversed dual basis ${}^{\vee}\mathbf{R}$ is dual-Seysen-reduced.*

Algorithm 2.5. SeysenReduce($\mathbf{R}$): Seysen-reduction of a basis

Input: upper triangular basis $\mathbf{R} \in \mathbb{R}^{k\times k}$ of Λ
Output: Seysen-reduced basis $\mathbf{R}' = \mathbf{RU}$ of Λ and $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$

1: **if** $\mathbf{R} \in \mathbb{R}^{1\times 1}$ **then**
2: **return** $\left(\mathbf{R}, \begin{bmatrix}1\end{bmatrix}\right)$
3: parse $\begin{bmatrix}\mathbf{R}_{11} & \mathbf{R}_{12}\\ \mathbf{0} & \mathbf{R}_{22}\end{bmatrix} = \mathbf{R}$ ▷ Blocks have half the size
4: $(\mathbf{R}'_{11}, \mathbf{U}_{11}) \leftarrow$ SeysenReduce($\mathbf{R}_{11}$)
5: $(\mathbf{R}'_{22}, \mathbf{U}_{22}) \leftarrow$ SeysenReduce($\mathbf{R}_{22}$)
6: $(\mathbf{V}, \mathbf{R}'_{12}) \leftarrow \mathrm{RO}(\mathbf{R}'_{11},\ \mathbf{R}_{12}\cdot\mathbf{U}_{22})$
7: $\mathbf{U}_{12} \leftarrow -\mathbf{R}_{11}^{-1}\cdot\mathbf{V}$ ▷ $\mathbf{R}'_{12} = \mathbf{R}_{12}\mathbf{U}_{22} - \mathbf{V}$
8: **return** $\left(\begin{bmatrix}\mathbf{R}'_{11} & \mathbf{R}'_{12}\\ \mathbf{0} & \mathbf{R}'_{22}\end{bmatrix}, \begin{bmatrix}\mathbf{U}_{11} & \mathbf{U}_{12}\\ \mathbf{0} & \mathbf{U}_{22}\end{bmatrix}\right)$

Proof. Without loss of generality, we will prove this lemma on an upper triangular R-factor $\mathbf{R} \in \mathbb{R}^{k\times k}$, and use induction on k. The case $k = 1$ is trivially satisfied.

Let us consider $k > 1$, an upper triangular basis $\mathbf{R} = \begin{pmatrix}\mathbf{R}_{11} & \mathbf{R}_{12}\\ \mathbf{0} & \mathbf{R}_{22}\end{pmatrix}$ and its lower triangular associated dual basis,

$$\mathbf{R}^\vee = \begin{pmatrix}\mathbf{S}_{11} & \mathbf{0}\\ \mathbf{S}_{21} & \mathbf{S}_{22}\end{pmatrix} = \mathbf{R}^{-\mathsf{T}} = \begin{pmatrix}\mathbf{R}_{11}^{-\mathsf{T}} & \mathbf{0}\\ -\mathbf{R}_{22}^{-\mathsf{T}}\mathbf{R}_{12}^{\mathsf{T}}\mathbf{R}_{11}^{-\mathsf{T}} & \mathbf{R}_{22}^{-\mathsf{T}}\end{pmatrix},$$

which is easily checked to be inverse transpose of $\mathbf{R}$. It follows by induction that the reversed submatrices ${}^\vee\mathbf{R}_{11}$ and ${}^\vee\mathbf{R}_{22}$ are Seysen-reduced. Because $\left((\mathbf{R}^\vee)_{k+1-i,k+1-j}\right)_{i,j=1}^{k}$ is the R-factor of ${}^\vee\mathbf{R}$, what remains to show is $\mathbf{S}_{21} \subset -\mathcal{P}(\mathbf{S}_{22})$. Since $\mathbf{R}$ is Seysen-reduced we know $\mathbf{R}_{12} \subset \mathcal{P}(\mathbf{R}_{11})$, or equivalently $\mathbf{R}_{11}^{-\mathsf{T}}\mathbf{R}_{12} \in (-1/2, 1/2]^{k\times k}$. By transposing and negating, we get $-\mathbf{S}_{22}^{-\mathsf{T}}\mathbf{S}_{21} = \mathbf{R}_{12}^{\mathsf{T}}\mathbf{R}_{11}^{-\mathsf{T}} \in (-1/2, 1/2]^{k\times k}$, which proves $\mathbf{S}_{21} \subset -\mathcal{P}(\mathbf{S}_{22})$. □

Lemma 2.97. *A Seysen-reduced upper triangular basis $\mathbf{R} \in \mathbb{R}^{k\times k}$ satisfies*

$$\lg(\|\mathbf{R}\|_\infty) \le \max\left(0, \frac{\lg(k/2)\lg(k/4)}{2}\right) + \max_{i=1,\dots,k}\lg(|\mathbf{R}_{ii}|)\ .$$

Proof. We follow Seysen's original proof using recursion [Sey93, Prop. 5]. By scaling $\mathbf{R}$, we assume without loss of generality that $\mathbf{R}$ is unitriangular.

For $k \leq 7$, one may check $\|\mathbf{R}\|_\infty \leq 1$ holds by close inspection. For example, for $k = 5$ the first row of $\mathbf{R}$ is given by $(1, r_2, r_3, r_4, r_5)$, with $|r_2| \leq \frac{1}{2} \cdot 1$, and $r_i \in \mathcal{P}([1, r_2])$ implies $|r_i| \leq \frac{1}{2}(1 + \frac{1}{2}) = 3/4$ for $i = 3, 4, 5$.

Now consider $\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{pmatrix}$ with $k \geq 8$. By induction, we have

$$\lg(\|\mathbf{R}_{11}\|_\infty) \leq \frac{\lg(\lfloor k/2 \rfloor/2)\lg(\lfloor k/2 \rfloor/4)}{2} \leq \frac{\lg(k/4)\lg(k/8)}{2} \quad , \quad \text{and}$$
$$\lg(\|\mathbf{R}_{22}\|_\infty) \leq \frac{\lg(\lceil k/2 \rceil/2)\lg(\lceil k/2 \rceil/4)}{2} \leq \frac{\lg(k/2)\lg(k/4)}{2} \quad .$$

We get $\|\mathbf{R}_{12}\|_\infty \leq \frac{1}{2}\lfloor k/2 \rfloor \cdot \|\mathbf{R}_{11}\|_\infty$ as each column of $\mathbf{R}_{12}$ is equal to a sum $\sum_{i=1}^{\lfloor k/2 \rfloor} c_i \cdot (\mathbf{R}_{11})_i$ with some $c_i \in (-1/2, 1/2]$. Thus,

$$\lg(\|\mathbf{R}_{12}\|_\infty) \leq \lg(k/4) + \frac{\lg(k/4)\lg(k/8)}{2} = \frac{\lg(k/2)\lg(k/4)}{2} \quad . \qquad \square$$

A dual-Seysen-reduced basis has slightly larger entries than a Seysen-reduced basis. Still, both grow as $\exp(\mathcal{O}(\log^2(k)))$ as $k \to \infty$.

Lemma 2.98. *A dual-Seysen-reduced upper triangular basis $\mathbf{R} \in \mathbb{R}^{k \times k}$ satisfies*

$$\lg(\|\mathbf{R}\|_\infty) \leq \frac{\lg(2k)\lg(k)}{2} + \max_{i=1,\ldots,k} \lg(|\mathbf{R}_{ii}|) \quad .$$

Proof. The proof proceeds similarly. For an R-factor $\mathbf{R} \in \mathbb{R}^{k \times k}$ with $k \geq 2$, using $\lceil k/2 \rceil \leq k$ we may bound

$$\lg(\|\mathbf{R}_{12}\|) \leq \lg\left(\frac{\lceil k/2 \rceil}{2}\right) + \lg(\|\mathbf{R}_{11}\|_\infty) \quad .$$

Denoting the maximum bound for a $k \times k$ R-factor by a_k, writing out the recursion

$$\begin{aligned} \lg(a_k) &\leq \lg\left(\frac{\lceil k/2 \rceil}{2}\right) + \lg(a_{\lceil k/2 \rceil}) \\ &\leq \lg\left(\frac{\lceil k/2 \rceil}{2}\right) + \lg\left(\frac{\lceil k/4 \rceil}{2}\right) + \lg(a_{\lceil k/4 \rceil}) \\ &\leq \cdots \leq \lg(a_2) + \sum_{i=0}^{\lceil \lg(k) - 2 \rceil} \lg\left(\frac{\lceil k/2^{i+1} \rceil}{2}\right) \quad , \end{aligned}$$

where we collected contributions of $\lg(\lceil \ell/2 \rceil/2)$ into a sum from $\ell = k, \lceil k/2 \rceil$, and so on up to the last term with $\ell = \lceil k/2^{\lceil \lg(k)-2 \rceil} \rceil$ ($\in \{3,4\}$). As $a_2 = 1$, we have

$$\begin{aligned}\lg(a_k) &\leq \sum_{i=1}^{\lceil \lg(k)-1 \rceil} \lg\left(\frac{\lceil k/2^i \rceil}{2}\right) \leq \sum_{i=1}^{\lceil \lg(k)-1 \rceil} \lg\left(\frac{k}{2^i}\right) \\ &\leq \lg^2(k) - (1+2+\cdots+\lfloor \lg(k) \rfloor) \leq \frac{\lceil \lg(k)-1 \rceil \cdot \lceil \lg(k) \rceil}{2} \\ &\leq \frac{\lg(k)(2\lg(k) - \lceil \lg(k) \rceil)}{2} \leq \frac{\lg(2k)\lg(k)}{2} \quad . \end{aligned}$$

□

Lemmata 2.96, 2.97 and 2.98 now lead to the following conclusion.

Corollary 2.99. *Given a Seysen-reduced basis $\mathbf{R} \in \mathbb{R}^{k \times k}$, its condition number is bounded by*

$$\kappa(\mathbf{R}) \leq k^2 2^{\ln^2(k)} \cdot \frac{\max_{i \in [k]} |\mathbf{R}_{ii}|}{\min_{i \in [k]} |\mathbf{R}_{ii}|} \quad .$$

2.5.3 Lagrange reduction

A basis for a rank-2 lattice Λ can be fully reduced in a polynomial number of operations.

Definition 2.100 (Lagrange-reduced)**.** A basis $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^n$ of a lattice Λ is *Lagrange-reduced* if $\|\mathbf{b}_1\| = \lambda_1(\Lambda)$ and $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$ hold.

Algorithm 2.6 finds a Lagrange-reduced basis in a polynomial number of operations. We remark that $(\_, \mathbf{b}_2) \leftarrow \mathrm{RO}([\mathbf{b}_1], \mathbf{b}_2)$ is equivalent to what is written on Lines 1 and 4. The same goes for NP as $\mathcal{P}([\mathbf{b}_1]) = \mathcal{P}([\mathbf{b}_1^*])$. Figure 2.5 visualises a Lagrange-reduced basis.

Lemma 2.101. *On input a basis $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^n$ of a lattice Λ, Algorithm 2.6 executes the* **while** *loop at most* $\max(1, \lg(\|\mathbf{b}_1\|^2/\det(\Lambda)))$ *times and then outputs a Lagrange-reduced basis for Λ.*

Proof. First, Algorithm 2.6, if it terminates, outputs a basis for Λ as any update of the basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ is of the form $\mathbf{B} \leftarrow \mathbf{BU}$ with $\mathbf{U} \in \mathrm{GL}_2(\mathbb{Z})$. Specifically, Line 3 uses $\mathbf{U}_{\text{swap}} = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, while Lines 1 and 4 use $\mathbf{U}_c = \left(\begin{smallmatrix} 1 & -c \\ 0 & 1 \end{smallmatrix}\right)$ when updating $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - c\mathbf{b}_1$ for some $c \in \mathbb{Z}$.

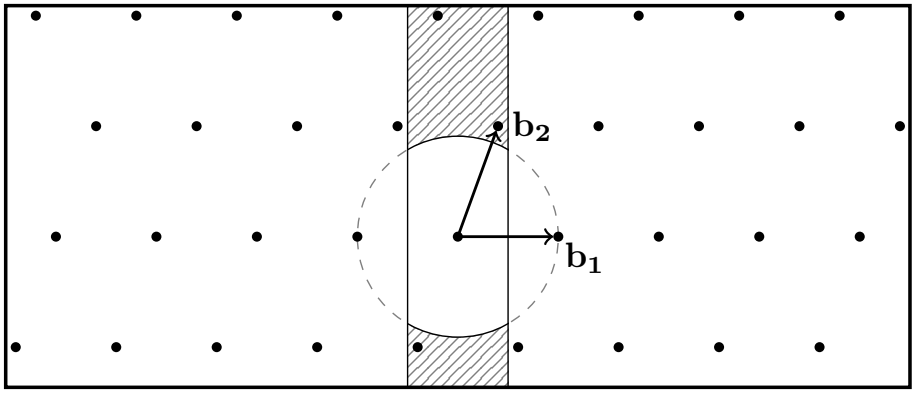


Figure 2.5: Possible final state of Algorithm 2.6

Algorithm 2.6. LagrangeReduce$(\mathbf{b}_1, \mathbf{b}_2)$: Lagrange basis reduction

Input: basis $[\mathbf{b}_1, \mathbf{b}_2]$ of Λ
Output: Lagrange-reduced basis $[\mathbf{b}_1, \mathbf{b}_2]$ of Λ

1: $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \lceil \langle \mathbf{b}_1, \mathbf{b}_2 \rangle / \|\mathbf{b}_1\|^2 \rfloor \cdot \mathbf{b}_1$
2: **while** $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$ **do**
3: swap $\mathbf{b}_1 \leftrightarrows \mathbf{b}_2$
4: $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \lceil \langle \mathbf{b}_1, \mathbf{b}_2 \rangle / \|\mathbf{b}_1\|^2 \rfloor \cdot \mathbf{b}_1$
5: **return** $\mathbf{b}_1, \mathbf{b}_2$

Directly after executing Line 1 or Line 4, Rounding-Off guarantees $\pi_{\mathrm{span}(\mathbf{b}_1)}(\mathbf{b}_2) \in \mathcal{P}([\mathbf{b}_1])$ holds, so we have $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$. The output thus satisfies this as well.

Finally, we bound the number of loop executions and prove that $\|\mathbf{b}_1\| = \lambda_1(\Lambda)$ holds on termination. Note $\|\mathbf{b}_1\|$ strictly decreases in each iteration, and we have the following loop invariant:

$$\det(\Lambda) = \det[\mathbf{b}_1, \mathbf{b}_2] = \|\mathbf{b}_1\| \cdot \|\mathbf{b}_2^*\| \ .$$

Assume after executing Line 3, we have $\|\mathbf{b}_1\|^2 \geq 2 \cdot \det(\Lambda)$. Using the loop invariant and this assumption, $\|\mathbf{b}_2^*\| = \det(\Lambda)/\|\mathbf{b}_1\| \leq \|\mathbf{b}_1\|/2$. Line 4 leaves $\mathbf{b}_2^*$ untouched, while the reduction now guarantees

$$\|\mathbf{b}_2\|^2 \leq \left\| \frac{\mathbf{b}_1}{2} \right\|^2 + \|\mathbf{b}_2^*\|^2 \leq \frac{\|\mathbf{b}_1\|^2}{4} + \frac{\|\mathbf{b}_1\|^2}{4} = \frac{1}{2}\|\mathbf{b}_1\|^2 \ .$$

Thus, one iteration significantly reduces the squared norm of $\mathbf{b}_1$, and the assumption $\|\mathbf{b}_1\|^2 \geq 2 \cdot \det(\Lambda)$ cannot hold for many iterations. Namely, on input $(\mathbf{b}_1^\circ, \mathbf{b}_2^\circ)$, if after k iterations $\|\mathbf{b}_1\|^2 \geq 2 \cdot \det(\Lambda)$ still holds, then we must have $\|\mathbf{b}_1\|^2 \leq 2^{-k}\|\mathbf{b}_1^\circ\|^2$. To prevent a contradiction here, we conclude $\|\mathbf{b}_1\|^2 < 2 \cdot \det(\Lambda)$ holds at the start ($k = 0$) or after $k \leq \lg(\|\mathbf{b}_1^\circ\|^2/(2 \cdot \det \Lambda))$ iterations.

We can conclude $\|\mathbf{b}_1\|^2 < 2 \cdot \det(\Lambda)$ holds after k iterations. Let $\mathbf{v} = \alpha\mathbf{b}_1 + \beta\mathbf{b}_2$ be a shortest nonzero vector of Λ with $\alpha, \beta \in \mathbb{Z}$. If $\|\mathbf{v}\| = \|\mathbf{b}_1\|$, the algorithm terminates with $\|\mathbf{b}_1\| = \lambda_1(\Lambda)$ as there are no nonzero lattice vectors of strictly smaller norm, and the lemma is proved.

Otherwise, $\|\mathbf{v}\| < \|\mathbf{b}_1\|$, and we have $\beta \neq 0$ as $\|\alpha\mathbf{b}_1\| \geq \|\mathbf{b}_1\|$ for nonzero α. The proof of Lemma 2.57 shows $\|\mathbf{v}\| \geq |\beta| \cdot \|\mathbf{b}_2^*\| > |\beta|\sqrt{\det(\Lambda)/2}$, where we use $\|\mathbf{b}_2^*\| = \det(\Lambda)/\|\mathbf{b}_1\| > \sqrt{\det(\Lambda)/2}$ in the second inequality. Because we also have $\|\mathbf{v}\| \leq \|\mathbf{b}_1\| < \sqrt{2\det(\Lambda)}$ and $\beta \in \mathbb{Z}$, we conclude $\beta \in \{-1, 1\}$. Let us assume $\beta = 1$ without loss of generality as we may negate $\mathbf{v}$. Now, $\mathbf{v} = \alpha\mathbf{b}_1 + \mathbf{b}_2$ has a squared norm

$$\begin{aligned}\|\mathbf{v}\|^2 &= \alpha^2\|\mathbf{b}_1\|^2 + \|\mathbf{b}_2\|^2 - 2\alpha \cdot \langle \mathbf{b}_1, \mathbf{b}_2 \rangle \\ &\geq (\alpha^2 - |\alpha|) \cdot \|\mathbf{b}_1\|^2 + \|\mathbf{b}_2\|^2 \geq \|\mathbf{b}_2\|^2 \ ,\end{aligned}$$

using $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$ and $\alpha^2 \geq |\alpha|$ for $\alpha \in \mathbb{Z}$. As $\mathbf{v}$ is a shortest vector of Λ, we conclude $\|\mathbf{b}_2\| = \|\mathbf{v}\| < \|\mathbf{b}_1\|$. Thus, the algorithm continues with iteration $k + 1$. During this iteration, the swap on Line 3 executes $\mathbf{b}_1 \leftarrow \mathbf{b}_2$, making $\mathbf{b}_1$ a shortest vector as $\|\mathbf{b}_1\| = \|\mathbf{v}\| = \lambda_1(\Lambda)$. Thus, the algorithm will return the shortest vector after at most $k + 1$ iterations. □

This Lemma shows that the algorithm can reduce an integer basis $\mathbf{B} \in \mathbb{Z}^{2\times 2}$ in at most $2\lg\|\mathbf{b}_1\|$ iterations, and each iteration performs an arithmetic operation on $\mathcal{O}(\log(\|\mathbf{B}\|_\infty))$ bit numbers, so the overall running time is $\mathcal{O}(\log^3(\|\mathbf{B}\|_\infty))$. More efficient algorithms [Sch91, Yap92] can achieve a runtime of $\mathcal{O}(L \log^{2+\varepsilon} L)$ where $L = \log(\|\mathbf{B}\|_\infty)$ and $\varepsilon > 0$, using fast integer arithmetic [SS71].

2.5.4 Lenstra–Lenstra–Lovász

Lenstra, Lenstra and Lovász provided a polynomial-time basis reduction algorithm [LLL82] that finds a short lattice point of norm at most $2^{k-1}\lambda_1(\Lambda)$ in a rank-k lattice.

LLL reduction has numerous applications such as factoring integer polynomials [LLL82, vH02, Klü10], attacking knapsack-based public key cryptosystems [LO83], and it was used to disprove Mertens' conjecture [OtR85]. A complete book has been written on LLL reduction [NV10]. Moreover, LLL is a fundamental tool in lattice-based cryptography and is used by stronger basis reduction algorithms.

First, we define what an LLL-reduced basis is. Then, we give an algorithm that reduces a basis to such basis using a combination of size-reduction and Lagrange reduction.

Definition 2.102 (LLL-reduced)**.** Let $\mathbf{B} \in \mathbb{R}^{n\times k}$ be a basis and $\delta \in (1/4, 1]$. Then, $\mathbf{B}$ is *δ-LLL-reduced* if it is size-reduced and it satisfies *Lovász' condition*, i.e., for all $i \in [k-1]$, we have

$$\delta\|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2.$$

A 1-LLL-reduced basis of rank two is Lagrange-reduced. An LLL-reduced basis has Gram–Schmidt norms that do not decrease by too much, i.e., writing $\pi_i(\mathbf{b}_{i+1}) = \mu_{i,i+1}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$, we have

$$\|\mathbf{b}_{i+1}^*\|^2 = \|\pi_i(\mathbf{b}_{i+1})\|^2 - \mu_{i,i+1}^2\|\mathbf{b}_i^*\|^2 \geq \left(\delta - \frac{1}{4}\right)\|\pi_i(\mathbf{b}_i)\|^2 \ .$$

This has the following corollary. We note that size-reduction is a stronger requirement than necessary, and we only use Lovász' condition and $|\mu_{i,i+1}| \leq \frac{1}{2}$, which is also implied by Seysen's reduction.

Corollary 2.103. *Let $\delta \in (1/4, 1]$ and $\gamma = 1/\sqrt{\delta - \frac{1}{4}} \geq \sqrt{4/3}$. A δ-LLL-reduced basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ of a lattice Λ satisfies:*

$$\forall i \in [k]\colon \quad \|\mathbf{b}_i^*\| \leq \gamma^{k-i} \cdot \lambda_i(\Lambda) \ , \tag{2.11}$$

$$\|\mathbf{b}_1\| \leq \gamma^{(k-1)/2} \cdot \det(\Lambda)^{1/k} \ . \tag{2.12}$$

Proof. The paragraph above shows $\|\mathbf{b}_j^*\| \leq \gamma\|\mathbf{b}_{j+1}^*\|$ for $j \in [k-1]$, so using induction we get $\|\mathbf{b}_i^*\| \leq \gamma^{j-i} \cdot \|\mathbf{b}_j^*\|$ for all $1 \leq i \leq j \leq k$.

Equation (2.11)**:** Let $\mathbf{v}_1, \ldots, \mathbf{v}_i \in \Lambda$ be a basis for a sublattice of Λ with $\|\mathbf{v}_j\| \leq \lambda_i(\Lambda)$ for all $j \in [i]$. Since the $\mathbf{v}_1, \ldots, \mathbf{v}_i$ generate a rank-i lattice, there exists $j \in [i]$ such that $\pi_i(\mathbf{v}_j) \neq \mathbf{0}$. By Lemma 2.57 we get

$$\lambda_i(\Lambda) \geq \|\mathbf{v}_j\| \geq \|\pi_i(\mathbf{v}_j)\| \geq \min_{i\leq j\leq k}\|\mathbf{b}_j^*\| \geq \|\mathbf{b}_i^*\|/\gamma^{k-i+1} \ .$$

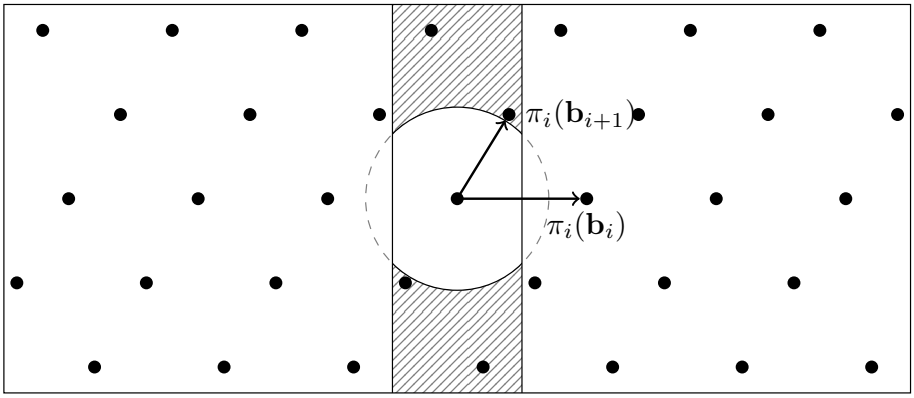


Figure 2.6: Situation in which the **if** condition on Line 4 is satisfied, cf. Figure 2.5. The radius of the circle is $\|\mathbf{b}_i^*\|\sqrt{\delta}$ where $\delta = \frac{1}{2}$.

Equation (2.12)**:** For all $i \in [k]$ we have $\|\mathbf{b}_1\| \leq \gamma^{i-1}\|\mathbf{b}_i^*\|$ so

$$\|\mathbf{b}_1\|^k \leq \prod_{i=1}^{k}\left(\gamma^{i-1}\|\mathbf{b}_i^*\|\right) = \gamma^{k(k-1)/2}\det(\Lambda) \ . \qquad \square$$

In particular, taking $\delta = 1/4$ as in the original LLL paper, yields $\gamma = 2$ and Equation (2.11) becomes $\|\mathbf{b}_1\| \leq 2^{k-1}\lambda_1(\Lambda)$.

A naïve approach to make a basis LLL-reduced is to perform Lagrange reduction on every rank-2 projected sublattice that is not Lagrange-reduced. However, such an algorithm may not terminate in polynomial time as Lagrange reduction may provide too small improvements, causing iteration over a super-polynomial number of lattice points of norm at most $\max_i\|\mathbf{b}_i\|$. The ingenious idea to run in polynomial time, is to *only* perform a Lagrange reduction if $\|\mathbf{b}_i^*\|$ can be improved by some fixed factor $\delta < 1$, which is depicted in Figure 2.6.

Algorithm 2.7 computes an LLL-reduced basis. The algorithm requires $\delta < 1$ to reduce an integer basis in polynomial time. Note that a basis can be size-reduced by Algorithm 2.4, while LLL requires many iterations before a basis satisfies Lovász' condition.

Theorem 2.104 ([LLL82, Prop. (1.26)])**.** *Let $\mathbf{B} \in \mathbb{Z}^{n\times k}$ be an integer basis, $\delta \in (1/4, 1)$ fixed and let $L \in \mathbb{R}$ be an upper bound on all basis vector norms, i.e., for all $i \in [k]$ we have $\|\mathbf{b}_i\| \leq L$. On input an integer basis $\mathbf{B} \in \mathbb{Z}^{n\times k}$ and fixed $\delta \in (1/4, 1)$, Algorithm 2.7 provides an LLL-reduced basis, by using at most $\mathcal{O}(nk^3\log(L))$ arithmetic operations on integers, and these integers have a binary length $\mathcal{O}(k\log L)$.*

Algorithm 2.7. LLL$(\mathbf{B}, \delta)$: LLL basis reduction

Input: basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ of Λ and $\delta \in (1/4, 1)$
Output: LLL-reduced basis $\mathbf{B}$ of Λ

1: $i \leftarrow 1$
2: **while** $i < k$ **do** ▷ invariant: $[\mathbf{b}_1, \ldots, \mathbf{b}_i]$ is LLL-reduced
3: $(\_, \mathbf{b}_{i+1}) \leftarrow \mathrm{NP}([\mathbf{b}_1, \ldots, \mathbf{b}_i], \mathbf{b}_{i+1})$ ▷ reduce $\mathbf{b}_{i+1}$
4: **if** $\delta \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$ **then** ▷ check Lovász' condition
5: $i \leftarrow i + 1$
6: **else**
7: swap $\mathbf{b}_i \leftrightarrows \mathbf{b}_{i+1}$
8: $i \leftarrow \max\{i - 1, 1\}$
9: **return** $\mathbf{B}$

Note, using naïve integer multiplication, this leads to $\mathcal{O}(nk^5 \log^3(L))$ bit operations. This theorem can be proven using the following potential function.

Definition 2.105. The *potential of a basis* $\mathbf{B} \in \mathbb{R}^{n \times k}$ is

$$\mathrm{Pot}(\mathbf{B}) = \prod_{i=1}^{k} \det(\mathbf{B}_{[1,i]}^{\mathsf{T}} \cdot \mathbf{B}_{[1,i]}) = \|\mathbf{b}_1^*\|^{2k} \cdot \|\mathbf{b}_2^*\|^{2k-2} \cdot \cdots \cdot \|\mathbf{b}_k^*\|^2 \ .$$

Proof of Theorem 2.104. The **while** loop on Line 2 has the invariant that is mentioned, and because it holds for $i = k$ on termination, the output is LLL-reduced.

Technically, one would need to show the GSO can be efficiently computed for size-reduction, and for computing $\pi_i(\mathbf{b}_i)$ and $\pi_i(\mathbf{b}_{i+1})$. As the input basis is integral, the squared norm of each Gram–Schmidt vector is rational, and thus can be handled by exact integer arithmetic. Such technical details can be found in the original paper [LLL82]. Instead, we will derive an upper bound on the number of swaps using the potential. Specifically, we will show the potential (1) is bounded from above, (2) is bounded from below by 1 and (3) decreases by a factor δ for each swap, from which the bound on the number of arithmetic operations can be derived.

Initially, we have $\|\mathbf{b}_i^*\| \leq \|\mathbf{b}_i\| \leq L$ so we may bound the potential by $\mathrm{Pot}(\mathbf{B}) \leq L^{k(k+1)}$, which shows (1).

Suppose at some point in the execution, the algorithm works with a basis $\mathbf{C}$. Such a basis has to be an integer basis $\mathbf{C} \in \mathbb{Z}^{n\times k}$ so the Gram matrix $\mathbf{C}_{[1,i]}^{\mathsf{T}} \cdot \mathbf{C}_{[1,i]} \in \mathbb{Z}^{i\times i}$ is an integer. This shows that each factor of $\mathrm{Pot}(\mathbf{C})$ is a positive integer, and thus $\mathrm{Pot}(\mathbf{C}) \in \mathbb{Z}_{\geq 1}$, which shows (2).

Note that size-reduction is a basis transformation on each $\mathbf{B}_{[1,i]}$ so it
2 does not affect $\mathrm{Pot}(\mathbf{B})$. Now suppose the basis is updated from $\mathbf{B}$ to $\mathbf{C}$ by swapping vectors $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$. This means the check on Line 4 failed, so we have $\|\pi_i(\mathbf{b}_{i+1})\|^2 < \delta\|\mathbf{b}_i^*\|^2$, and therefore

$$\begin{aligned}\det(\mathbf{C}_{[1,i]}^{\mathsf{T}}\mathbf{C}_{[1,i]}) = \prod_{j=1}^{i}\|\mathbf{c}_j^*\|^2 &= \|\pi_i(\mathbf{b}_{i+1})\|^2 \cdot \prod_{j=1}^{i-1}\|\mathbf{b}_j^*\|^2 \\ &< \delta\prod_{j=1}^{i}\|\pi_j(\mathbf{b}_j)\|^2 = \delta\det(\mathbf{B}_{[1,i]}^{\mathsf{T}}\mathbf{B}_{[1,i]}) \ .\end{aligned}$$

As $\mathbf{C}_{[1,j]}$ is unchanged for $j < i$, and transformed by some $\mathrm{GL}_k(\mathbb{Z})$ for $j \geq i+1$, going from $\mathbf{B}$ to $\mathbf{C}$, the potential only changes in the jth factor. That is, $\mathrm{Pot}(\mathbf{C}) < \delta\,\mathrm{Pot}(\mathbf{B})$, which shows (3).

Hence, Algorithm 2.7 performs at most $k(k+1)\cdot\log_{1/\delta}(L)$ swaps. □

Algorithm 2.7 can be altered to take as input any generating set $\mathbf{B} \in \mathbb{Z}^{n\times k}$ such that the output is an LLL-reduced basis for the same lattice [Poh87], by detecting linear relations during LLL reduction.

LLL-reducedness is a self-dual property, in the following sense.

Proposition 2.106. *Given a δ-LLL-reduced basis $\mathbf{B} \in \mathbb{R}^{n\times k}$, size-reducing the reverse dual ${}^{\vee}\mathbf{B}$ yields a δ-LLL-reduced basis.*

Proof. The Lovász' condition is a *local* property, i.e., it is a statement on the projected sublattices $\mathbf{B}_{[i,i+1]}$ of rank two, so by Proposition 2.77 the claim follows from the case $k = 2$.

Without loss of generality consider a δ-LLL-reduced basis and its reverse dual in dimension 2, given by

$$\mathbf{B} = \begin{pmatrix} 1 & \mu_{12} \\ 0 & a \end{pmatrix} \quad , \quad \text{and} \quad {}^{\vee}\mathbf{B} = [\mathbf{d}_1, \mathbf{d}_2] = \begin{pmatrix} 0 & 1 \\ \frac{1}{a} & -\frac{\mu_{12}}{a} \end{pmatrix} \quad ,$$

respectively. Because $\mathbf{B}$ is δ-LLL-reduced, we have $|\mu_{12}| \leq \frac{1}{2}$ and $\delta \leq \mu_{12}^2 + a^2$. Hence, we have $\delta\|\mathbf{d}_1\|^2 = \delta/a^2 \leq \mu_{12}^2/a^2 + 1 = \|\mathbf{d}_2\|^2$. □

Floating-point variants

Although Theorem 2.104 shows that LLL reduction runs in polynomial time, it may be too slow in practice to LLL-reduce an integer basis of high dimension using exact integer arithmetic. In this case, one may speed up the algorithm using floating-point numbers to compute the Gram–Schmidt vectors and coefficients [SE94]. However, floating-point arithmetic may lead to numerical stability issues and the algorithm may not terminate or may not output an LLL-reduced basis anymore. Nguyen and Stehlé [NS09] developed a floating-point variant of LLL reduction requiring

$$\mathcal{O}(nk^4(k + \log L) \log L)$$

bit operations using naïve multiplication. This algorithm was the first with a runtime depending quadratically on $\log L$. Moreover, this floating-point variant has been implemented in the fplll software, and is the most widely used lattice reduction software nowadays.

This algorithm computes floating-point approximations of Gram–Schmidt coefficients using the Cholesky decomposition of the Gram matrix. Alternatively, there is an L^2 variant [MSV09] that uses Householder reflections to compute the QR decomposition. These Householder reflections were assumed to be numerically more stable [KS01b, Sch06] as given by a perturbation analysis [CSV12]. This 'H-LLL' algorithm has been implemented in fplll thereafter. Stehlé [Ste10] provides more background on this topic.

There is still active research [KEF21, RH23, DPS25] in designing faster LLL reduction algorithms using a combination of segments [KS01a, KS01b, NS16], iterative compression, Seysen's reduction and linear algebra libraries.

2.5.5 Block Korkine–Zolotarev

While the first basis vector in Corollary 2.103 has a norm that is exponential factor larger than that of a shortest vector, one may obtain smaller factors by spending more time on basis reduction. Algorithm 2.8 contains pseudocode for this so-called *block Korkine–Zolotarev* (BKZ) algorithm [Sch87, SE94]. This algorithm takes a block size β as input that allows to control the runtime and the approximation factor. Its runtime is, depending on the implementation of the SVP oracle, at least exponential in the block size β.

Algorithm 2.8. BKZ($\mathbf{B}, \beta$): BKZ basis reduction

Input: basis $\mathbf{B} \in \mathbb{R}^{n \times k}$ of Λ and block size $2 \leq \beta \leq k$
Output: β-BKZ-reduced basis $\mathbf{B}$ of Λ

1: $\mathbf{B} \leftarrow \mathrm{LLL}(\mathbf{B},\ 0.99)$
2: **repeat**
3: **for** $i = 1, 2, \ldots, k-1$ **do** ▷ one tour
4: $j \leftarrow \min(k, i+\beta-1)$
5: $\mathbf{v} \leftarrow \mathrm{SVP}(\mathbf{B}_{[i,j]})$
6: $(\mathbf{w}, \_) \leftarrow \mathrm{NP}\left(\mathbf{B}_{[1,j]},\ \mathbf{v}\right)$ ▷ lift $\mathbf{v}$ to $\mathbf{w} \in \Lambda$
7: **if** $\mathbf{w} \neq \pm\mathbf{b}_i$ **then**
8: $\mathbf{B}' \leftarrow [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{w}, \mathbf{b}_i, \ldots, \mathbf{b}_k]$ ▷ insert $\mathbf{w}$
9: $\mathbf{B} \leftarrow \mathrm{LLL}(\mathbf{B}',\ 0.99)_{[1,k]}$ ▷ remove linear dependence
10: **until** $\mathbf{B}$ has not changed
11: **return** $\mathbf{B}$

Line 5 calls an oracle SVP that, on input $\mathbf{C}$, is able to find a shortest vector in the lattice $\mathcal{L}(\mathbf{C})$ of rank at most β. This oracle can be realised using lattice point enumeration [GNR10], or lattice sieving [AKS01]. The runtime grows exponentially as a function of the block size β, even when require the oracle to output a lattice point of a random lattice having norm $1.05\,\lambda_1(\Lambda)$, for example.

Line 8 inserts the vector $\mathbf{w}$ into the basis $\mathbf{B}$ at index i, causing $\mathbf{B}'$ to be rank-deficient as $\mathbf{w}$ is a linear combination of $\mathbf{b}_1, \ldots \mathbf{b}_j$. Line 9 turns $\mathbf{B}$ back into a basis by eliminating the linear dependency using Pohst's modification of LLL [Poh87]. That is, column $k+1$ of matrix $\mathbf{B}'$ is the zero vector after LLL-reduction.

Algorithm 2.8 performs many tours because of Line 10. Yet, the first tours improve the basis the most. Hence, one limits the number of tours in practice and the norm of the first basis vector can still be bounded heuristically [HPS11].

Lemma 2.107 ([LN25])**.** *Suppose Algorithm 2.8 receives as input a basis $\mathbf{B}' \in \mathbb{R}^{n \times k}$ and a block size β. After $\Theta(k^2 \log(k)/\beta^2)$ executions of the loop on Line 2, the first basis vector $\mathbf{b}_1$ of the reduced basis $\mathbf{B}$ has norm,*

$$\|\mathbf{b}_1\| \leq \gamma_\beta^{\frac{k-1}{2(\beta-1)} + \frac{\beta(\beta-2)}{2k(\beta-1)}} \cdot \det(\Lambda)^{1/k} \quad ,$$

where γ_β is defined in Definition 2.17.

Proposition 2.92 shows one needs short Gram–Schmidt vectors to have a good basis. Therefore, we use the basis profile to determine the reduction quality.

Definition 2.108 (basis profile and quality)**.** The *profile* of a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ is the sequence $(\lg\|\mathbf{b}_i^*\|)_{i=1}^k$. We consider three quality measures:

(1) the *kth root Hermite factor* (RHF) given by

$$\delta(\mathbf{B}) = \frac{\|\mathbf{b}_1\|^{1/k}}{(\det\Lambda)^{1/k^2}} \ ,$$

not to be confused with the parameter of Algorithm 2.7;

(2) the *slope* of the line given by simple linear regression on the graph of the basis profile, i.e.,

$$\mathrm{sl}(\mathbf{B}) = \frac{\sum_{i=1}^k \lg\|\mathbf{b}_i^*\| \left(i - \frac{k+1}{2}\right)}{\sum_{i=1}^k \left(i - \frac{k+1}{2}\right)^2} \ , \text{ and}$$

(3) the *potential* as in Definition 2.105.

For random lattices, the slope is usually a negative number as Gram–Schmidt norms generally decrease. Intuitively, a good basis has a small potential, which corresponds to a slope that is close to zero or positive.

Proposition 2.109. *Given a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$, we have*

$$\lg \mathrm{Pot}(\mathbf{B}) + \frac{(k+1)k(k-1)}{6}\mathrm{sl}(\mathbf{B}) = (k+1)\lg(\det\Lambda) \ .$$

Proof. Let us denote $S_k = \sum_{i=1}^k \left(i - \frac{k+1}{2}\right)^2$ for the denominator in the definition of $\mathrm{sl}(\mathbf{B})$. It is easy to verify $S_k = \frac{(k+1)k(k-1)}{12}$ because S_k is a cubic polynomial in k that equals the given identity when plugging in $k = 0, 1, 2$ and $k = 3$. Therefore,

$$2S_k \cdot \mathrm{sl}(\mathbf{B}) = \sum_{i=1}^k 2i\lg\|\mathbf{b}_i^*\| - (k+1)\cdot\lg(\det\Lambda) \qquad \text{and}$$

$$\lg\mathrm{Pot}(\mathbf{B}) = 2(k+1)\cdot\lg(\det\Lambda) - \sum_{i=1}^k 2i\lg\|\mathbf{b}_i^*\| \ .$$

As a consequence we get $\lg\mathrm{Pot}(\mathbf{B}) + 2S_k\cdot\mathrm{sl}(\mathbf{B}) = (k+1)\cdot\lg(\det\Lambda)$. □

An LLL-reduced basis using LLL-parameter $\delta = 0.99$ has $\delta(\mathbf{B}) < 1.08$ as can be computed from Equation (2.12). For a random lattice, the expected RHF of an LLL-reduced basis is $\delta(\mathbf{B}) \approx 1.02$ [NS06].

Using the bound on γ_β and $\beta(\beta-2)/k(\beta-1) < 1$, Lemma 2.107 proves a BKZ-reduced basis has an RHF bounded by

$$\delta(\mathbf{B}) \leq (2\,\mathrm{GH}(\beta))^{\frac{1}{\beta-1}+\frac{1}{k}} \quad . \tag{2.13}$$

For random lattices, heuristically a BKZ-reduced basis has a smaller RHF than the worst-case bound [GN08], similar to the RHF after LLL-reduction.

Heuristic 2.110 (BKZ behaviour)**.** *Let β be fixed. On input block size β and some basis $\mathbf{B}' \in \mathbb{R}^{n\times n}$ of a full-rank random lattice Λ, Algorithm 2.8 outputs a basis $\mathbf{B}$ having RHF $\delta(\mathbf{B}) \sim \mathrm{GH}(\beta)^{1/(\beta-1)}$ as $n \to \infty$.*

See Chen's thesis for a detailed justification [Che13, Sec. 4.1].

Heuristic Justification. In short, the SVP oracle guarantees that $\mathbf{b}_i^*$ is a shortest vector of the lattice generated by the basis $\mathbf{B}_{[i,i+\beta-1]}$ for any $1 \leq i \leq n-\beta$. All these rank-β lattices are random, so we may use Heuristic 2.41 to obtain

$$\log\|\mathbf{b}_i^*\| = \log \mathrm{GH}(\beta) + \frac{1}{\beta}\sum_{j=i}^{i+\beta-1} \log\|\mathbf{b}_j^*\| \quad .$$

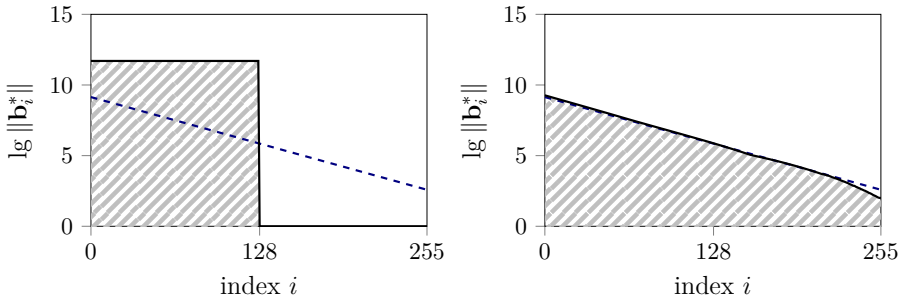
Writing $a_i = \log\|\mathbf{b}_i^*\|$ and $c = \frac{\beta}{\beta-1}\log \mathrm{GH}(\beta)$, there is a recurrence relation,

$$a_i = c + \frac{a_{i+1} + \cdots + a_{i+\beta-1}}{\beta-1} \quad .$$

A solution to this recurrence is given by $a_{i+1} = a_i - 2c/\beta$ for $i \geq 1$. Approximating the last β Gram–Schmidt norms by the solution of the recurrence yields

$$\begin{aligned}\log\left(\frac{\|\mathbf{b}_1\|^{1/n}}{\det(\Lambda)^{1/n^2}}\right) &= \frac{a_1}{n} - \frac{1}{n^2}\sum_{i=1}^{n} a_i = \frac{2c}{\beta n^2}(1+2+\cdots+n) \\ &= \frac{c(n+1)}{\beta n} \xrightarrow{(\text{as } n\to\infty)} \frac{c}{\beta} = \frac{\log(\mathrm{GH}(\beta))}{\beta-1} \quad . \qquad \triangle\end{aligned}$$

After BKZ-reduction, one may approximate the basis profile by a line [Sch03], see Figure 2.7 for motivation.



(a) Input (b) Output

Figure 2.7: Example of the basis profile before (a) and after (b) BKZ reduction using a q-ary lattice of dimension 256. The dashed blue line shows the GSA prediction.

Heuristic 2.111 (geometric series assumption, GSA)**.** *On input a block size β and a basis for a random lattice, Algorithm 2.8 outputs a basis* $\mathbf{B}$ *having Gram–Schmidt norms that form a geometric series. Equivalently, the basis profile lies on a line.*

It is known that the GSA does not accurately predict the last β Gram–Schmidt norms [AD21], which is also clear from Figure 2.7. Additionally, some of the first Gram–Schmidt norms are smaller than the GSA prediction when β is small [BSW18].

BKZ 2.0 [CN11] is a variant of BKZ consisting of four improvements:

(1) Early aborts [HPS11], which significantly improves over plain BKZ as the number of calls to the SVP oracle grows exponentially [GN08].
(2) Sound pruning [GNR10], i.e., use a pruned lattice point enumeration tree, introducing a probability that the SVP oracle is successful. If the SVP oracle was unsuccessful, we retry using a rerandomised local basis.
(3) Preprocessing of local bases, i.e., a local basis is reduced by BKZ using a smaller block size $\beta' < \beta$ instead of LLL.
(4) Taking a shorter enumeration radius, i.e., the SVP oracle on Line 5 is asked to find a vector of an approximate length according to Heuristic 2.41 that is shorter than $\|\mathbf{b}_i^*\|$.

BKZ 2.0 is implemented in fplll. Moreover, there exist models for predicting the Gram–Schmidt norms when using BKZ 2.0 to reduce a

basis for a random lattice [CN11, BSW18, DDGR20, PV21].

HKZ reduction

Algorithm 2.8 is called HKZ-reduction for $\beta = k$.

2

Definition 2.112 (HKZ-reduced)**.** A basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ of a lattice Λ is *Hermite–Korkine–Zolotarev-reduced* (HKZ-reduced) if it is size-reduced, and

$$\forall i \in [k]\colon \|\mathbf{b}_i^*\| = \lambda_1(\pi_i(\Lambda)) \ . \tag{2.14}$$

This notion of reduction is named after Hermite [Her1850] and Korkine–Zolotarev [KZ1873] for their work on size-reduction and Equation (2.14), respectively.

HKZ-reduction of a lattice is computationally expensive as the oracle is asked to solve SVP in dimension k. Note that the basis is already HKZ-reduced when Algorithm 2.8 reaches Line 10. Thus, for $\beta = k$, after performing one tour in Algorithm 2.8, the basis is HKZ-reduced and the algorithm can return the basis.

Note that BKZ-reduction using block size β causes the projected sublattice $\mathbf{B}_{[n-\beta+1,n]}$ to be HKZ-reduced.

2.6 Lattice sieving

Lattice sieves provide a way to efficiently produce a list of many short lattice vectors [NV08, MV10, BDGL16]. Although there are many variations, e.g. [BLS16], one may think of a sieve as initially generating a list L of random linear combinations of some basis vectors defining a lattice $\Lambda \subset \mathbb{R}^n$, and then iteratively sieving, i.e., finding reductions that replace $\mathbf{v} \in L$ by a shorter $\mathbf{v} - \mathbf{w}$ for some $\mathbf{w} \in L$.

Throughout this thesis, we assume a lattice sieve will never return both $\mathbf{w}$ and $-\mathbf{w}$, since this optimisation is used in most implementations.

Definition 2.113 (lattice sieve)**.** A lattice sieve is a probabilistic algorithm that has the following input and output. Input consists of a basis $\mathbf{B} \in \mathbb{R}^{n\times k}$ of a random lattice Λ, a saturation radius $r_{\text{sat}} \in \mathbb{R}_{\geq 1}$ and saturation ratio $f_{\text{sat}} \in (0, 1]$. Output is a list of $N = \left\lceil \frac{1}{2} f_{\text{sat}} r_{\text{sat}}^n \right\rfloor$ lattice points, each of norm at most $r_d = r_{\text{sat}}\,\mathrm{GH}(k) \cdot \det(\Lambda)^{1/k}$. We use $\textsc{Sieve}(\Lambda, r_{\text{sat}}, f_{\text{sat}})$ to denote such a lattice sieve.

In order to analyse an algorithm that uses lattice sieving, we make a heuristic assumption regarding the distribution of the lattice vectors that a lattice sieve algorithm outputs.

Heuristic 2.114. *Let $\Lambda \subset \mathbb{R}^n$ be a full-rank unit-volume lattice and let $r_{\text{sat}} \in \mathbb{R}_{\geq 1}$, $f_{\text{sat}} \in (0, 1]$. The output* $\text{Sieve}(\Lambda, r_{\text{sat}}, f_{\text{sat}})$ *is a list of N i.i.d. samples from $\mathrm{U}(r_d \mathcal{B}^n)$ where $r_d = r_{\text{sat}}\,\mathrm{GH}(n)$.*

This heuristic makes three simplifying assumptions on the output of a lattice sieve.

(1) The output list is assumed to consist of N i.i.d. samples from the set of possible outputs, i.e., lattice vectors of norm at most r_d. Thus, the simplified list may have duplicates. Yet, there is a moderate difference with the actual output of a sieve, as sampling also produces many distinct values. Namely, using linearity of expectation one can show that sampling n times uniformly from $[n]$ yields in expectation $n(1 - 1/e) \approx 0.63n$ distinct values as $n \to \infty$.
(2) Moreover, the output samples are assumed to follow the uniform distribution on $\Lambda \cap (r_d \mathcal{B}^n)$.
(3) Last, we forget the lattice structure Λ in the output list. Heuristic 2.42 predicts that the norm distribution of lattice vectors is similar to that of points uniformly from a ball. When sieving a random lattice, this assumption seems fair, because we do not expect the lattice to be distorted in any particular direction.

We remark that this heuristic does not assume all vectors are of the same length r_d, cf. Guo–Johansson [GJ21]. Instead, the heuristic predicts there may be some shorter vectors, albeit with smaller probability. Still, most of the vectors are close to the boundary of the ball.

By default, a lattice sieve uses a saturation radius of $r_{\text{sat}} = \sqrt{4/3}$, because in that case enough vectors are initially generated to reduce vectors in the sieve in each step [NV08]. Although in theory, one sometimes assumes a lattice sieve finds *all* lattice vectors inside a ball of radius r_d, i.e., $f_{\text{sat}} = 1$, in practice a much lower saturation ratio is chosen for efficiency. For instance, the G6K software [ADH+19] uses a saturation ratio of 0.5 by default, and even lower saturation ratios of 0.375 were used in sieves to break certain SVP challenges with GPUs [DSvW21].

One can also sieve on a lattice that is not full-rank. Sieve algorithms require both time and memory exponential in the rank of Λ in general [BDGL16].

2.7 Signature scheme

A *signature scheme* is a triple of probabilistic polynomial-time algorithms $\Pi = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify})$ such that Verify is deterministic.

- On input 1^n, KeyGen outputs a public and private key $(\mathsf{pk}, \mathsf{sk})$. We assume n can be determined from either key.
- On input sk and a message m from a message space that may depend on pk, Sign outputs a signature sig.
- On input pk, a message m and a signature sig, Verify outputs a bit $b \in \{0, 1\}$. We say sig is a valid signature on m if and only if $b = 1$.

In our practical cryptanalysis of Section 5.3 we discuss two types of forgery an adversary may produce, strong and weak.

- A *strong forgery* is a valid signature on a message for which an adversary does not know a signature. If an adversary cannot produce a strong forgery, we call the signature scheme *weakly unforgeable*, or EUF-CMA secure.
- A *weak forgery* is a valid signature on a message for which an adversary already knows a valid signature. Moreover, to be a weak forgery on a message m, this signature must be different from all the valid signatures on m known to the adversary. If an adversary can produce neither weak nor strong forgery, we call the signature scheme *strongly unforgeable*, or SUF-CMA secure.

Part II
Dual-Sieve Attack

Chapter 3

Dual-Sieve Attack Heuristics

This chapter is based on joint work with L. Ducas published at CRYPTO 2023 [DP23c].

Guo and Johansson [GJ21], and MATZOV [MAT22] have independently claimed improved attacks against various lattice-based schemes in NIST's PQC standardisation process, such as ML-KEM and ML-DSA, by using an FFT on top of the so-called Dual-Sieve attack.

However, this chapter shows that a heuristic used in above works not only theoretically contradicts with both formal theorems and well-tested heuristics in certain regimes, but also provides incorrect predictions in experiments. We conclude that this heuristic significantly overestimates the success probability of the Dual-Sieve attack.

In the process, it is shown that use of the FFT is not specific to a Dual-Sieve attack against learning with errors but is more generally useful against bounded distance decoding.

3.1 Introduction

Many post-quantum cryptosystems base their security on the hardness of the learning with errors (LWE) problem, introduced by Regev in 2005 [Reg05, Reg09], which is basically the bounded distance decoding (BDD) problem in q-ary lattices. One possible type of attack against BDD is the so-called *dual attack*, which uses the idea by Aharonov and Regev [AR04] that short dual vectors allow distinguishing points close to a lattice from points far away from the lattice. This idea can even be traced back in a pure geometric context to earlier work by Håstad [Hås88], and is not even limited to lattices, as it was already implicit in the very construction of low-density parity-check codes [Gal62]. In contrast, a lattice-based attack operating solely on the *primal* lattice is called a *primal attack*. The best dual and primal attacks are then typically used by the lattice cryptanalyst to parametrise LWE cryptosystems.

More specifically, the dual attack attempts to solve the search-BDD problem by performing a reduction to the decision-BDD problem, where one is asked to determine whether a target point in Euclidean space is either a *uniform target*, i.e., sampled uniformly modulo the lattice, or a *BDD target*, i.e., sampled from a distribution that is concentrated around lattice points. Here, a short dual vector provides a score function that assigns a high score to BDD targets and an expected score of zero to uniform targets. Only by using many dual vectors and considering the sum of individual scores [LW21], one can hope to distinguish successfully between the two target distributions with a high probability. To get exponentially many short dual vectors, Alkim, Ducas, Pöppelmann and Schwabe initially suggested [ADPS16] to use a lattice sieve [NV08, MV10, BDGL16] on the dual lattice.

We refer to this style of attack as a *Dual-Sieve* attack. The concrete cryptanalytic impact of this idea can be further improved by guessing multiple coordinates of the secret rather than just one [Alb17, EJK20], and then finding the right solution among these candidates rather than just a few.

Another development of the dual attack is also reminiscent from a cryptanalytic technique of code-based cryptography by Levieil and Fouque [LF06]. The idea is to batch the score evaluation of many algebraically related candidates via a *fast Fourier transform* (FFT). For carefully crafted parameters, the cost of computing all those scores is

barely larger than the cost of naïvely computing a single score. This led Guo and Johansson [GJ21] to claim an improved attack on various NIST post-quantum standardisation candidates, followed quickly by an independent technical report by Israel's Center of Encryption and Information Security (MATZOV) [MAT22]. We refer to this style of attack as a *Dual-Sieve-FFT* attack. The latter has already been followed up upon, with a quantum variant [AS23] and a coding-theoretic enhanced variant [CST22]. In their analysis of the Dual-Sieve attack, all these four works use a heuristic saying that individual score functions given by a set of dual vectors, are mutually independent variables.

3.1.1 Contributions

This chapter provides three contributions.

Generalisation of the Dual-Sieve-FFT attack (Section 3.4)

The original principle of the dual attack [AR04, Hås88] is stated for the BDD problem in any lattice. However, the recent instances of the Dual-Sieve attack [ADPS16, Alb17, EJK20] and the Dual-Sieve-FFT attack [GJ21, MAT22, CST22, AS23] are stated specifically for LWE. One exception is the work of [LW21], which is geometrically enlightening but limited to the Dual-Sieve attack.

Our first contribution is therefore to generalise the FFT trick [GJ21] to the setting of BDD. Beyond the theoretical satisfaction of abstracting the technique to its mathematical core, this generalisation also offers further improvement over the work of [GJ21]. Namely, for the same algorithmic cost, we can further improve the shortness of the dual vectors and therefore their distinguishing power.

Contradictions to the independent score heuristic (Section 3.5)

A second observation regarding this literature is that the analysis of the Dual-Sieve attack (with or without FFT) relies on one specific heuristic, which has received essentially no attention so far. Namely, it is assumed that all the individual scores, given by each dual vector, are mutually independent.

We approach the analysis of this heuristic by looking at the conclusions it leads to. The geometric point of view offered by the work of Laarhoven and Walter [LW21] is pivotal in that respect: judging the reasonability of

a heuristic conclusion is very much enabled by the language of geometry. In particular, [LW21] concludes with a heuristic algorithm that solves decision-BDD, even when the noise slightly exceeds the Gaussian heuristic (GH), i.e., the expected minimal distance of a random lattice. This should raise suspicion, as a random point is on average not expected to be further than GH away from the lattice. We show this suspicion is fair: it follows from a result of Debris-Alazard, Ducas, Resch, and Tillich [DADRT23] that the above task is statistically impossible to solve, even to an unbounded attacker.

3

The contradiction above is, however, limited to a rather theoretical regime of the Dual-Sieve attack, which is not that of the recent concrete cryptanalytic claims [EJK20, GJ21, MAT22, CST22, AS23], where the noise is below GH. Still, here the attack needs to distinguish between a BDD sample and a uniform sample with a very high success probability. We will show that this situation also leads to a contradiction.

Namely, we argue that the heuristic probability of having a distinguishing failure is much smaller than the probability of a uniform target lying closer to the lattice than a BDD target. The claimed high success probability of distinguishing between BDD and uniform contradicts the basic principle that targets closer to the lattice get a higher score in expectation. It turns out that the parameters used in [GJ21, MAT22] specifically fall into that contradictory regime that requires a distinguisher with a very high success probability.

Experimental invalidation of prior model (Section 3.6)

To develop a more precise understanding, we zoom in on the score distribution for BDD targets and uniform targets. Extensive experiments were performed to discover that both distributions deviate from the predictions made under the independent score heuristic. First, the *body* of the distribution of scores for uniform targets is properly predicted, but not its *tail*: after a predicted rapid decrease, visually resembling a *waterfall*, this distribution hits a *floor*. This is perfectly in line with our second contradictory regime: some uniform targets will be close to the lattice, and should therefore have a high score.

The score distribution for BDD targets is also predicted incorrectly, and this is no longer just a matter of the tail. Contrary to prediction, this score is not distributed like a Gaussian. It is in fact not even symmetric around its average, and its variance appears exponentially larger than

predicted. In particular, a BDD target will more likely have a low score than what is heuristically predicted.

Open data

All the written code and the obtained data to generate the figures, can be found in the GitHub repository https://github.com/ludopulles/DoesDualSieveWork. The experiments are written in Python and use the G6K and FPyLLL libraries [ADH+19, FPL25], and there is a binding to some C code to accelerate the FFT.

3.1.2 Coding theory analogue

In coding theory, the dual attack is called "statistical decoding" [Gal62, AlJ01, Ove06]. Here, with the use of many low-weight parity-check equations $\mathcal{H}$, which is analogous to short dual lattice vectors, one can decide whether a target $\mathbf{t} \in \mathbb{F}_q^n$ is close to some codeword or not, based on a similar scoring function.

Recently, there was a work by Carrier, Debris-Alazard, Meyer-Hilfiger and Tillich [CDMT22] that claims to have an improved algorithm for statistical decoding that outperforms the state-of-the-art information set decoding algorithm [BM18] on sparse binary codes. However, this work was based on the coding-theory analogue of the independent score heuristic [CDMT22, Assumption 3.7], which states that individual scores $(\langle \mathbf{h}, \mathbf{t} \rangle)_{\mathbf{h} \in \mathcal{H}}$ are i.i.d. Bernoulli variables. Here, similarly it was soon realised this heuristic leads to a flawed analysis, and two proposed fixes [MT23] were enough to overcome the problems. Namely, first, given a $[n, k]$-code $\mathcal{C}$, they model the number of weight w elements in a coset $\mathcal{C} + \mathbf{t}$ for $\mathbf{t} \leftarrow \mathrm{U}(\mathbb{F}_2^n)$ as a Poisson process with parameter $\lambda = \binom{n}{w}/2^{n-k}$. In addition, they use Fourier analysis on the score function, and using both fixes, they then show that the statistical decoding algorithm can be corrected while only increasing the asymptotic runtime cost by a logarithmic factor.

Remark 3.1. Our code $\mathcal{C}$ in the paragraph above is written in [MT23, Prop. 1] as a shortened $[n - s, k - s]$-code, instead. This shortened code plays a similar rôle as the sublattice in our reduction from search-BDD to decision-BDD in Section 3.2.2.

3.2 Dual attack

3.2.1 Solving decision-BDD

The idea of using short dual vectors for solving decision-BDD can be traced back at least to [AR04] in the lattice literature, and can be viewed as the lattice analogue of an old decoding technique [Gal62, AlJ01, Ove06]. Given a BDD sample $\mathbf{t} = \mathbf{v} + \mathbf{e}$ with $\mathbf{v} \in \Lambda$, for any dual vector $\mathbf{w} \in \Lambda^\vee$ one has,

$$\langle \mathbf{t}, \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle + \langle \mathbf{e}, \mathbf{w} \rangle \equiv \langle \mathbf{e}, \mathbf{w} \rangle \pmod 1 \ .$$

In particular, if the error $\mathbf{e}$ and the dual vector $\mathbf{w}$ are of small enough ℓ_2-norm, $\langle \mathbf{t}, \mathbf{w} \rangle$ should be close to an integer. Moreover, $\langle \mathbf{t}, \mathbf{w} \rangle \pmod 1$ is thus well-defined for any $\mathbf{t} \in \mathbb{R}^n/\Lambda$. A natural score to consider as some indication of the target $\mathbf{t}$ being close to the lattice Λ is therefore given by,

$$f_{\mathbf{w}}(\mathbf{t}) = \frac{\chi_{\mathbf{w}}(\mathbf{t}) + \chi_{-\mathbf{w}}(\mathbf{t})}{2} = \cos(2\pi \cdot \langle \mathbf{t}, \mathbf{w} \rangle) \ ,$$

where $\chi_{\mathbf{w}}$ is defined in Lemma 2.73.

If a target $\mathbf{t}$ is indeed close to the lattice, $f_{\mathbf{w}}(\mathbf{t})$ should be close to 1, but the converse does not need to be true. To improve confidence in the fidelity of this score, one may naturally consider the total score over many dual vectors $\mathcal{W} \subset \Lambda^\vee$ given by,

$$f_{\mathcal{W}}(\mathbf{t}) = \sum_{\mathbf{w} \in \mathcal{W}} f_{\mathbf{w}}(\mathbf{t}) \ .$$

This function is called the simple distinguisher f_{simple} by Laarhoven and Walter [LW21], and it closely resembles the Aharonov–Regev [AR04] distinguisher, given by

$$f^{\text{AR}}_{\mathcal{W}}(\mathbf{t}) = \sum_{\mathbf{w} \in \mathcal{W}} \rho_{1/(2\pi\sigma)}(\mathbf{w}) f_{\mathbf{w}}(\mathbf{t}) \ .$$

By checking whether $f_{\mathcal{W}}(\mathbf{t})$ is above or below some optimally chosen threshold, one can say from which distribution $\mathbf{t}$ is drawn, effectively solving decision-BDD with a high success probability.

In carefully crafted circumstances, in particular regarding the construction of the set of short dual vectors $\mathcal{W}$, this approach can give a provable worst-case distinguisher [AR04] or certificate [Hås88].

More recent works [LW21, GJ21, MAT22] have reused this idea more heuristically, in a context where $\mathcal{W}$ simply is the set of all the dual vectors smaller than a certain radius. In such a situation, $\mathcal{W}$ is typically the output of a sieve algorithm [BDGL16]. Moreover, all the nonzero dual vectors have approximately the same weight, and [LW21] shows that the simple distinguisher $f_{\mathcal{W}}$ works asymptotically almost as good as the Aharonov–Regev one $f^{\mathrm{AR}}_{\mathcal{W}}$, although the former can be computed faster in practice.

3.2.2 Reducing search-BDD to decision-BDD

Recent dual attacks [EJK20, GJ21, MAT22] and follow-up works reduce search-BDD on Λ to decision-BDD on a (projected) sublattice as follows. A search-BDD instance consists of a lattice Λ and a BDD sample $\mathbf{t} = \mathbf{v}+\mathbf{e}$ where $\mathbf{v} \in \Lambda$ and $\mathbf{e}$ has small norm. First, a sparsification $\Lambda' \subset \Lambda$ is chosen. Then, we try to solve decision-BDD on the sublattice Λ' with $|\Lambda/\Lambda'|$ many targets: $(\mathbf{t}-\mathbf{g})_{\mathbf{g}+\Lambda' \in \Lambda/\Lambda'}$. There is exactly one target drawn from a BDD distribution, namely the target $\mathbf{t}-\mathbf{g}$ where $\mathbf{g} \in \mathbf{v}+\Lambda'$ is the 'correct guess.' If decision-BDD is solved successfully, i.e., only $\mathbf{t}-\mathbf{g}$ is identified as a BDD sample for the correct guess $\mathbf{g}$, then search-BDD is solved modulo Λ', i.e., $\mathbf{v} \equiv \mathbf{g} \pmod{\Lambda'}$. Now, Λ' and $\mathbf{t}-\mathbf{g}$ forms a new search-BDD instance that is easier because the new lattice Λ' is sparser and the distance between $\mathbf{t}-\mathbf{g}$ and Λ' is at most $\|\mathbf{e}\|$. By recursively solving this easier search-BDD instance, one may ultimately recover $\mathbf{v}$ completely. gamely, after a certain number of iterations, say ℓ, the ℓth easier search-BDD instance $(\Lambda^{(\ell)}, \mathbf{t}^{(\ell)})$ is easy enough that Babai's NP algorithm, upon input $\Lambda^{(\ell)}$ and $\mathbf{t}^{(\ell)}$, outputs $\mathbf{v}' \in \Lambda^{(\ell)}$ such that $\mathbf{t}^{(\ell)} = \mathbf{v}' + \mathbf{e}$ holds. In this case, $\mathbf{v} = \mathbf{v}' + \sum_{i=1}^{\ell} \mathbf{g}^{(i)}$, where $\mathbf{g}^{(i)}$ denotes the correct guess in the ith iteration, is the solution of the original search-BDD instance.

Remark 3.2. Many of the recent dual attacks, such as Guo–Johansson and MATZOV [GJ21, MAT22], however, run a sieve on a lattice $L \subset (\Lambda')^{\vee}$ of much lower rank $\beta_{\mathrm{sieve}} < n$, from which a set of dual vectors $\mathcal{W} \subset L$ is obtained, to solve decision-BDD. Observe that the distinguisher function now satisfies

$$f_{\mathcal{W}}(\mathbf{t}) = f_{\mathcal{W}}(\pi_{\mathrm{span}(L)}(\mathbf{t})) \ ,$$

where π_V denotes the projection map onto the vector space V. This shows that $f_{\mathcal{W}}$ is now merely a distinguisher for $L^{\vee}$: it assigns high scores

to all targets $\mathbf{t} \in \mathbb{R}^n$ that have their projection $\pi_{\mathrm{span}(L)}(\mathbf{t})$ close to $L^\vee$.

More specifically, recent dual attacks first perform lattice reduction, such as BKZ, on Λ' to obtain some reduced basis $\mathbf{D}$ for $(\Lambda')^\vee$. Then, L is selected as the lattice generated by the basis $\mathbf{D}_{[1,\beta_{\mathrm{sieve}}]}$. By duality, then $({}^\vee\mathbf{D})_{[n-\beta_{\mathrm{sieve}}+1,n]}$ is a basis for $L^\vee$, obtained by projecting Λ' away from the first $n-\beta_{\mathrm{sieve}}$ vectors of ${}^\vee\mathbf{D}$.

Note, when $\mathbf{e}$ follows an n-dimensional Gaussian distribution with parameter σ, then $\pi_{\mathrm{span}(L)}(\mathbf{e})$ follows a β_{sieve}-dimensional Gaussian distribution with parameter σ. On the other hand, if a guess $\mathbf{g} \notin \mathbf{v}+\Lambda'$ is incorrect, then $f_{\mathcal{W}}$ assigns a high score to the target $\mathbf{t}-\mathbf{g}$ when the point

$$\pi_{\mathrm{span}(L)}(\mathbf{t}) = \pi_{\mathrm{span}(L)}(\mathbf{e}) + \pi_{\mathrm{span}(L)}(\mathbf{v}-\mathbf{g}), \tag{3.1}$$

is close to $L^\vee$. Since the lower rank lattice L was only picked after some BKZ reduction, and $\mathbf{v}-\mathbf{g}$ is nonzero, we presume that $\pi_{\mathrm{span}(L)}(\mathbf{v}-\mathbf{g})$ acts as a uniform point in $\mathrm{span}(L)/L$. In conclusion, because the distinguisher is only distinguishing for $L^\vee$, we see that the target in Equation (3.1) corresponds to a BDD sample if $\mathbf{g} \in \mathbf{v}+\Lambda'$, and else to a uniform sample modulo $L^\vee$.

Remark 3.3. We note that the work of [PS24] sidesteps the uniform model for incorrect targets, by using dual vectors from the full-rank lattice Λ' and assuming a priori $\|\mathbf{e}\| < \frac{1}{2}\lambda_1(\Lambda)$. In this case, incorrect targets can be proved to be far enough from the lattice in the worst case, without any statistical nor heuristic argument as it was already the case in [AR04]. This is, however, not the regime used in recent concrete attack claims [GJ21, MAT22], and is in fact separated from the contradictory regime in Figure 3.3 by a factor 2 on the error length $\|\mathbf{e}\|$.

3.3 Prior dual attack models

In this subsection, we explain how the Dual-Sieve attack is analysed in the works of [LW21, GJ21, MAT22], and explicitly state where they have used heuristics.

3.3.1 Laarhoven–Walter

First, Laarhoven and Walter analyse in [LW21, Lem. 6] the distribution of the score $f_{\mathbf{w}}(\mathbf{t})$ for BDD targets $\mathbf{t}$ with a distance of exactly r to

the primal lattice. In the derivation, however, they approximate this distribution by targets sampled from a continuous Gaussian of parameter $\sigma = r/\sqrt{n}$.

Lemma 3.4 ([LW21, Lem. 6]). *Let $\Lambda \subset \mathbb{R}^n$ be a full-rank lattice and $\mathbf{w} \in \Lambda^\vee$ be a dual vector.*

(a) If $\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$, then $\mathbb{E}[f_{\mathbf{w}}(\mathbf{t})] = 0$, and $\mathbb{V}[f_{\mathbf{w}}(\mathbf{t})] = 1/2$,

(b) If $\mathbf{t} \leftarrow \mathcal{N}_n(0, \sigma^2) \pmod{\Lambda}$ with $\sigma \in \mathbb{R}_{>0}$, then

$$\mathbb{E}[f_{\mathbf{w}}(\mathbf{t})] = e^{-2\pi^2\sigma^2\|\mathbf{w}\|^2}, \quad \text{and} \quad \mathbb{V}[f_{\mathbf{w}}(\mathbf{t})] = \frac{1}{2} - \Theta\left(e^{-4\pi^2\sigma^2\|\mathbf{w}\|^2}\right). \tag{3.2}$$

Proof. For the variance, we will use $f_{\mathbf{w}}(\mathbf{t})^2 = \frac{1}{2} + \frac{1}{2}\cos\left(4\pi\left\langle \mathbf{w}, \mathbf{t}\right\rangle\right) = \frac{1}{2} + \frac{1}{2} f_{2\mathbf{w}}(\mathbf{t})$.

(a) Integrating over a fundamental region $\mathbf{B} \cdot [0,1]^n$ shows $\mathbb{E}[f_{\mathbf{w}}(\mathbf{t})] = 0$ since $\int_0^1 \cos(\alpha + 2\pi k x)\mathrm{d}x = 0$ holds for all $k \in \mathbb{Z}$ and $\alpha \in \mathbb{R}$. Then by the above, it readily follows,

$$\mathbb{V}[f_{\mathbf{w}}(\mathbf{t})] = \frac{1}{2} + \frac{1}{2}\,\mathbb{E}[f_{2\mathbf{w}}(\mathbf{t})] = \frac{1}{2}.$$

(b) Because a Gaussian is a radial distribution,

$$\mathbb{E}[f_{\mathbf{w}}\left(\mathbf{t}\right)] = \mathop{\mathbb{E}}_{x \leftarrow \mathcal{N}(0,\sigma^2)}[\cos\left(2\pi x\|\mathbf{w}\|\right)] = \exp\Big(-2\pi^2\sigma^2\|\mathbf{w}\|^2\Big),$$

and

$$\mathbb{V}[f_{\mathbf{w}}\left(\mathbf{t}\right)] = \frac{1}{2} + \frac{1}{2}\,\mathbb{E}[f_{2\mathbf{w}}\left(\mathbf{t}\right)] - \mathbb{E}[f_{\mathbf{w}}\left(\mathbf{t}\right)]^2 = \frac{1}{2} + \frac{1}{2}\varepsilon^4 - \varepsilon^2,$$

where $\varepsilon = \exp(-2\pi^2\sigma^2\|\mathbf{w}\|^2) \in (0,1)$.

□

However, to conclude on the behaviour of the total score $f_{\mathcal{W}}(\mathbf{t})$, one needs to resort to some heuristic. In [LW21], they resort to the following heuristic.

Heuristic 3.5 (independent score heuristic). *For any fixed set $\mathcal{W} \subset \Lambda^\vee$ and for any distribution of targets $\mathbf{t}$ considered in Lemma 3.4, the random variables $(\langle \mathbf{w}, \mathbf{t}\rangle \bmod 1)_{\mathbf{w}\in\mathcal{W}}$ are mutually independent.*

By combining the above heuristic with Heuristic 2.3—which looks fair, given the exponential size of $\mathcal{W}$ in the context of interest—one can model the total score $f_{\mathcal{W}}(\mathbf{t})$ of each type of sample as a Gaussian distribution of centre $|\mathcal{W}| \cdot E$ and variance $|\mathcal{W}| \cdot V$, where E and V are the expectation and variance given by the above Lemma 3.4.

One may then deduce the success probability of distinguishing with the score function using the following lemma.

Lemma 3.6. *Let $X \leftarrow \mathcal{N}(E_X, V_X)$ and $Y \leftarrow \mathcal{N}(E_Y, V_Y)$ be independent Gaussian random variables. Then,*

$$\Pr[X > Y] = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{E_X - E_Y}{\sqrt{2(V_X + V_Y)}}\right). \tag{3.3}$$

Proof. Consider the variable $Z = Y - X$, which is also Gaussian, specifically $Z \sim \mathcal{N}(E_Z, V_Z)$ where $E_Z = E_Y - E_X$ and $V_Z = V_X + V_Y$. The event $X > Y$ is equivalent to $Z < 0$, so the result follows from Equation (2.2). □

This lemma leads Laarhoven and Walter to the following heuristic claim.

Heuristic Claim 3.7 ([LW21, Lem. 9])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a random unit-volume lattice, $r > 0$ and $\mathcal{W} \subset \Lambda^\vee$ a set consisting of the α^n shortest vectors of $\Lambda^\vee$, where $\alpha = \min\{\beta \mid e^2 \ln(\beta) = \beta^2 r^2\}$. Then, we have*

$$\Pr[f_{\mathcal{W}}(\mathbf{t}_{\text{BDD}}) > f_{\mathcal{W}}(\mathbf{t}_{\text{unif}})] \geq \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{1}{\sqrt{2}}\right) \approx 0.84,$$

where $\mathbf{t}_{\text{unif}} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$ and $\mathbf{t}_{\text{BDD}} \leftarrow \mathrm{U}(r\,\mathrm{GH}(n)\mathcal{S}^{n-1})$ are sampled independently.

Heuristic Justification. First, we approximate the BDD sample $\mathbf{t}_{\text{BDD}}$ by a Gaussian distribution of parameter $\sigma = r\,\mathrm{GH}(n)/\sqrt{n}$. According to the Gaussian heuristic, the lengths of the vectors in $\mathcal{W}$ are concentrated around $\alpha \cdot \mathrm{GH}(n)$. By the independent score heuristic and Lemma 3.4, the score function for the BDD sample follows a Gaussian distribution $\mathcal{N}(E_X, V_X)$, where

$$E_X = \alpha^n \exp\left(-\frac{2\pi^2\alpha^2 r^2 \cdot \mathrm{GH}(n)^4}{n}\right) = \alpha^n \exp\left(-\frac{\alpha^2 r^2 \cdot n}{2e^2}\right) = \alpha^{n/2},$$

by construction of α. The variance is $V_X \approx \frac{1}{2} \cdot \alpha^n$.

On the other hand, uniform samples give a score distribution Y following a Gaussian distribution $\mathcal{N}(E_Y, V_Y)$ where $E_Y = 0$ and $V_Y = \alpha^n/2$ by case (a) of Lemma 3.4.

Hence, by Lemma 3.6, the probability of having $X > Y$ equals

$$\frac{1}{2} + \frac{1}{2} \operatorname{erf}\left(\frac{\alpha^{n/2}}{\sqrt{2 \cdot \alpha^n}}\right) \approx 0.84. \qquad \triangle$$

3.3.2 Guo–Johansson and MATZOV

The analysis proposed by Guo–Johansson [GJ21] is somewhat less explicit. Instead of analysing the score directly, they consider the statistical distance between the distribution of $\langle \mathbf{t}, \mathbf{w} \rangle \bmod 1$ for $\mathbf{t}$ uniform and Gaussian.

Using the same independent score heuristic, they then conclude on the statistical distance between the tuples $(\langle \mathbf{t}, \mathbf{w} \rangle \bmod 1)_{\mathbf{w} \in \mathcal{W}}$ for $\mathbf{t}$ uniform and Gaussian. While there exist optimal distinguishers (introduced in [LW21] using a lemma dating back to Neyman–Pearson [NP33]), it differs from the score function $f_{\mathcal{W}}$, but they seem to assume that the scoring function is not that far from optimal. An argument for such a statement is given by Laarhoven and Walter [LW21, Corollary 2], but it is not mentioned by Guo and Johansson [GJ21].

The analysis of MATZOV [MAT22] is on the contrary quite explicit on computing the distribution of scores, while taking into account the severe technical complications introduced by their *modulus switching*. Namely, they increase the number of dual vectors with a factor D_{round} in [MAT22, Section 5] to account for the effect of rounding the Fourier coefficients after performing a modulus switch. Another factor D_{arg} is also introduced, but we view it as dubious, see [DP23a, App. A.4].

In any case, we can essentially recover the key claims of [GJ21, MAT22] directly from the above analysis as well. Note that we express the result more generally in terms of BDD in an arbitrary lattice rather than a specific LWE instance. Moreover, we state this key claim here without loss of generality for a unit-volume, full-rank lattice, because of renormalisation and because the dual vectors used in distinguishing actually come from the dual of a projected sublattice. Indeed, the general setting of the Dual-Sieve attack [ADPS16, EJK20, GJ21, MAT22] first

applies BKZ reduction on the dual lattice $\Lambda^\vee_{\text{LWE}}$ of dimension d, and then runs a sieve on a lower-rank sublattice $\Lambda^\vee \subset \Lambda^\vee_{\text{LWE}}$, see Remark 3.2.

Heuristic Claim 3.8 (Key claim of [GJ21, MAT22], reconstructed)**.** *Let $\Lambda \subset \mathbb{R}^n$ be a random unit-volume lattice, $\mathcal{W} \subset \Lambda^\vee$ the set consisting of the $(4/3)^{n/2}$ shortest vectors of $\Lambda^\vee$, and $\sigma > 0$. Taking $\ell = \sqrt{4/3} \cdot \mathrm{GH}(n)$ and $\varepsilon = \exp(-2\pi^2\sigma^2\ell^2)$, we then have*

$$\Pr[f_{\mathcal{W}}(\mathbf{t}_{\text{BDD}}) > f_{\mathcal{W}}(\mathbf{t}_{\text{unif}})] \geq 1 - e^{-\frac{1}{2}|\mathcal{W}|\varepsilon^2},$$

where $\mathbf{t}_{\text{unif}} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$ and $\mathbf{t}_{\text{BDD}} \leftarrow \mathcal{N}_n(0, \sigma^2)$ are sampled independently.

Heuristic Justification. Similar to the Heuristic Justification of Heuristic Claim 3.7, the score distribution for $\mathbf{t}_{\text{BDD}}$ is approximately $X \sim \mathcal{N}\left(\varepsilon \cdot |\mathcal{W}|, \frac{1}{2}|\mathcal{W}|\right)$, as lengths of vectors in $\mathcal{W}$ are concentrated around $\ell = \sqrt{4/3} \cdot \mathrm{GH}(n)$ by the Gaussian heuristic. The uniform sample $\mathbf{t}_{\text{unif}}$ has a score distribution of approximately $Y \sim \mathcal{N}\left(0, \frac{1}{2}|\mathcal{W}|\right)$. Thus, by Lemma 3.6,

$$\Pr[f_{\mathcal{W}}(\mathbf{t}_{\text{BDD}}) > f_{\mathcal{W}}(\mathbf{t}_{\text{unif}})] = 1 - \frac{1}{2}\operatorname{erfc}\left(\sqrt{|\mathcal{W}|/2} \cdot \varepsilon\right).$$

If $\varepsilon\sqrt{|\mathcal{W}|} \leq 1/\sqrt{2\pi}$, the above probability is trivially at least $\frac{1}{2} \geq 1 - e^{-\frac{1}{2}|\mathcal{W}|\varepsilon^2}$. In the other case, Lemma 2.1 yields the claim:

$$\begin{aligned}\Pr[f_{\mathcal{W}}(\mathbf{t}_{\text{BDD}}) > f_{\mathcal{W}}(\mathbf{t}_{\text{unif}})] &\geq 1 - \frac{1}{\sqrt{2\pi|\mathcal{W}|} \cdot \varepsilon} e^{-\frac{1}{2}|\mathcal{W}|\varepsilon^2} \\ &> 1 - e^{-\frac{1}{2}|\mathcal{W}|\varepsilon^2}\ .\end{aligned}$$ △

An important corollary of the above claim is that one can distinguish one BDD sample from $O(e^{\frac{1}{2}|\mathcal{W}|\varepsilon^2})$ uniform samples with constant probability, by marking the sample that evaluates to the highest score as the BDD sample. In the case of search-BDD, then with constant probability, one can recover the lattice vector $\mathbf{v}$ closest to $\mathbf{t}$ modulo the sparsified lattice Λ', if the sparsification factor is bounded by the above quantity.

3.4 Generalisation of the Dual-Sieve-FFT attack

As established by the literature [Hås88, AR04, LW21], scoring target points to obtain information about their distance to a primal lattice Λ

using short dual vectors is very general, and not limited to LWE lattices. In this section, we will show that this is also the case of the extra FFT trick as proposed by Guo and Johansson [GJ21].

3.4.1 Abstracting Guo–Johansson

The general idea is as follows. Given a lattice Λ, one first crafts a sparsification Λ' of Λ, i.e., a sublattice $\Lambda' \subset \Lambda$ of finite index. This gives rise to a finite abelian group of cosets $G = \Lambda/\Lambda'$. Now, to solve BDD for a target $\mathbf{t} \in \mathbb{R}^n/\Lambda$ on the lattice Λ, one solves BDD for the target $\mathbf{t}$ on all the cosets $\Lambda' + g$, or equivalently, solve BDD for all the targets $\mathbf{t} - g$ with $g + \Lambda' \in G$ on the sublattice Λ'. For the correct choice of coset $g + \Lambda'$, the distance to $\mathbf{t}$ is the same as in the initial BDD problem, but the sublattice is sparser, making this BDD problem easier than the original one. However, we now have $|G|$ instances to consider.

With the help of the DFT, the score function $f_{\mathcal{W}}$ can be computed for all targets $\mathbf{t}-g$ in a batch. That is, applying the DFT_G in Equation (2.10) on a sequence $(\chi_{\mathbf{w}'}(g-\mathbf{t}))_{g\in G}$ for some $\mathbf{w}' \in (\Lambda')^\vee$ gives another sequence that at index $\chi_{\mathbf{w}} \in \widehat{G}$ has a value of,

$$\sum_{g\in G} \chi_{\mathbf{w}'}(g-\mathbf{t})\overline{\chi_{\mathbf{w}}(g)} = \chi_{\mathbf{w}'}(-\mathbf{t}) \cdot \sum_{g\in G} \frac{\chi_{\mathbf{w}'}(g)}{\chi_{\mathbf{w}}(g)}, \tag{3.4}$$

where $\mathbf{w}+\Lambda^\vee \in (\Lambda')^\vee/\Lambda^\vee$ as $\widehat{G}$ is isomorphic to $(\Lambda')^\vee/\Lambda^\vee$ by Lemmata 2.73 and 2.74. Note that $\chi_{\mathbf{w}'}(g)$ is well-defined for $g \in \Lambda/\Lambda'$ as $\chi_{\mathbf{w}'}$ is Λ'-periodic. By the orthogonality of characters, note that

$$\sum_{g\in G} \frac{\chi_{\mathbf{w}'}(g)}{\chi_{\mathbf{w}}(g)} = \begin{cases} |G|, & \text{if } \mathbf{w}' \in \mathbf{w} + \Lambda^\vee, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, Equation (3.4) is zero everywhere except at index $\chi_{\mathbf{w}'}$, where it is equal to $|G| \cdot \chi_{\mathbf{w}'}(-\mathbf{t})$.

By $\mathbb{C}$-linearity of the DFT, one can obtain an expression for the DFT of $f_{\mathcal{W}}(g-\mathbf{t})$ for any finite set of dual vectors $\mathcal{W} \subset (\Lambda')^\vee$. More specifically, if for all $\mathbf{w} \in \mathcal{W}$ we have $-\mathbf{w} \in \mathcal{W}$, i.e., it is *symmetric*, we have

$$\mathrm{DFT}_G\Big((f_{\mathcal{W}}(\mathbf{t}-g))_{g\in G}\Big) = |G| \cdot \Big(\sum_{\mathbf{w}'\in\mathcal{W}\cap(\mathbf{w}+\Lambda^\vee)} f_{\mathbf{w}'}(\mathbf{t})\Big)_{\mathbf{w}+\Lambda^\vee\in\widehat{G}}.$$

Neglecting this scalar $|G|$, we therefore construct a batch of score functions, by performing an inverse FFT on the sequence $\sum_{\mathbf{w}' \in \mathbf{w}+\Lambda^\vee} f_{\mathbf{w}'}(-\mathbf{t})$ for all (dual) cosets $\mathbf{w} + \Lambda^\vee \in (\Lambda')^\vee/\Lambda^\vee$. Then, the entry with $g + \Lambda' \in G$ that has the highest score is most likely the coset $g + \Lambda'$ containing the lattice point in Λ that is closest to $\mathbf{t}$.

3.4.2 Implementation

In this section, we will give a concrete implementation of an algorithm that performs the general Dual-Sieve-FFT attack on a lattice Λ.

Concretely, the lattice Λ is specified by a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, and one can take a simple sparsification such as Λ' being specified by the basis $[d_1\mathbf{b}_1, \ldots, d_n\mathbf{b}_n]$ for suitable $d_1, \ldots, d_n \in \mathbb{N}$.

In fact, any sparsification is, after a basis change, of this shape. When the sublattice $\Lambda' = \mathbf{B}' \cdot \mathbb{Z}^n \subset \mathbf{B} \cdot \mathbb{Z}^n = \Lambda$ is described by a matrix $\mathbf{B}'$, we can express the basis $\mathbf{B}'$ in terms of $\mathbf{B}$, i.e., find the matrix $\mathbf{A} \in \mathbb{Z}^{n \times n}$ such that $\mathbf{B}' = \mathbf{B} \cdot \mathbf{A}$.

Then, put $\mathbf{A}$ in SNF, i.e., find matrices $\mathbf{S}, \mathbf{T} \in \mathrm{GL}_n(\mathbb{Z})$ and a diagonal matrix $\mathbf{D}$ such that $\mathbf{A} = \mathbf{SDT}$ and thus, we have $\mathbf{B}'\mathbf{T}^{-1} = (\mathbf{BS}) \cdot \mathbf{D}$. As $\mathbf{A}$ was full-rank, $\mathbf{D}$ is full-rank and invertible over $\mathbb{Q}$, so here Λ' is described by the basis $[d_1\mathbf{b}'_1, \ldots, d_n\mathbf{b}'_n]$, where $\mathrm{diag}(d_1, \ldots, d_n) = \mathbf{D}$ and $\mathbf{BS} = [\mathbf{b}'_1, \ldots, \mathbf{b}'_n]$ is a basis for Λ. Note that the SNF is efficiently computable [Sto96].

Hence, without loss of generality, we have a sparsification $\Lambda' \subset \Lambda$, where $\mathbf{B}$ is a basis for Λ and $\mathbf{B}' = [d_1\mathbf{b}_1, \ldots, d_n\mathbf{b}_n]$ is a basis for Λ'. Then Algorithm 3.1 will find the coset of Λ' that is most likely to contain a lattice vector closest to target $\mathbf{t}$.

Structure of the quotient group

From a geometric perspective, concerning the length of the vectors in $\mathcal{W}$, the structure of the group $G = \Lambda/\Lambda'$ does not appear to matter at all, only its size does. On the other hand, while asymptotically all group structures allow computing DFT_G in time $O(|G| \log |G|)$, the structure of the group matters quite a lot in practice and the case $G = (\mathbb{Z}/2\mathbb{Z})^k$, i.e., the Walsh–Hadamard Transform, should definitely be the best choice. That is, one should construct the sublattice Λ' as generated by $\mathbf{B}' = [2\mathbf{b}_1, \ldots, 2\mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n]$, which has index 2^k in Λ.

Algorithm 3.1. DualFFT($\mathbf{B}, \mathbf{B}', \mathcal{W}, \mathbf{t}$): Dual-Sieve-FFT attack

Input: two bases $\mathbf{B}', \mathbf{B}$ for full-rank lattices $\Lambda' \subset \Lambda \subset \mathbb{R}^n$, a set of short dual vectors $\mathcal{W} \subset (\Lambda')^\vee$, and a target $\mathbf{t} \in \mathbb{R}^n/\Lambda'$ such that $\mathbf{B}' = \mathbf{B} \cdot \mathrm{diag}(d_1, \ldots, d_n)$ holds for certain $d_1, \ldots, d_n \in \mathbb{Z}_{\geq 1}$
Output: the lattice coset $\mathbf{g} + \Lambda' \in \Lambda/\Lambda'$ closest to $\mathbf{t}$

1: initialise a table T of dimension $d_1 \times d_2 \times \cdots \times d_n$ with zeros
2: **for** $\mathbf{w} \in \mathcal{W}$ **do**
3: compute $j_i \in \{0, 1, \ldots, d_i - 1\}$ s.t. $\mathbf{w} \in \frac{j_1}{d_1}\mathbf{b}_1^\vee + \cdots + \frac{j_n}{d_n}\mathbf{b}_n^\vee + \Lambda^\vee$
4: $\mathsf{T}[j_1, j_2, \ldots, j_n] \leftarrow \mathsf{T}[j_1, j_2, \ldots, j_n] + \cos(2\pi \langle \mathbf{w}, \mathbf{t} \rangle)$
5: $\mathsf{S} = \mathrm{DFT}^{-1}_{\mathbb{Z}/d_1\mathbb{Z} \times \cdots \times \mathbb{Z}/d_n\mathbb{Z}}(\mathsf{T})$
6: $(j_1, j_2, \ldots, j_n) \leftarrow \underset{\substack{k_1, k_2, \ldots, k_n \\ 0 \leq k_i < d_i}}{\arg\max} \{\mathsf{S}[k_1, k_2, \ldots, k_n]\}$
7: **return** $j_1\mathbf{b}_1 + \cdots + j_n\mathbf{b}_n$

Randomised sparsification

Note that the analysis of the length of the vectors in $\mathcal{W}$ requires applying Gaussian heuristic to the densification of the dual, induced by the sparsification of the primal.

This might require care. Indeed, if the basis is well reduced before we apply the dual densification $\mathrm{diag}\left(\frac{1}{d_1}, \ldots, \frac{1}{d_n}\right)$, this might create a dual lattice which is not random-looking; in particular it might contain a few vectors shorter than predicted by Gaussian heuristic, with an unclear impact on the rest of $\mathcal{W}$.

We do not expect this to be an issue if the basis $\mathbf{B}$ is adequately randomised before constructing the sparsification.

3.4.3 Advantages

Not only is it theoretically more satisfying to apply the FFT trick to the general decoding problem rather than to the specific LWE problem, it also makes recursion more straightforward.

Shorter dual vectors

The algorithm of Guo and Johansson [GJ21] seems to perfectly fit this formalisation, where the basis $\mathbf{B}$ is the standard basis associated with the q-ary representation of the lattice, and $\mathbf{B}' = \mathbf{B} \cdot \mathrm{diag}(\gamma, \ldots, \gamma, 1, \ldots 1)$

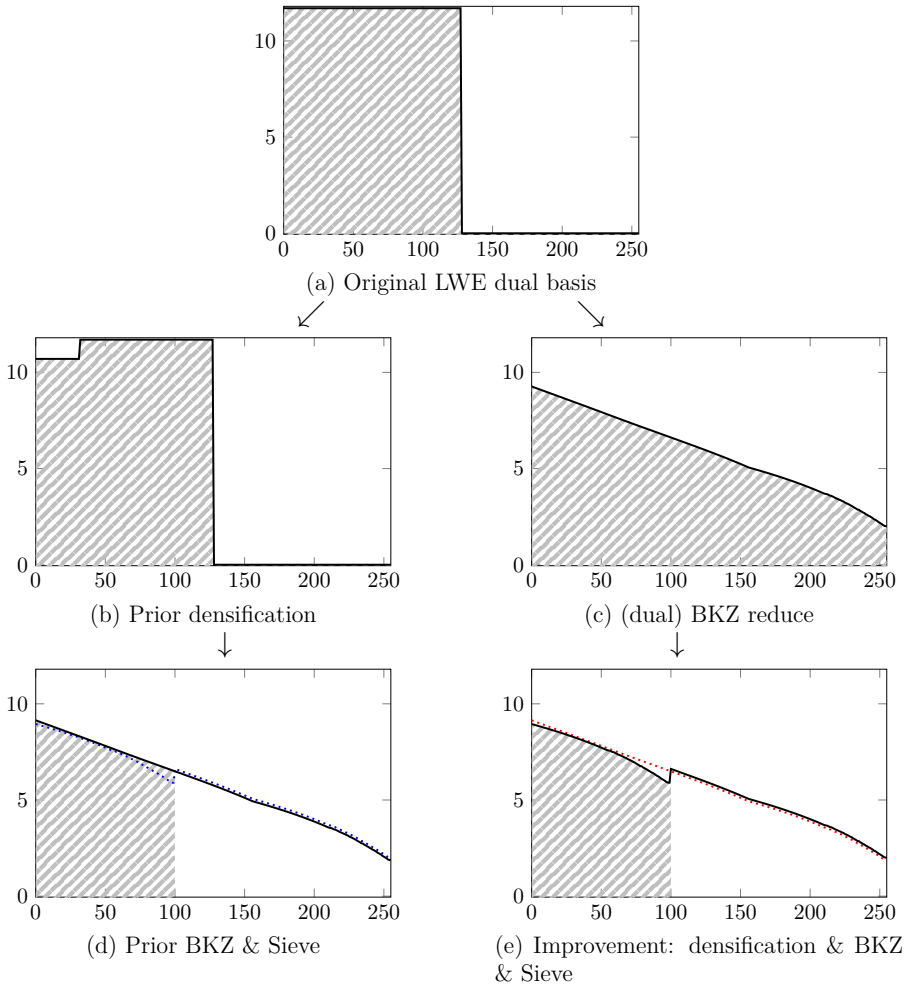


Figure 3.1: Simulation of dual basis profile, i.e., $(\log_2\|\mathbf{d}_i^*\|)_{i=0}^{n-1}$ as a function of the basis index i using the BKZ 2.0 simulator [CN11]. Used parameters are $n = 128, \gamma = 2, k = 32, \beta_{\mathrm{BKZ}} = \beta_{\mathrm{sieve}} = 100, q = 3329$. Prior work follows the path (a) → (b) → (d), while the suggested improvement follows the path (a) → (c) → (e). The filled regions in (d) and (e) represent the log volume of the sublattice used for sieving, and the dotted line shows (e) and (d) respectively for comparison. Note that the densification in (e) shortens $({}^{\vee}\mathbf{B})_{[\beta-k,\beta-1]}$ by a factor γ. See paragraph § Randomised sparsification (p. 99) regarding the lengths of dual vectors in $\mathcal{W}$.

with k many γ's. The set of short dual vectors is obtained by first running BKZ reduction with block size β_{BKZ} on the dual of $\mathbf{B}'$, and then sieving in the sublattice generated by the first β_{sieve} vectors of this reduced dual basis. The impact of the sparsification $\mathbf{B}' = \mathbf{B} \cdot \mathrm{diag}(\gamma, \ldots, \gamma, 1, \ldots 1)$ on the length of the vectors in $\mathcal{W}$ is that these are shortened by a factor $\gamma^{k/n}$. That is, the sparsification has been *diluted* over n dimensions.

Instead, consider first applying dual-BKZ reduction with block size β_{BKZ} on $\mathbf{B}$, and then taking $\mathbf{B}' = \mathbf{B} \cdot \mathrm{diag}(1, \ldots 1, \gamma, \ldots, \gamma)$. In this way, the densification of the reversed dual basis is given by ${}^{\vee}(\mathbf{B}') = {}^{\vee}\mathbf{B} \cdot \mathrm{diag}\left(\frac{1}{\gamma}, \ldots, \frac{1}{\gamma}, 1, \ldots 1\right)$ remains concentrated on the k first dual vectors![1] Therefore, the sparsification now makes the vectors in $\mathcal{W}$ shorter by a factor $\gamma^{k/\beta_{\mathrm{sieve}}}$ (assuming $k \leq \beta_{\mathrm{sieve}}$).

We leave the concrete exploitation of this improvement to the Dual-Sieve-FFT attack as future work, because the next section will invalidate the analysis of [GJ21, MAT22], so we first need a fixed analysis before thinking about this possible improvement. A visualisation of this improvement can be found in Figure 3.1.

Remark 3.9. The algorithm of MATZOV [MAT22] differs quite a bit from that of Guo–Johansson [GJ21] by resorting to a modulus switching technique, and it is claimed that this technique allows to decrease dimension at the cost of some extra error. We remain rather circumspect regarding a perceived superiority of the variant of MATZOV [MAT22] over that of Guo–Johansson [GJ21]. For more details, see [DP23a, App. A.6].

3.5 Contradictions to the independent score heuristic

In this section, we consider two regimes, in which we show that the analyses of [LW21, GJ21, MAT22] give rise to absurd conclusions when using the independent score heuristic. Both regimes are concerned with distinguishing between a BDD target and a uniform target.

In the first regime, we consider BDD samples where the expected distance to the lattice exceeds the Gaussian heuristic, a task that was recently proved statistically impossible in a random lattice [DADRT23].

The second regime is concerned with the case of finding a planted solution among many candidates. For certain parameters, the analysis

[1] The warning on randomised sparsification from Section 3.4.2 still applies here; the basis randomisation should be applied to the block $\mathbf{B}_{[n-\beta_{\mathrm{sieve}}+1,n]}$.

of [GJ21, MAT22] predicts a successful guess of the desired target, despite the existence of many other candidates that are even closer to the lattice than the planted solution. We would like to stress that this contradiction, regardless whether the FFT trick is used or not, merely arises from the large number of candidates. Phrased differently, the works of [GJ21, MAT22] predict a much lower failure probability on distinguishing between a BDD and uniform target than what one can reasonably assume.

Recall that in this section, as in [LW21], we implicitly renormalise the lattice Λ to have volume 1.

3.5.1 Distinguishing the indistinguishable

Set up

Recall that the main result of Laarhoven and Walter [LW21, Lem. 9], reformulated as Heuristic Claim 3.7, provides an algorithm to distinguish between a BDD instance at distance $r \cdot \mathrm{GH}(n)$, and a uniform target modulo the lattice. After a precomputation depending solely on the lattice Λ, this algorithm has exponential complexity α^n, where α satisfies $e^2 \ln(\alpha) = \alpha^2 r^2$, and the heuristic analysis claims that it is successful with constant probability. That is, the algorithm outputs 'BDD' with constant probability if its input is a sampled BDD instance $\mathbf{t}$, and it outputs 'uniform' with constant probability if its input $\mathbf{t}$ is sampled uniformly modulo the lattice.[2]

Lemma 3.10. *The equation $e^2 \ln(x) = x^2 r^2$ admits a real solution in x if and only if $r^2 \leq e/2$.*

Proof. The statement and its proof are illustrated in Figure 3.2. First, note that $x \mapsto e^2 \ln(x)$ is concave, while $x \mapsto x^2 r^2$ is convex for any $r \in \mathbb{R}$. We discuss three cases.

Case 1: $r^2 = e/2$.

The parabola $y = r^2 x^2$ intersects tangentially the curve $y = e^2 \ln x$

[2]Note that this algorithm gets only the sample $\mathbf{t}$ as input, whereas Heuristic Claim 3.7 considers an algorithm that gets as input $\mathbf{t}_{\mathrm{BDD}}$ *and* $\mathbf{t}_{\mathrm{unif}}$ without knowing which is which. By saying 'BDD' if the score for $\mathbf{t}$ is at least the threshold value $\frac{1}{2}\,\mathbb{E}[f_{\mathcal{W}}(\mathbf{t}_{\mathrm{BDD}})]$ and 'uniform' otherwise, one derives a success probability of $\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}(\frac{1}{2}) \approx 0.76$.

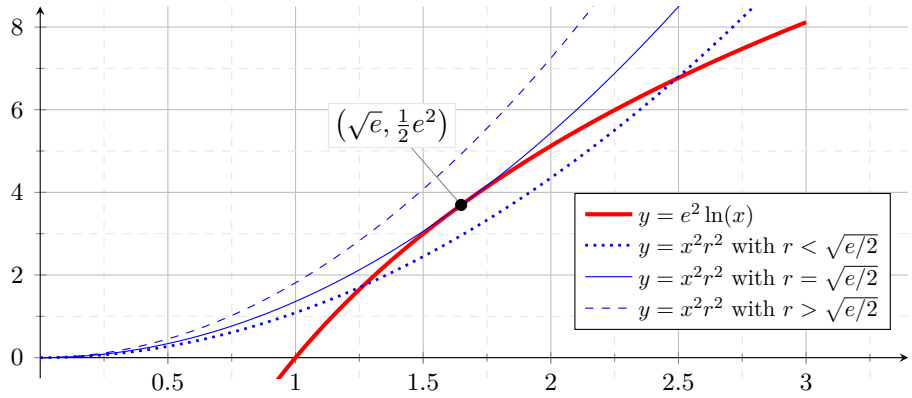


Figure 3.2: Equation $e^2 \ln(x) = x^2r^2$ for various r

at $(x, y) = (\sqrt{e}, \frac{1}{2}e^2)$, with slope $\frac{dy}{dx}\Big|_{x=\sqrt{e}} = e^{3/2}$. By convexity, we have $e^2 \ln(x) < x^2r^2$ for any $x \neq \sqrt{e}$.

Case 2: $r^2 > e/2$.
Note that r^2x^2 is strictly increasing in r^2 for any x. Reusing the first case, we see $e^2 \ln(x) < x^2r^2$ holds for all x, so there are no solutions in this case.

Case 3: $r^2 < e/2$.
We have $e^2 \ln(x) > x^2r^2$ at $x = \sqrt{e}$. We also have $e^2 \ln(x) < x^2r^2$ when $x = 1$. The Intermediate Value Theorem then tells there is a solution with $x \in (1, \sqrt{e})$. □

This lemma suggests that the algorithm works successful supposedly for all $r < \sqrt{e/2} \approx 1.1658$. It is suspicious that the algorithm works beyond $r > 1$ because a uniform target often has a lattice point within a distance of $r\,\mathrm{GH}(n)$. Namely, every lattice Λ satisfies by Proposition 2.11,

$$\mathop{\mathbb{E}}_{\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)} \left[\left| \left\{ \mathbf{v} \in \Lambda \;\middle|\; \|\mathbf{v} - \mathbf{t}\| \leq r\,\mathrm{GH}(n) \right\} \right| \right] = r^n \ .$$

Thus, a nearby lattice point is guaranteed for a BDD target and likely for a uniform target, making distinguishing hard.

Still, one could imagine a scenario where with small probability a target has few close vectors, but most likely it will not, making distinguishing statistically possible.

The contradiction

It has been shown recently by Debris-Alazard et al. [DADRT23] that the above scenario does not occur with random lattices. More specifically, for a formally defined notion of random lattices and a fixed $r > 1$, it is proved that for errors from a uniform distribution on the ball of radius $r\,\mathrm{GH}(n)$, the statistical distance between the error modulo the lattice and $\mathrm{U}(\mathbb{R}^n/\Lambda)$ is exponentially small as a function of the dimension [DADRT23, Prop. 4.3].

That is, for all $r > 1$, no algorithm, whatever its complexity, can even succeed with probability greater than $\frac{1}{2} + O(1) \cdot r^{-n/2}$. Yet, [LW21, Lem. 9] (reformulated as Heuristic Claim 3.7) claims a constant success probability. ↯

Discussion

One could counter-argue that the claim [LW21, Lem. 9] is given for a uniform distribution on a sphere, a case not contradicted by Debris-Alazard et al. [DADRT23]. However, the actual analysis in [LW21] is done for a Gaussian distribution, a case which *is* also covered by Debris-Alazard et al. [DADRT23, Thm. 4.6].

The suspect heuristic

We note that this counter-argument applies only to the heuristic [LW21, Lem. 9], that is given a single sample, and not to [LW21, Lem. 8]. Indeed, in the context of the heuristic [LW21, Lem. 8] where exponentially many samples (either all uniform or all BDD) are given, the exponentially small statistical distance can be compensated for using numerous samples, as discussed between both Lemmata.

We note in particular that [LW21, Lem. 8] does not require the independent score heuristic, as it uses only one dual vector. In fact, after close inspection of the reasoning behind [LW21, Lem. 8], we could not identify any step that should be too hard to make formally provable, up to minor conditions and small losses in the concrete efficiency of the distinguisher. Indeed, with enough effort, it appears that all the other heuristics and approximations could be dealt with formally.

This sets the independent score heuristic as the prime suspect leading to the erroneous [LW21, Lem. 9].

3.5.2 Candidates closer than the solution (asymptotic)

Set up

Recall that the key claim of [GJ21] and [MAT22], reconstructed as Heuristic Claim 3.8 considers the case where the set of dual vectors comes from a lattice sieve [NV08, MV10, BDGL16], i.e., it consists of $N = (4/3)^{n/2}$ dual vectors of length $\ell = \sqrt{4/3}\cdot\mathrm{GH}(n)$. Given one uniform sample and one BDD sample with an error sampled from a Gaussian distribution of parameter σ, it was claimed that one can distinguish with a failure probability that is exponentially small in $N\varepsilon^2$, whenever $N \geq 1/\varepsilon^2$, where $\varepsilon = \exp(-2\pi^2\sigma^2\ell^2)$. Let us consider the constraint the other way around by asking the following question.

Question 3.11. How large can one take σ to have a failure probability of at most p_{fail}?

It can be seen that the failure probability in Heuristic Claim 3.8 is at most p_{fail} when $\varepsilon \geq \sqrt{\frac{\ln(1/p_{\mathrm{fail}}^2)}{N}}$. This constrains σ to satisfy

$$\sigma = \sqrt{\frac{\ln(1/\varepsilon)}{2\pi^2\ell^2}} \leq \frac{\sqrt{\ln N - \ln\ln(1/p_{\mathrm{fail}}^2)}}{2\pi\ell} = \frac{\sqrt{\frac{n}{2}\ln\frac{4}{3} - \ln\ln(1/p_{\mathrm{fail}}^2)}}{2\pi\sqrt{\frac{4}{3}}\cdot\mathrm{GH}(n)}. \tag{3.5}$$

With the approximation $\mathrm{GH}(n) \approx \sqrt{\frac{n}{2\pi e}}$, one then arrives at $\sigma \leq \sqrt{C - \frac{C'\ln\ln(1/p_{\mathrm{fail}}^2)}{n}}$ for some constants $C = \frac{3e\ln(4/3)}{16\pi} \approx 0.047$ and $C' = \frac{3e}{8\pi} \approx 0.32$. This means that Heuristic Claim 3.8 supposedly still distinguishes a BDD sample with an expected distance of $\sqrt{C\cdot n} \approx 0.89\ \mathrm{GH}(n)$ from a uniform sample with a failure probability that is *double-exponentially small*: $p_{\mathrm{fail}} = \exp(-\frac{1}{2}\exp(n^{.99}))$.

The contradiction

We will show that the above heuristic claim leads to a contradiction, already for a single exponentially small failure probability of $p_{\mathrm{fail}} = (0.48)^n$.

Lemma 3.12. *Let Λ be a unit-volume lattice, and $r > 0$ such that $r < \frac{\lambda_1(\Lambda)}{2\ \mathrm{GH}(n)}$. Then, for a target $\mathbf{t}$ uniform in $\mathbb{R}^n/\Lambda$, it holds with probability r^n that $\mathbf{t}$ is at distance at most $r\ \mathrm{GH}(n)$ from the lattice.*

Proof. Note that the volume of the ball of radius $r \cdot \mathrm{GH}(n)$ is exactly r^n by definition of $\mathrm{GH}(n)$. Furthermore, because $r \cdot \mathrm{GH}(n) < \lambda_1(\Lambda)/2$, all translations of this ball by points in Λ are disjoint. Said otherwise, this ball does not intersect itself modulo the lattice. More formally, its projection onto the torus $\mathbb{R}^n/\Lambda$ is injective. Hence, the ball modulo the lattice also has volume r^n in $\mathbb{R}^n/\Lambda$. The probability of a uniform target $\mathbf{t}$ falling into that ball is therefore $r^n / \operatorname{Vol}_n(\mathbb{R}^n/\Lambda) = r^n / \det(\Lambda) = r^n$. □

Let us use this lemma in the case of a random unit-volume lattice, or more specifically, one where we expect $\lambda_1(\Lambda) \approx \mathrm{GH}(n)$. Using Lemma 3.12 with $r = 0.49$, the probability that $\mathbf{t}_{\mathrm{unif}}$ lies in the ball of radius r $\mathrm{GH}(n)$ is r^n. Assume this event happens. In this case, the uniform target lies at a distance r $\mathrm{GH}(n)$ from the lattice, which is much shorter than the distance of 0.89 $\mathrm{GH}(n)$ the BDD sample was away from the lattice. However, the score function $f_{\mathcal{W}}$ is precisely meant to associate a larger score to closer targets, so in this case we expect to have $f_{\mathcal{W}}(\mathbf{t}_{\mathrm{unif}}) > f_{\mathcal{W}}(\mathbf{t}_{\mathrm{BDD}})$, with at least a constant probability. Thus, in fact the failure probability of distinguishing is close to 0.49^n, which is much more likely than the claimed 0.48^n. ↯

Discussion

One might counter-argue that $f_{\mathcal{W}}$ only probabilistically classifies vectors by their distance to the lattice and might somehow still give the particularly close uniform sample a lower score than the BDD sample. However, the probability that the uniform target $\mathbf{t}$ lies at distance $\mathrm{GH}(n)/n^2 = \Theta(n^{-3/2})$ away from the lattice, is n^{-2n}, which is (only) super-exponentially small. In this case we have $\langle \mathbf{t}, \mathbf{w} \rangle \leq O(1/n)$ for any $\mathbf{w} \in \mathcal{W}$ output by a sieve, and approximating the cosine we know the score $f_{\mathcal{W}}(\mathbf{t})$ for this target $\mathbf{t}$ will be at least $N(1 - O(1/n^2))$, which is essentially maximal.

The suspect heuristic

The discussion above points to the same suspect as Section 3.5.1, namely, the independent score heuristic. Indeed, under independence the probability of one uniform target reaching a constant fraction of the maximal score N should decrease as fast as $\exp(-\Theta(N))$, but we have shown that this probability is in fact at least $\exp(-\Theta(n \log n))$. Independence cannot

hold when N grows asymptotically at least as fast as $\omega(n \log n)$, and this should be visible in the tail of the score distribution for uniform targets.

3.5.3 Candidates closer than the solution (concrete)

Set up

In the contradiction above, we choose p_{fail} as small as 0.48^n to be able to invoke Lemma 3.12 to have the uniform sample at distance r GH$(n) < \frac{1}{2}\lambda_1(\Lambda)$, quantifying the probability that a random target in $\mathbb{R}^n/\Lambda$ falls close to the lattice. This leads to an *extreme contradiction*: we analysed the probability that the uniform target $\mathbf{t}_{\text{unif}}$ is at a distance 0.49 GH(n) from the lattice, much closer than $\mathbf{t}_{\text{BDD}}$ at distance 0.89 GH(n). However, even when there is a uniform sample at distance e.g. 0.88 GH(n) from the lattice, $\mathbf{t}_{\text{unif}}$ can get a higher score than the $\mathbf{t}_{\text{BDD}}$.

To extend Lemma 3.12 up to radii $r < 1$, we will resort to a heuristic instead. In this regime, translations of the ball by lattice points may start to intersect. In practice, however, the volume of this intersection remains rather small and should not affect the volume so much.

Heuristic Claim 3.13. *Let Λ be a random unit-volume lattice, and $r \in (0, 1)$. For a target $\mathbf{t}$ sampled uniformly in the torus $\mathbb{R}^n/\Lambda$, we have a probability of $r^n \cdot \left(1 - n^{O(1)} r^n\right)$ that $\mathbf{t}$ is at distance at most r GH(n) from the lattice.*

Heuristic Justification. Contrary to the proof of Lemma 3.12, we now have to subtract from r^n the probability that a target $\mathbf{t}$ has at least two lattice points at distance less than r GH(n) away, which by Heuristic Claim 2.48 happens with probability $O(n\sqrt{n})r^{2n}$. △

The contradiction

The idea is to arrive at a contradiction whenever we instantiate the claim from above with the largest possible $r > 0$ up to which it is more likely to have a uniform target within r GH(n) away from the lattice than the heuristically claimed failure probability p_{fail}.

For a given dimension n and relative distance $r \in (0, 1)$, Heuristic Claim 3.13 lower bounds the probability that we fail to distinguish a BDD sample from a uniform sample. Indeed, when the uniform sample happens to be closer to the lattice than the BDD sample, with constant

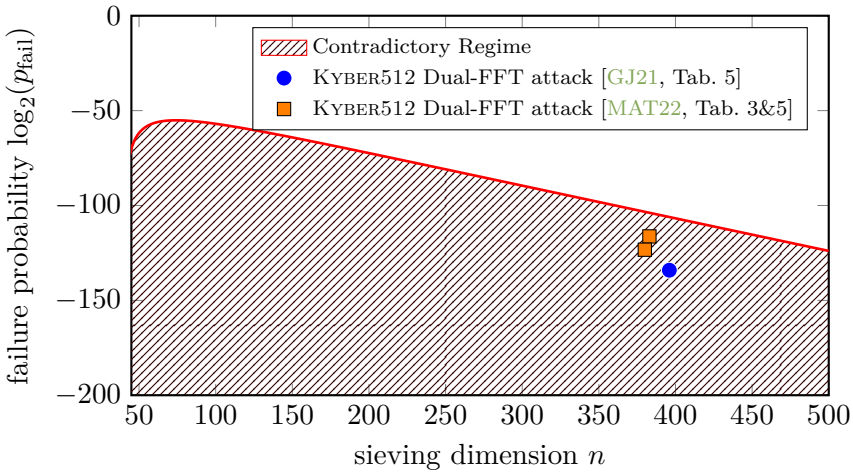


Figure 3.3: The concrete contradictory regime, in which Heuristic Claim 3.8 *overpredicts* the success probability of distinguishing in dimension n, compared to a *simple* lower bound derived from Heuristic Claim 3.13. By determining the largest $\sigma \in [0, \mathrm{GH}(n)/\sqrt{n}]$ for which the contradiction arises, we know the upper boundary of the contradictory regime by computing the probability for that σ. For any pair (n, p_{fail}) falling in the concrete contradictory regime, a distinguisher cannot distinguish, with failure probability at most p_{fail}, the uniform distribution from a BDD distribution having σ satisfy Equation (3.5). Data was obtained with the script `volumetric_contr.py`.

probability a higher score will be assigned to it. Now by identifying all the $r \in (0, 1)$ for which the lower bound is *bigger* than the upper bound from Heuristic Claim 3.8, we have found a regime in which the two heuristics are in clear contradiction with each other. We will call this regime the '*concrete contradictory regime*', and it is depicted in Figure 3.3. ↯

Contradictory regime in the context of concrete attacks against LWE

Above, we have determined the contradictory regime when obtaining the set $\mathcal{W}$ by a sieve over the full lattice, and assuming its volume was 1. It is not hard to see that scaling the lattice up or down is not going to affect the conclusion. Indeed, the Gaussian heuristic of the primal, σ and r will scale with the lattice, while the length ℓ of the dual vectors

will scale inversely; this leaves ε and p unaffected.

In the context of the cryptanalytic literature [EJK20, GJ21, MAT22], the set $\mathcal{W}$ does not come from the full dual lattice $\Lambda^\vee_{\mathrm{LWE}} \subset \mathbb{R}^n$, but comes from a dual lattice $L \subset \Lambda^\vee_{\mathrm{LWE}}$ of rank β_{sieve}, by Remark 3.2. Now, targets are projected onto the β_{sieve}-dimensional space spanned by all $\mathcal{W}$. Hence, we effectively run the distinguisher here over a projected sublattice of dimension β_{sieve}.

In this scenario, the contradictory regime is solely determined by β_{sieve} and the needed failure probability p_{fail}, and not by other quantities such as the LWE parameters and β_{BKZ}. Indeed, the LWE parameters and β_{BKZ} are going to influence the volume of the lattice on which we run the final sieve to obtain $\mathcal{W}$.

This might not perfectly be representative of the exact analysis of MATZOV [MAT22] in that we do not make a special analysis of the modulus switching effect on the score distribution. Instead, this treats modulus switching as adding an implicit error, increasing σ. This remains a strong signal on the credibility of the heuristic analysis in that regime.

Another point raising discussion is the fact that our contradiction is established in the case where the uniform targets $\mathbf{t}$ are independent. This is not formally the case when those targets come from a partial enumeration, though such a heuristic has been used in the past, for example underlying the analysis of the hybrid attack [How07]. More critically, we see no mention of such dependence and how they would affect the algorithm in the existing analysis [EJK20, GJ21, MAT22]. While we do not claim that it is impossible, we view the notion that such dependencies could fix the algorithm as quite doubtful, and requiring specific substantiation with analysis and experiments.

In other words, while our contradiction does not formally disprove the recent claims on the Dual-Sieve attack [EJK20, GJ21, MAT22], it does invalidate the reasoning leading to these claims. Because we do not see an obvious reason why this or that detail would solve the issues raised here, it seems reasonable to presume these claims are incorrect.

The parameters of Guo–Johansson and MATZOV

We now turn to two concrete instantiations of the dual attack [GJ21, MAT22] against the Kyber512 [SAB+22] parameter set with a focus towards the 'asymptotic model' [MAT22, Assumption 7.2]. This 'asymptotic model' is an optimistic estimate of the number of dimensions for

free [Duc18], whereas the other used 'G6K model' [GJ21, MAT22] is debated in [DP23a, App. A.2].

In [GJ21, Table 5], we find a sieving dimension $\beta_{\text{sieve}} = 396$ (where all the dual vectors come from), a guessing dimension $t_1 = 20$, and an FFT dimension $t = 78$. The guessing part considers all 7 possible values $\{-3, \ldots, 3\}$ of each coordinate, while the FFT is done with $\gamma = 2$, giving rise to $T = 7^{20} \cdot 2^{78} \approx 2^{134.1}$ targets of which all are uniform samples except for one.

In [MAT22, Table 3], we find a sieving dimension $\beta_{\text{sieve}} = 380$ (where all the dual vectors come from), a guessing dimension $k_{\text{enum}} = 19$, and an FFT dimension $k_{\text{fft}} = 34$. The guessing part enumerates over $\{-3, \ldots, 3\}^{k_{\text{enum}}}$ in order of decreasing probability from the used binomial distribution, while the FFT is done with $p = 5$, giving rise to, according to [MAT22], $T = 2^{19 \cdot H(\chi_s)} \cdot 5^{34} \approx 2^{123.3}$ many targets. Using an improved cost metric, they also give [MAT22, Table 5] another set of parameter where $\beta_{\text{sieve}} = 383$ and $T = 2^{17 \cdot H(\chi_s)} \cdot 5^{33} \approx 2^{116.3}$.[3]

To get a constant success probability in the dual attack, the distinguisher needs to have a failure probability p_{fail} of the order $1/T$: taking $p_{\text{fail}} = 1/T$ yields a total success probability of $(1 - p_{\text{fail}})^T \geq e^{-p_{\text{fail}} T} = 1/e$ of distinguishing one BDD sample from T (independent) uniform samples.

For both instantiations [GJ21, MAT22] the dual attack is used rather deep in its contradictory regime, as depicted in Figure 3.3. ↯

3.6 Experiments

In this section, we provide further substantiation of the concrete contradiction (Sec. 3.5.3) with experimental evidence. We hope that the experiments will provide insight on what exactly is the issue with the prior analysis, and will show how the issue with the analysis can be resolved. In our analysis, we focus on the case where $\mathcal{W}$ is the output of a full sieve with a saturation radius of $r_{\text{sat}} = \sqrt{4/3}$ and a saturation ratio of $f_{\text{sat}} = 0.90$.

We look at two distributions: the score for uniform targets, and the score for BDD targets with a Gaussian error. There are two plausible

[3] We are not quite sure how this quantity was derived, and it seems incorrect. See [DP23a, App. A.5].

diagnoses for how the contradictory regime appears: the BDD scores are smaller than predicted, or the uniform scores are higher than predicted.

Because the concrete contradictory regime of Figure 3.3 starts when p_{fail} is very small, like 2^{-55} in dimension 70, these unpredicted high scores might be very rare. However, these events are important to determine the tail of the uniform score distribution, so the experiments require numerous samples. Naïvely, it would take a long time to run such large scale experiments, but the same FFT trick from Section 3.4 makes it feasible to run experiments on this scale!

3.6.1 Implementation details

We used the G6K software [ADH+19] for running the experiments, using Python on a high-level with a binding to some C code for computing the WHT. For the uniform targets we wrote the script `unif_scores.py`, which computes scores for many points sampled uniformly from $(\mathbb{Z}/q\mathbb{Z})^n$, where $\mathcal{W}$ is the output of a full sieve on the dual of

$$\mathbf{B} \cdot \operatorname{diag}(2, \ldots, 2, 1, \ldots, 1),$$

with the number of twos equal to the FFT dimension. This setup allowed us to get roughly 2^{25} samples per second per CPU core. The scores were stored in buckets of width 1 while the exceptionally high scores were kept in a list. Here, we sieve using the built-in dual mode of G6K, which works only implicitly with the dual basis, see https://github.com/fplll/fplll/wiki/FPLLL-Days-5-Summary.

In addition, BDD scores are obtained using the script `bdd_scores.py`. The script randomly samples a q-ary lattice for the dual lattice $\Lambda^\vee$, so the script did not use the dual mode in G6K, making the implementation a little bit easier. Then, it computed the score function for the BDD distribution being a Gaussian distribution of parameter $\sigma = f_{\text{GH}} \cdot \text{GH}(n)/\sqrt{nq}$ for some parameter $f_{\text{GH}} \in (0, 1]$, where $1/\sqrt{q}$ is a normalisation factor needed because the dual lattice has determinant $q^{n/2}$.

For the uniform scores, we extracted the $2N = f_{\text{sat}} r_{\text{sat}}^n$ shortest dual vectors from the database. Because G6K returns at most one out of $\{\mathbf{w}, -\mathbf{w}\}$, G6K provides only N dual vectors of length below $r_{\text{sat}} \cdot \text{GH}(n)$. The other N dual vectors that were extracted, are therefore slightly larger than $r_{\text{sat}} \cdot \text{GH}(n)$, and the whole database is rather concentrated around this length. This does *not* affect our conclusions, as the prediction under

the heuristic analysis plotted in Figure 3.4 does not depend on the length of dual vectors.

Remark 3.14. This halving of the number of short vectors, which is also in Heuristic 2.114, was missed in the prior works of [GJ21, MAT22] leading to some irrelevant complication in the analysis of [MAT22], as discussed in [DP23a, App. A.5].

3.6.2 Uniform targets

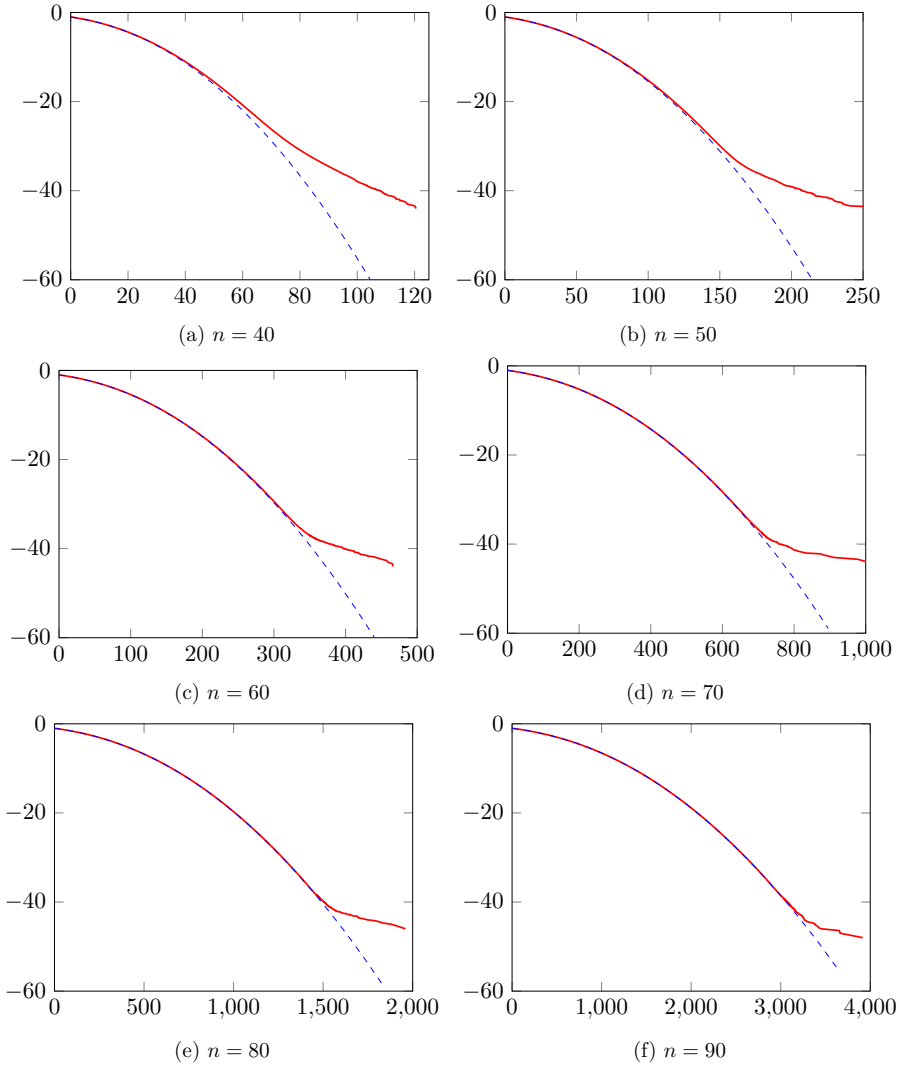
We measured the score distribution for uniform targets over lattices of various dimension, and plotted our result in Figure 3.4. On each of these curves, we see a clear deviation from prediction for rare events: large scores are more likely to occur than predicted. After following a *waterfall* shape, i.e., a quadratic decay in logarithmic scale, the score probability seems to reach a *floor*, where it decays much slower than a normal distribution predicts. This is perfectly in accordance with the contradiction discussed in Sections 3.5.2 and 3.5.3: we start encountering vectors that are quite close to the lattice, which should have a rather high score.

Conclusion

Comparing Figure 3.3 with the floors appearing in Figure 3.4, we conclude that the floor seems to appear earlier than expected by the contradictory regime, vindicating the notion that the analysis might fail even in an earlier regime than our predicted contradiction. As we will see in the next section, the floor behaviour is not the only thing that the independent score heuristic predicts incorrectly.

3.6.3 BDD targets

We measured the score distribution for BDD targets sampled from a Gaussian distribution of parameter $\sigma = 0.7 \cdot \mathrm{GH}(n)/\sqrt{n}$, over lattices of various dimensions n, and plotted our result in Figure 3.5. The predicted score distribution is based on the independent score heuristic, but also takes into account the exact lengths of each dual vector in Equation (3.2), instead of approximating all the lengths to be equal to $\sqrt{4/3} \cdot \mathrm{GH}(n)$, to make the prediction more accurate.

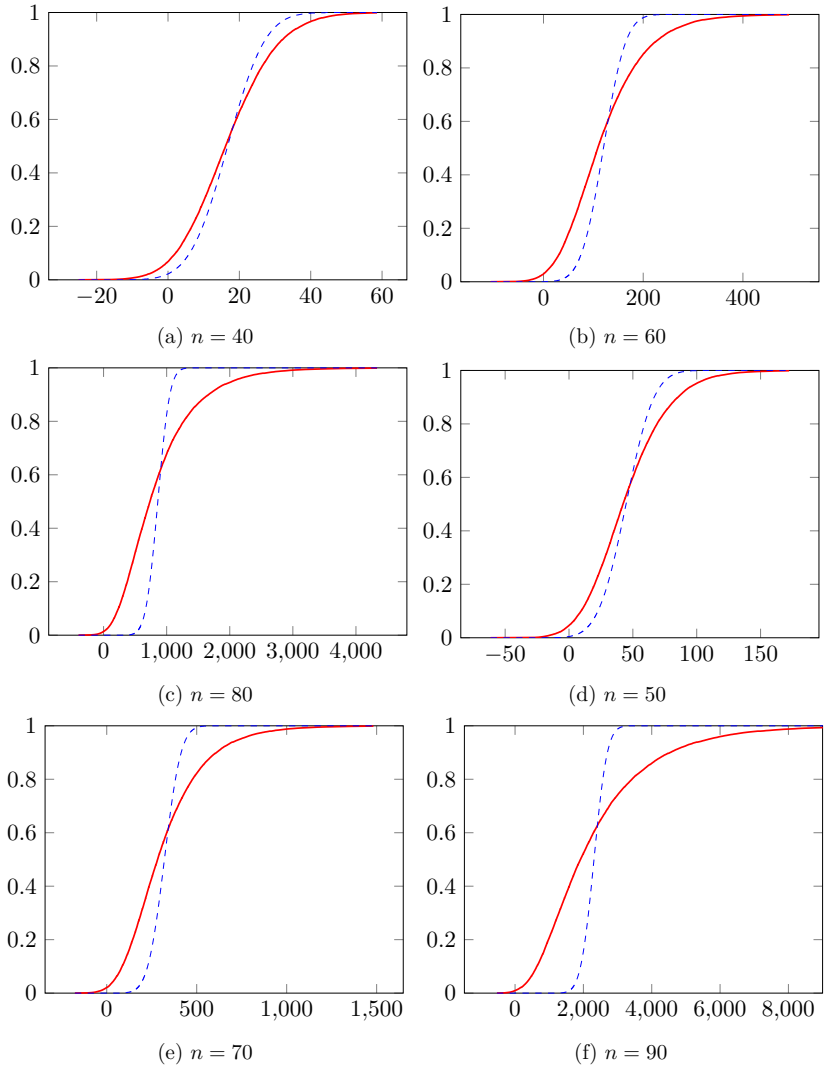


(a) $n = 40$ (b) $n = 50$

(c) $n = 60$ (d) $n = 70$

(e) $n = 80$ (f) $n = 90$

Figure 3.4: Binary logarithm of survival function (SF) of the score distribution for uniform targets in various dimensions n, according to prediction from the Heuristic Analysis (dashed blue line), and experiments (red line). Each experimental curve has 2^{45} samples, which were obtained with `code/unif_scores.py` and are listed in `data/unif_scores_nX.csv` of the auxiliary files.



(a) $n = 40$ (b) $n = 60$

(c) $n = 80$ (d) $n = 50$

(e) $n = 70$ (f) $n = 90$

Figure 3.5: CDF of the score for Gaussian BDD targets in various dimensions n, with parameter $\sigma = 0.7 \cdot \mathrm{GH}(n)/\sqrt{n}$, according to prediction from the Heuristic Analysis (dashed blue line), and experiments (red line). Each experimental curve has 2^{15} samples, which were obtained with `code/bdd_scores.py` and are listed in `data/bdd_scores_nX.csv` of the auxiliary files.

Table 3.1: Experimental BDD score distribution versus prediction

n	40	50	60	70	80	90
pred. std. dev. σ	8.31	17.19	35.41	72.80	149.52	307.01
meas. std. dev. σ	11.86	30.24	80.19	221.03	614.16	1732.72
ratio pred./meas.	1.43	1.76	2.26	3.04	4.11	5.64
pred. mean μ	16.71	44.68	120.26	320.53	859.99	2310.28
meas. mean μ	16.75	45.32	120.60	322.51	863.15	2325.99
ratio pred./meas.	1.00	1.01	1.00	1.01	1.00	1.01
meas. mean μ	16.75	45.32	120.60	322.51	863.15	2325.99
meas. median	16.13	42.27	108.74	282.54	735.47	1914.36
ratio med./mean	0.96	0.93	0.90	0.88	0.85	0.82

The first thing one notices is that the distribution is significantly more spread out than predicted. The variance is significantly higher. In fact, as shown in Table 3.1, the ratio between the actual and predicted standard deviation, i.e., square root of variance, appears to grow drastically with the dimension.

One might also want to consider the average. However, since the average is linear, its prediction does not require the independent score heuristic, and the prediction is indeed close to the measured average.

A more interesting statistic is the median. According to the heuristic analyses of [LW21, MAT22] reconstructed in Section 3.3, the median is predicted to be equal to the average. In practice, however, the median is noticeably lower. This partially implies that the distribution is quite asymmetric around its average, contrary to the analysis' prediction.

Conclusion

All in all, it is fair to say that the heuristic analyses of [LW21, MAT22], reconstructed in Section 3.3 of the dual attack is completely off when it comes to predicting the score for BDD targets, with or without modulus switching. The distribution is definitely *not* Gaussian, nor even symmetric around its average, and its variance is hugely underestimated.

3.7 Conclusion

Our theoretical contradictions in Section 3.5 and the experiments in Section 3.6 both demonstrate that the independent score heuristic, which has been used abundantly recently to analyse the Dual-Sieve attack, is invalid. Two independent phenomena uncovered by the experiments indicate that the success probability of the Dual-Sieve attack (with or without FFT) is presumably overestimated by this independent score heuristic, at least in certain regimes of interest.

In particular, the concrete cryptanalytic claims of numerous works, including [ADPS16, EJK20, GJ21, MAT22, CST22, AS23], should be considered at least unsubstantiated, as these are currently based on this flawed heuristic. Still, some of those claims might not be that far from reality, but those of [GJ21, MAT22, CST22, AS23] are so deep in the contradictory regime that these are presumably significantly far away from reality.

After mentioning some relevant pitfalls to avoid when fixing the analysis of the dual attack, we conclude this chapter with further improvements to the dual attack, which may help with overcoming the issues of independent score heuristic.

3.7.1 Possible pitfalls

Note that pulling the parameters of an attack outside the contradictory regime in Figure 3.3 is not a convincing way to prevent mispredictions while using the independent score heuristic. Indeed, the contradictory regime shows a fundamental flaw with this heuristic, and experiments indicate that there may well be an impact outside the regime. That is, a sound model needs to be used with theoretically justified predictions matching experimental behaviour measured in Figures 3.4 and 3.5 and beyond.

Moreover, we warn against a flawed argument for a fix, that would consist of constructing the set of dual vectors $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$ from two BKZ reductions and sieves instead of just one, yielding the score function $f_{\mathcal{W}}(\mathbf{t}) = f_{\mathcal{W}_1}(\mathbf{t}) + f_{\mathcal{W}_2}(\mathbf{t})$. One might argue that dual distinguishing now happens in a lattice of dimension 2β instead of β, which would push the attack possibly far outside the contradictory regime of Figure 3.3. By considering the experiments in Figure 3.4, however, one can see this argument is invalid. Each score distribution $f_{\mathcal{W}_i}$ is going to hit its floor,

and thus $f_{\mathcal{W}}$ will also have a floor starting from essentially the same score. Indeed, it suffices to hit the floor of *at least one* of the functions to hit the floor of the aggregate.

Instead, a more credible approach is the following. Run two (or a few) BKZ reductions and sieves, obtain two sets of short dual vectors $\mathcal{W}_1, \mathcal{W}_2$ and then consider the aggregate score as the *minimum* of both scores $f' = \min(f_{\mathcal{W}_1}, f_{\mathcal{W}_2})$ rather than its sum. Now to hit the floor of this new aggregate function f', a uniform target would need to hit *both* floors simultaneously. If $f_{\mathcal{W}_1}, f_{\mathcal{W}_2}$ are sufficiently independent (an assumption that would need substantiation), such event is much rarer than hitting either floor. However, note that taking such a minimum aggregate of scores might also amplify the issues with low scores for BDD targets observed in Section 3.6.3. A robust model for all the distributions at hand is therefore still required.

Note this list of pitfalls is not comprehensive, and thus any potential fix of an attack, needs to be motivated at least by theoretical arguments or experimental validation.

3.7.2 Possible improvements

The experiments on the independent score heuristic show that current parametrisations of the attacks in [GJ21, MAT22, CST22, AS23] will result in many false positives. However, if this number of false positives is still reasonable, one might mitigate the issue at hand, although the attack cost will be somewhat higher. Indeed, one may consider the Dual-Sieve-FFT technique as a first 'filtering stage' in an attack with multiple stages: the remaining problem is still the problem of finding one BDD target among many candidates, but in an easier lattice, i.e., of lower dimension and/or sparser. If this remaining problem is sufficiently easier to accommodate all these targets, the Dual-Sieve-FFT attack might be salvaged, although the cost of such an attack will be higher than what was listed in [GJ21, MAT22, CST22, AS23]. Also note that first filtering and then filtering the 'survivors' again, is conceptually not that far off from the idea of taking the smallest of both scores.

Alternatively, one could potentially improve the dual attack further by applying different weights to the individual scores, to achieve better separation between the BDD and uniform distributions [AR04, LW21].

Chapter 4

Accurate Score Prediction for Dual-Sieve Attacks

This chapter is based on joint work with L. Ducas published in the Journal of Cryptology [DP26], which combines published work [DP23c] with new material [DP23b].

We propose a heuristic for the output of a lattice sieve that seems weaker than the independent score heuristic. When recovering part of the secret in the Dual-Sieve attack, we use this heuristic to devise predictions for the score distribution associated to candidates: for correct candidates with noise drawn from any radial distribution, we predict scores using a central limit heuristic; for incorrect candidates, we predict scores by approximating the Voronoi cell by a ball.

In addition, we compare the predicted score distributions to experimental data, and observe that these predictions are qualitatively and quantitatively quite accurate. This allows one to accurately estimate the number of false-positives and false-negatives encountered in the Dual-Sieve attack, and it opens the door to a sound analysis thereof. One can now, for example, explore a way of mitigating the many false-positives.

4.1 Introduction

The previous chapter has revealed that the independent score heuristic, as used in the analysis of the Dual-Sieve attack [LW21, GJ21, MAT22], leads to an incorrect prediction for the score distribution of both BDD and uniform targets. In particular, the experiments in Section 3.6 suggest that this heuristic overestimates the success probability of distinguishing between a BDD target and a uniform target. Subsequently, by parametrising a Dual-Sieve attack according to this flawed heuristic, the attack will most likely fail.

For assessing the security of LWE-based cryptosystems, such as ML-KEM and ML-DSA, it is required to know precisely what the concrete cost is of running a state-of-the-art primal *or* dual attack. For this, we will need a new, reliable model for the Dual-Sieve attack, which possibly requires a newly devised heuristic.

In Section 3.5, the independent score heuristic contradicts in three regimes with other lemmata or well-tested heuristics, and thus seems to be the reason for the mispredictions in Section 3.6. These mispredictions indicate that there may be some correlation between the inner products $\langle \mathbf{w}, \mathbf{t} \rangle \pmod 1$, while computing the score of a target $\mathbf{t}$.

The situation may be resolved by identifying the 'confounding variable' that causes dependencies between these random variables. In this case, the confounding variable seems to be the target $\mathbf{t}$, or more precisely its norm $\|\mathbf{t}\|$. For example, scaling the target by a factor α, multiplies *all* the inner products by α. Instead, by fixing the confounding variable $\mathbf{t}$, one may more reasonably hope the inner products $\langle \mathbf{w}, \mathbf{t} \rangle \pmod 1$ become independent again. Fixing the target $\mathbf{t}$ leads very naturally to studying the score distribution of $\mathbf{t}$, where we now consider the randomness of the lattice and the set of short dual vectors.

4.1.1 Contributions

We propose new models for the score functions of BDD targets and uniform targets in Section 4.3. In Section 4.4, we then verify the predictions given by these models in an experimental setting, and observe that the predictions are quite accurate.

Accurate score distribution models (Section 4.3)

First, we will model the set of short dual vectors obtained by lattice sieving [BDGL16], using Heuristic 2.114 as being uniformly from some ball. Such a model is more accurate compared to that of existing literature, which approximates the distribution of lattice sieve vectors by i.i.d. Gaussians [MAT22, Assumption 4.4]. Although our choice somewhat complicates the Fourier analysis, it is still feasible thanks to the well-known Bessel functions, which we introduce in Section 4.2.

For a fixed target $\mathbf{t}$, we derive a novel *analytic expression* for the expectation value and variance of the score distribution. Note that in this case, each individual score has the same individual score distribution, so we can resort to Heuristic 2.3—which seems reasonable as there are exponentially many dual vectors—to reason that the sum of individual scores is normally distributed.

While the case of a particular target is of limited interest in our context of search-BDD, fortunately, we can easily extend the analysis to any other radial distribution, such as Gaussian or uniform in a ball.

Although it is notoriously hard to determine the exact shape of the Voronoi cell [Laa21], one can roughly approximate the Voronoi cell by a ball of same volume. This then approximates the uniform target distribution by a radial distribution, which allows us to predict the score distribution for uniform targets using the above prediction.

Experimental validation (Section 4.4)

The overall approach remains heuristic, but the previously identified issues have now been sidestepped, and we further confirm those score distribution predictions with extensive experiments. All the predictions that are made throughout this work have been extensively tested by large experiments to verify whether the proposed heuristics are reasonable in the context of solving decision-BDD.

Figures 4.1 and 4.2 show the predicted score distributions for BDD targets drawn uniformly from a sphere or ball, and from a Gaussian, using a lattice of dimension 90. In addition, the predicted score distribution for uniform targets is shown in Figure 4.3. This figure contains data of experiments in dimensions $40, 50, 60$ and 70, each based on 2^{48} samples.

Finally, Figure 4.4 shows the score distribution for uniform targets when using a larger saturation radius inside lattice sieving. According to

our prediction, the 'waterfall-floor' phenomenon can be observed with fewer samples, compared to the default saturation radius of $r_{\text{sat}} = \sqrt{4/3}$. The experiments confirm this behaviour.

Open data

All the written code and the obtained data to generate the figures, can be found in the GitHub repository https://github.com/ludopulles/AccurateScorePredictionDualSieveAttacks. Experiments are written in Python and use the G6K and FPyLLL libraries [ADH+19, FPL25], and there is a binding to some C code to accelerate the FFT.

4.1.2 Related work

4

There are several related works [BW25, PS24, CDMT24], which are concurrent works with [DP23b].

The work of Bashiri and Wiemers [BW25] proposes an analysis of the variance of the score of the BDD target and the uniform targets. They use a significantly different approach, and obtain predictions compatible with the experimental data from Section 3.6.3. This does not directly allow us to conclude on false-negative nor false-positive rates, as it does not fully characterise the distribution.

The work of Pouly and Shen [PS24] proposes a provable variant of the dual attack using discrete Gaussian sampling in place of sieving vectors. Notably, their provable regime does not intersect the heuristic contradictory regime in Section 3.5.3. In fact, and quite interestingly, both regimes are separated by a constant factor of 2 on the length of the BDD target.

The work of Carrier, Debris-Alazard, Meyer-Hilfiger and Tillich, while mostly focusing on the case of statistical decoding for codes, also includes a short section on the case of lattices [CDMT24, Sec. 8]. Here, they try to predict, also using Bessel functions, the floor phenomenon for uniform targets modulo the lattice. They use a somewhat comparable but not identical reasoning and derivations as that of our Section 4.3.4.

4.2 Bessel function

The class of Bessel functions is defined as follows, and will be useful for the expected score of errors from a ball later on. The following definition

is based on [Wat1922, §3.3 (4)], which considers the more general setting where α and t may be complex.

Definition 4.1 (Bessel function)**.** For any $\alpha > -\frac{1}{2}$, the *Bessel function (of the first kind) of order* α is given by

$$J_\alpha(t) = \frac{(t/2)^\alpha}{\sqrt{\pi}\cdot\Gamma\left(\alpha+\frac{1}{2}\right)}\int_{-1}^{1} e^{its}\left(1-s^2\right)^{\alpha-\frac{1}{2}}\mathrm{d}s,$$

for $t > 0$.

Observe that $J_\alpha(t)$ is real because $I(-s) = \overline{I(s)}$ where $I(s)$ denotes the integrand in the definition, cf. [AS64, 9.1.20].

It is well known that the Bessel function of some order α has an infinite number of positive roots. Let us denote with $j_{\alpha,n}$, the nth positive root of the Bessel function of order α.

Lemma 4.2 ([Wat1922, §15.81])**.** *The first positive root of the Bessel function of order* α *satisfies*

$$j_{\alpha,1} = \alpha + 1.855757\ldots\cdot\alpha^{1/3} + O(\alpha^{-1/3}),$$

as $\alpha\to\infty$*, where* $1.855757\ldots$ *can be computed numerically up to arbitrary precision. In addition, we have* $J_\alpha(x) > 0$ *for all* $0 < x < j_{\alpha,1}$.

For convenience later on, we define the following function.

Definition 4.3. The real-valued function ξ has domain $\mathbb{R}_{>-1/2}\times\mathbb{R}_{>0}$ and is given by

$$\xi(\alpha,x) = \frac{\Gamma(\alpha+1)}{(\pi x)^\alpha}J_\alpha(2\pi x)\ , \tag{4.1}$$

for any (α, x) in its domain.

The image of the function $x\mapsto\xi(\alpha,x)$ is contained in $[-1,1]$.

Remark 4.4. By using [AS64, 9.1.10], and the crude approximation $\Gamma(\alpha+k+1)/\Gamma(\alpha+1) = (\alpha+1)\cdot(\alpha+2)\cdot\cdots\cdot(\alpha+k)\approx(\alpha+1)^k$, we get the following approximation for small, positive x:

$$\begin{aligned}\xi(\alpha,x) &= \sum_{k=0}^{\infty}\frac{(-\pi^2x^2)^k}{k!(\alpha+1)(\alpha+2)\ldots(\alpha+k)}\\ &\approx \sum_{k=0}^{\infty}\frac{(-\pi^2x^2)^k}{k!(\alpha+1)^k} = e^{-\frac{\pi^2x^2}{\alpha+1}}\ .\end{aligned}$$

However, this approximation is not needed in computations, since ξ can be computed easily in many programming languages.

Remark 4.5. In high dimensions, one may get some numerical errors when computing ξ directly based on Equation (4.1). These issues are circumvented by computing ξ using following identity

$$\xi(\alpha, x) = {}_0F_1\Big(\ ;\ \alpha+1;\ -\pi^2 x^2\Big)\ ,$$

where ${}_0F_1$ is the confluent hypergeometric function [AS64, 9.1.69]. For example, in Python one can use the function `hyp0f1` from the `mpmath` package, which implements ${}_0F_1$.

4.2.1 Relation to the Fourier transform

Bessel functions occur very naturally in the context of Fourier transforms, in particular as the Fourier transform of the indicator function of a ball.

Lemma 4.6. *The Fourier transform of $f = \mathbf{1}_{r\mathcal{B}^n}$ is given by*

$$\widehat{f}(\mathbf{x}) = \left(\frac{r}{\|\mathbf{x}\|}\right)^{\frac{n}{2}} \cdot J_{n/2}(2\pi r\|\mathbf{x}\|) = \mathrm{Vol}_n(r\mathcal{B}^n)\cdot \xi\left(\frac{n}{2}, r\|\mathbf{x}\|\right)\ ,$$

where $\mathbf{x} \in \mathbb{R}^n$.

The following proof is based on [Fis13].

Proof. First, since f is radial, $\widehat{f}$ is also radial. Hence, let us do the proof only for $\mathbf{x} = (s, 0, \ldots, 0)$. Then,

$$\begin{aligned}\widehat{f}(\mathbf{x}) &= \int_{r\mathcal{B}^n} e^{-2\pi i s \mathbf{y}_1}\mathrm{d}\mathbf{y} = \int_{-r}^{r} e^{-2\pi i s t}\,\mathrm{Vol}_{n-1}\Big(\sqrt{r^2-t^2}\mathcal{B}^{n-1}\Big)\mathrm{d}t \\ &= \frac{r^n \pi^{\frac{n-1}{2}}}{\Gamma(\frac{n}{2}+\frac{1}{2})}\int_{-1}^{1} e^{2\pi i r s u}(1-u^2)^{\frac{n-1}{2}}\mathrm{d}u = \left(\frac{r}{s}\right)^{n/2} J_{\frac{n}{2}}(2\pi r s)\ .\end{aligned}$$

For the last equality, note that we have

$$\begin{aligned}\left(\frac{r}{\|\mathbf{x}\|}\right)^{\frac{n}{2}} J_{\frac{n}{2}}(2\pi r\|\mathbf{x}\|) &= \left(\frac{r}{\|\mathbf{x}\|}\right)^{\frac{n}{2}} \frac{(\pi r\|\mathbf{x}\|)^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}\,\xi\left(\frac{n}{2}, r\|\mathbf{x}\|\right) \\ &= \mathrm{Vol}_n(r\mathcal{B}^n)\,\xi\left(\frac{n}{2}, r\|\mathbf{x}\|\right)\ .\end{aligned}$$

□

4.3 Score distribution models

In this section, we will gradually develop new predictions for the score distributions of both BDD targets and uniform targets, which hopefully match closely experimental data from Section 3.6.

In Section 4.3.1, we derive analytic expressions for the mean and variance of the individual score function $f_{\mathbf{w}}(\mathbf{t})$. In Section 4.3.2, we derive expressions for the exact mean and variance of the score distribution for a fixed BDD target based on Heuristic 2.114. Here, we present a new heuristic that says the score distribution for a single target $\mathbf{t}$ *is* Gaussian and only depends on $d(\mathbf{t}, \Lambda)$, when the probability is taken over the lattice. Using this heuristic and by integrating over the radius, we predict in Section 4.3.3 the score distribution for a radial target distribution, such as uniform in a ball and Gaussian. Moreover, by approximating the Voronoi cell by a ball of volume $\det(\Lambda)$, we can also predict the score distribution for an error uniform modulo Λ, which is done in Section 4.3.4.

Throughout this section, let us assume Λ has full rank and $\det(\Lambda) = 1$ without loss of generality, because the ambient space can be restricted to $\mathrm{span}(\Lambda)$, see Proposition 2.59, and any lattice Λ can be rescaled to one of unit-volume.

We strongly advise the reader to think carefully for which lattice Λ to use the predictions of this section. Namely, in many of the recent dual attacks, it is a projected sublattice of Λ_{LWE} having rank-β_{sieve}, see Remark 3.2.

4.3.1 Individual score function

First, let us fix an arbitrary vector $\mathbf{t} \in \mathbb{R}^n$, and look at the score function $f_{\mathbf{w}}(\mathbf{t}) = \cos(2\pi \langle \mathbf{w}, \mathbf{t} \rangle)$, where $\mathbf{w}$ is sampled uniformly from a sphere. In this case, the mean and variance are expressible in terms of Bessel functions, and thus in terms of ξ from Definition 4.3.

Lemma 4.7. *Given an arbitrary vector* $\mathbf{t} \in \mathbb{R}^n$ *and* $r \in \mathbb{R}_{>0}$*, when the random variable* $\mathbf{w}$ *follows the uniform distribution on the* $(n-1)$-

dimensional sphere of radius r, we have

$$\mathop{\mathbb{E}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{S}^{n-1})}[f_{\mathbf{w}}(\mathbf{t})] = \xi\left(\frac{n}{2}-1,\ r\|\mathbf{t}\|\right) \ , \qquad \textit{and}$$
$$\mathop{\mathbb{V}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{S}^{n-1})}[f_{\mathbf{w}}(\mathbf{t})] = \frac{1}{2}+\frac{1}{2}\xi\left(\frac{n}{2}-1,\ 2r\|\mathbf{t}\|\right) - \xi\left(\frac{n}{2}-1,\ r\|\mathbf{t}\|\right)^2 \ .$$

Proof. The expectation value follows directly from a classic result from Fourier Analysis, see e.g. [GS64, p. 198] or [SW71, p. 154], as we have

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{S}^{n-1})}[f_{\mathbf{w}}(\mathbf{t})] &= \frac{1}{\mathrm{Vol}_n(r\mathcal{S}^{n-1})}\int_{r\mathcal{S}^{n-1}} e^{2\pi i\cdot\langle \mathbf{w},\mathbf{t}\rangle}\,\mathrm{d}\mathbf{w} \\ &= \frac{\Gamma(\frac{n}{2})\cdot J_{\frac{n}{2}-1}(2\pi r\|\mathbf{t}\|)}{(\pi r\|\mathbf{t}\|)^{\frac{n}{2}-1}}.\end{aligned}$$

The trigonometric identity $f_{\mathbf{w}}(\mathbf{t})^2 = \frac{1}{2}+\frac{1}{2}\cos(4\pi\,\langle \mathbf{w},\mathbf{t}\rangle) = \frac{1}{2}+\frac{1}{2}f_{\mathbf{w}}(2\cdot\mathbf{t})$ allows us to derive a similar expression for the variance, by observing

$$\begin{aligned}\mathop{\mathbb{V}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{S}^{n-1})}[f_{\mathbf{w}}(\mathbf{t})] &= \mathbb{E}_{\mathbf{w}}[f_{\mathbf{w}}(\mathbf{t})^2] - \mathbb{E}_{\mathbf{w}}[f_{\mathbf{w}}(\mathbf{t})]^2 \\ &= \frac{1}{2}+\frac{1}{2}\mathbb{E}_{\mathbf{w}}[f_{\mathbf{w}}(2\cdot\mathbf{t})] - \mathbb{E}_{\mathbf{w}}[f_{\mathbf{w}}(\mathbf{t})]^2 \ ,\end{aligned}$$

and reusing the result for the expectation value. □

Remark 4.8. The individual score function has been studied before, for example by Laarhoven and Walter [LW21, Section 4.1]. However, they approximate the distribution for $\mathbf{w}$ by $\mathcal{N}_n(0, r^2/n)$, yielding only an approximation for the expectation value and variance, whereas the above lemma has an exact expression for both. Still, if one is inclined to work with a crude approximation, by Remark 4.4, the expectation in the above lemma is for small values r approximated by $\exp\left(-\frac{2\pi^2\|\mathbf{t}\|^2 r^2}{n}\right)$, which matches the approximated expectation in [LW21].

As we will reuse the mean and variance for the ball, let us introduce the following notation.

Definition 4.9. For $n\in\mathbb{Z}_{\geq 1}$, $r_p, r_d\in\mathbb{R}_{>0}$ let us denote

$$E_{n,r_d}(r_p) = \xi\left(\frac{n}{2}, r_d\cdot r_p\right), \qquad \text{and}$$
$$V_{n,r_d}(r_p) = \frac{1}{2}+\frac{1}{2}E_{n,r_d}(2\cdot r_p) - (E_{n,r_d}(r_p))^2.$$

The result in Lemma 4.7 can be translated easily to a uniform distribution on a ball, because sampling uniformly from the n-dimensional ball can be done by first sampling from a $(n+1)$-dimensional sphere and then dropping the last two coordinates [BGMN05, Corollary 4].

Corollary 4.10. *Given an arbitrary vector $\mathbf{t} \in \mathbb{R}^n$ and $r \in \mathbb{R}_{>0}$, when the random variable $\mathbf{w}$ follows the uniform distribution on the n-dimensional ball of radius r, we have*

$$\mathop{\mathbb{E}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{B}^n)}[f_{\mathbf{w}}(\mathbf{t})] = E_{n,r}(\|\mathbf{t}\|), \quad \text{and} \quad \mathop{\mathbb{V}}_{\mathbf{w}\leftarrow \mathrm{U}(r\mathcal{B}^n)}[f_{\mathbf{w}}(\mathbf{t})] = V_{n,r}(\|\mathbf{t}\|) \ .$$

4.3.2 Total score function

The next step is to determine the expectation value and variance of the total score $f_{\mathcal{W}}(\mathbf{t})$, in the case that the set $\mathcal{W}$ is obtained by lattice sieving. It is now necessary to take into account the randomness with which the lattice Λ is sampled, because the lattice has influence on the set of dual vectors $\mathcal{W} \subseteq r_d\mathcal{B}^n \cap \Lambda^\vee$.

In this section, let us fix a target $\mathbf{t} \in \mathbb{R}^n$ initially. We will study the score distribution for this target $\mathbf{t}$ by considering the randomness of the lattice and the set of dual vectors $\mathcal{W}$.

Upon first reading the rest of this section, the reader is encouraged to think that the set of dual vectors is $\mathcal{W} = r_d\mathcal{B}^n \cap \Lambda^\vee$, or rather: half of them to break the negation symmetry (recall Section 2.6). To argue about the dual vectors we will make use of Heuristic 2.114, which can be used for more general saturation parameters. When sieving in practice, a saturation ratio $f_{\mathrm{sat}} < 1$ is used to obtain enough dual vectors in the ball, and it is very natural to believe that different saturation parameters do not change the score distribution significantly, other than having an effect on N, the expected size of $\mathcal{W}$. Although all the shortest vectors are usually found, most of the missing $1 - f_{\mathrm{sat}}$ portion of vectors is around the boundary. Still, we believe this does not noticeably affect the score distribution for reasonably large f_{sat}.

By using Heuristic 2.114, the $\mathbf{w} \in \mathcal{W}$ are i.i.d. samples and match the distribution in Corollary 4.10, too. Because heuristically the scores $(f_{\mathbf{w}}(\mathbf{t}))_{\mathbf{w}\in\mathcal{W}}$ are i.i.d., we expect that the total score is Gaussian for large $|\mathcal{W}|$ by Heuristic 2.3. This argument results in the following heuristic.

Heuristic 4.11. *Fix a target $\mathbf{t} \in \mathbb{R}^n$ of norm $\|\mathbf{t}\| \in (0, \mathrm{GH}(n))$. The score distribution $f_{\mathcal{W}}(\mathbf{t})$ is Gaussian with mean $N \cdot E_{n,r_d}(\|\mathbf{t}\|)$ and variance*

$N \cdot V_{n,r_d}(\|\mathbf{t}\|)$ when sampling Λ and obtaining N dual vectors $\mathcal{W} \leftarrow \text{Sieve}(\Lambda^\vee, r_{\text{sat}}, f_{\text{sat}})$ from a sieve algorithm, where E_{n,r_d} and V_{n,r_d} are defined in Definition 4.9 and $r_d = r_{\text{sat}}\,\text{GH}(n)$. In particular, the CDF of the score distribution is as follows:

$$\Pr_{\Lambda,\mathcal{W}}[f_{\mathcal{W}}(\mathbf{t}) \le x] = \frac{1}{2} + \frac{1}{2}\,\text{erf}\left(\frac{x - N \cdot E_{n,r_d}(\|\mathbf{t}\|)}{\sqrt{2N \cdot V_{n,r_d}(\|\mathbf{t}\|)}}\right). \tag{4.2}$$

Heuristic Justification. First, based on Heuristic 2.114, we assume, over the randomness of Λ and $\mathcal{W}$, that $\mathcal{W}$ is a set of i.i.d. $\mathbf{w} \leftarrow \text{U}(r_d\mathcal{B}^n)$ of size N. Then, Corollary 4.10 proves that each individual score $(f_{\mathbf{w}})_{\mathbf{w}\in\mathcal{W}}$ has mean $E_{n,r_d}(\|\mathbf{t}\|)$ and variance $V_{n,r_d}(\|\mathbf{t}\|)$.

Because the $(f_{\mathbf{w}}(\mathbf{t}))_{\mathbf{w}\in\mathcal{W}}$ are i.i.d. individual scores, we can use Heuristic 2.3 here. In this case, the total score distribution will have mean $N \cdot E_{n,r_d}(\|\mathbf{t}\|)$ and variance $N \cdot V_{n,r_d}(\|\mathbf{t}\|)$. △

Remark 4.12. The heuristic should not be used for targets $\mathbf{t}$ of norm above $\text{GH}(n)$. Given a random lattice, one expects there exists $\mathbf{t}' \in \mathbf{t} + \Lambda$ of norm $\|\mathbf{t}'\| \le \text{GH}(n)$ by the Gaussian heuristic. Taking $\mathbf{t}' \in \mathbf{t} + \Lambda$ of minimal norm we have $f_{\mathcal{W}}(\mathbf{t}) = f_{\mathcal{W}}(\mathbf{t}')$, but using Heuristic 2.114 and Corollary 4.10 yields a *different* expected score of $E_{n,r_d}(\|\mathbf{t}'\|)$ for $\mathbf{t}'$, which is much higher than $E_{n,r_d}(\|\mathbf{t}\|) \approx 0$ if $\|\mathbf{t}'\| < 0.5\,\text{GH}(n)$, say. Hence, use of Heuristic 2.114, which forgets that $\mathcal{W}$ are dual vectors, results in contradictory predictions in the case $\|\mathbf{t}\| > \text{GH}(n)$.

On the other hand, given a target $\mathbf{t}$ of norm $\alpha\,\text{GH}(n)$ for a constant $\alpha < 1$, we expect $\|\mathbf{t}\| = d(\mathbf{t}, \Lambda)$ holds for most random lattices as $n \to \infty$ by Heuristic Claim 2.48, avoiding this issue with Heuristic 2.114.

Interestingly, the expected score $f_{\mathcal{W}}(\mathbf{t})$ is very similar to the score when one takes $\mathcal{W} = r_d\mathcal{B}^n \cap \Lambda^\vee$. Without any heuristics, Theorem 2.75 and Lemma 4.6 yield the following identity:

$$\begin{aligned} f_{\mathcal{W}}(\mathbf{t}) &= \sum_{\mathbf{w}\in\Lambda^\vee} \mathbf{1}_{r_d\mathcal{B}^n}(\mathbf{w}) e^{2\pi i\langle \mathbf{w},\mathbf{t}\rangle} \\ &= \text{Vol}_n(r_d\mathcal{B}^n) \cdot \sum_{\mathbf{v}\in\Lambda} \xi\left(\frac{n}{2},\ r_d\|\mathbf{v}+\mathbf{t}\|\right). \end{aligned} \tag{4.3}$$

For $r_p \ll \text{GH}(n)$, the vector $\mathbf{v} = \mathbf{0}$ gives the main contribution of $\xi(\frac{n}{2},\ r_d r_p)$ to the summation in Equation (4.3), because $\xi(n/2,\ -)$ is a rapidly decaying function.

At this point, one could argue that Heuristic 4.11 is not better at predicting a score distribution than the independent score heuristic because it still uses Heuristic 2.3. However, observe that Heuristic 4.11 uses Heuristic 2.3 for i.i.d. $f_{\mathbf{w}}(\mathbf{t})$, after first fixing the target $\mathbf{t}$ independent of the lattice.

On the other hand, usage of the independent score heuristic leads to a prediction that uses Heuristic 2.3 for the individual scores $f_{\mathbf{w}}(\mathbf{t})$ *irrespective of the target distribution for* $\mathbf{t}$, which ignores the dependency of its norm on the mean score. Heuristic 4.11, being restricted to a fixed target norm, thus seems reasonable. However, large experiments are ultimately needed to gain confidence in the heuristic, and these can be found in Section 4.4.2.

Lastly, the following lemma contains an upper bound on the norm of a BDD target such that it is distinguishable from targets uniform modulo the lattice.

Lemma 4.13. *If* $r_p \cdot r_d < \frac{n}{4\pi}$, *then* $E_{n,r_d}(r_p) > 0$.

Proof. The first zero of the function $x \mapsto \xi(\frac{n}{2},\ x)$ is at

$$x = \frac{1}{2\pi} j_{n/2,1} > \frac{n}{4\pi} > r_p r_d \ ,$$

where we use Lemma 4.2 in the first inequality. □

Using the approximation $\mathrm{GH}(n) \approx \sqrt{\frac{n}{2\pi e}}$, note the condition on $r_p r_d$ translates to roughly $r_p r_d < \frac{e}{2}\,\mathrm{GH}(n)^2$ for large n.

Note that above lemma is sharp, i.e., if we take $r_p r_d = \alpha n$ for some constant $\alpha > \frac{1}{4\pi}$, then $r_p r_d = \frac{n}{4\pi} + \left(\alpha - \frac{1}{4\pi}\right)n$. Lemma 4.2 implies there exists some $N \in \mathbb{N}$ such that for all $n \geq N$ we have $r_p r_d > \frac{1}{2\pi} j_{n/2,1}$, so there will be some $n \geq N$ giving a negative expectation score.

4.3.3 Radial error distributions

The predictions for a particular error can now be extended to any error distribution χ that is radial, i.e., the PDF is the same at two points of the same norm. In particular, given the radial distribution χ, let us write $f(r)$ to denote the probability (density) of a sample $\mathbf{e} \leftarrow \chi$ having norm r, i.e.,

$$f(r) = \frac{\mathrm{d}}{\mathrm{d}r} \Pr_{\mathbf{t} \leftarrow \chi}[\|\mathbf{t}\| \leq r].$$

We refer to $f(r)$ as the *norm distribution of* χ.

Using Heuristic 4.11 we arrive at the following claim.

Heuristic Claim 4.14. *Let χ be an error distribution with norm distribution $f(r)$, such that the support of χ is mostly contained in $\mathrm{GH}(n)\mathcal{B}^n$. The score distribution $f_{\mathcal{W}}(\mathbf{t})$ for an error vector $\mathbf{t} \leftarrow \chi$ has the following CDF:*

$$\Pr_{\Lambda, \mathbf{t}\leftarrow\chi}[f_{\mathcal{W}}(\mathbf{t}) \leq x] = \frac{1}{2} + \frac{1}{2}\int_0^\infty \operatorname{erf}\left(\frac{x - N \cdot E_{n,r_d}(r)}{\sqrt{2N \cdot V_{n,r_d}(r)}}\right) f(r)\,\mathrm{d}r, \tag{4.4}$$

where the probability is taken over randomness from sampling Λ and obtaining N dual vectors $\mathcal{W} \leftarrow \textsc{Sieve}(\Lambda^\vee, r_{\mathrm{sat}}, f_{\mathrm{sat}})$ from a sieve algorithm.

4

There are now two radial distributions of particular interest: Gaussian and the uniform distribution on the ball.

Gaussian error

The norm distribution of samples from $\mathcal{N}_n(0,1)$, is the χ-distribution of order n, which has a PDF given by:

$$f(r;n) = \frac{r^{n-1}\exp\left(-\frac{r^2}{2}\right)}{2^{\frac{n}{2}-1}\,\Gamma\left(\frac{n}{2}\right)} \qquad (r \in \mathbb{R}_{>0}).$$

Note that one should not get confused with the (perhaps more well-known) χ^2-distribution, which considers the *squared norm*.

Now instantiating Heuristic Claim 4.14 for $\mathcal{N}_n(0,\sigma^2)$, the Gaussian error distribution of parameter $\sigma > 0$, yields the following claim.

Heuristic Claim 4.15. *Let $\sigma \in (0, \mathrm{GH}(n)/\sqrt{n})$, and let $f(x;n)$ be the PDF of the χ-distribution. The score distribution $f_{\mathcal{W}}(\mathbf{t})$ for an error $\mathbf{t} \leftarrow \mathcal{N}_n(0,\sigma^2)$ has the following CDF:*

$$\Pr_{\substack{\Lambda,\mathcal{W},\\ \mathbf{t}\leftarrow\mathcal{N}_n(0,\sigma^2)}}[f_{\mathcal{W}}(\mathbf{t}) \leq x] = \frac{1}{2} + \frac{1}{2}\int_0^\infty \operatorname{erf}\left(\frac{x - N \cdot E_{n,r_d}(r)}{\sqrt{2N \cdot V_{n,r_d}(r)}}\right) f\left(\frac{r}{\sigma};n\right)\frac{\mathrm{d}r}{\sigma}, \tag{4.5}$$

where the probability is taken over randomness from sampling Λ and obtaining N dual vectors $\mathcal{W} \leftarrow \textsc{Sieve}(\Lambda^\vee, r_{\mathrm{sat}}, f_{\mathrm{sat}})$ from a sieve algorithm.

Note that the χ-distribution is very concentrated around $\sigma\sqrt{n}$, so the best numerical approximations are obtained when the numerical integration is giving special attention to the region around $\sigma\sqrt{n}$ (see Section 4.4.1).

Error uniform from a ball

Consider the distribution $\mathrm{U}(r_p\mathcal{B}^n)$ for some $r_p > 0$. In this case,

$$\Pr_{\mathbf{t}\leftarrow\mathrm{U}(r_p\mathcal{B}^n)}[\|\mathbf{t}\| \le r] = r^n/r_p^n$$

for any $r \in [0, r_p]$, while this equals 1 for $r \ge r_p$. Strictly speaking, the CDF of the norm distribution is not differentiable at $r = r_p$, so $f(r_p)$ is not defined. Still, one can mitigate this issue by integrating from 0 to r_p in Equation (4.4), yielding the following claim.

Heuristic Claim 4.16. *Let $r_p \in (0, \mathrm{GH}(n))$. The score distribution $f_{\mathcal{W}}(\mathbf{t})$ for an error $\mathbf{t} \leftarrow \mathrm{U}(r_p\mathcal{B}^n)$ has the following CDF:*

$$\Pr_{\substack{\Lambda,\mathcal{W},\\ \mathbf{t}\leftarrow\mathrm{U}(r_p\mathcal{B}^n)}}[f_{\mathcal{W}}(\mathbf{t}) \le x] = \frac{1}{2} + \frac{1}{2}\int_0^{r_p} \mathrm{erf}\left(\frac{x - N\cdot E_{n,r_d}(r)}{\sqrt{2N\cdot V_{n,r_d}(r)}}\right)\frac{nr^{n-1}\,\mathrm{d}r}{r_p^n}, \tag{4.6}$$

where the probability is taken over randomness from sampling Λ and obtaining N dual vectors $\mathcal{W} \leftarrow \textsc{Sieve}(\Lambda^\vee, r_{\mathrm{sat}}, f_{\mathrm{sat}})$ from a sieve algorithm.

4.3.4 Error uniform modulo lattice

Given uniform targets $\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$, the inner product $\langle \mathbf{w}, \mathbf{t}\rangle \pmod 1$ follows the uniform distribution on $\mathbb{R}/\mathbb{Z}$, for any (nonzero) dual vector $\mathbf{w} \in \Lambda^\vee$. Since experiments in Section 3.6 show that the total score function $f_{\mathcal{W}}(\mathbf{t})$ is not Gaussian, we cannot use Heuristic 2.3 for the individual scores $(f_{\mathbf{w}}(\mathbf{t}))_{\mathbf{w}\in\mathcal{W}}$ as it would imply $f_{\mathcal{W}}(\mathbf{t})$ is Gaussian, which is contradictory to experiments. In particular, Section 3.5.2 reveals that use of Heuristic 2.3 would underestimate the probability on an exceptionally high score, because it would not consider the probability of a uniform target landing close to Λ, to which a distinguisher assigns a high score.

In this section, we want to derive predictions for uniform targets, without using the independent score heuristic. The above motivation

shows that having a high score is driven by a target close to the lattice, so it seems important to know how the distance $d(\mathbf{t}, \Lambda)$ from a uniform target to a lattice is distributed to predict the score distribution.

First, because the Voronoi cell tiles the space, assume targets are sampled from $\mathrm{U}(\mathcal{V}(\Lambda))$. The score for such targets would ideally be predicted by using Heuristic Claim 4.14, but $\mathrm{U}(\mathcal{V}(\Lambda))$ is only a radial distribution in dimension $n = 1$. Instead, because the dual vectors already have rotational invariance when using Heuristic 2.114, we could approximate the score distribution by still using Heuristic Claim 4.14 with norm distribution $f(r) = \mathrm{d}F(r)/\mathrm{d}r$, where

$$F(r) = \Pr_{\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)}[d(\mathbf{t}, \Lambda) \leq r] = \mathrm{Vol}_n(\mathcal{V}(\Lambda) \cap r\mathcal{B}^n). \tag{4.7}$$

However, already computing the Voronoi cell is considered a difficult task [MV10]. Note that we have $\frac{1}{2}\lambda_1\mathcal{B}^n \subseteq \mathcal{V}(\Lambda) \subseteq \mu(\Lambda)\mathcal{B}^n$, where $\mu(\Lambda)$ is the covering radius of the lattice. Hence, $F(r) = \mathrm{Vol}_n(r\mathcal{B}^n)$ for $r \leq \frac{1}{2}\lambda_1$. For radii $r \in (\frac{1}{2}\lambda_1(\Lambda), \mu(\Lambda))$, there is no easy expression for $F(r)$ because the ball of radius r is not necessarily contained in the Voronoi cell. However, we still have the upper bound,

$$F(r) \leq \mathrm{Vol}_n(r\mathcal{B}^n), \tag{4.8}$$

which is not much smaller than what is derived in Heuristic Claim 3.13 for $r = (1 - \varepsilon)\,\mathrm{GH}(n)$ with $\varepsilon > 0$. The upper bound in Equation (4.8) can thus be seen as a first order approximation of $F(r)$. The following heuristic implies $F(r)$ equals the upper bound in Equation (4.8).

Heuristic 4.17. *Let $\Lambda \subset \mathbb{R}^n$ be a random full-rank unit-volume lattice. Then,*

$$\mathcal{V}(\Lambda) = \mathrm{GH}(n) \cdot \mathcal{B}^n.$$

Using Heuristic 4.17 and Heuristic Claim 4.16, directly results in the following claim.

Heuristic Claim 4.18. *The score distribution $f_{\mathcal{W}}(\mathbf{t})$ for an error vector $\mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)$ has the following CDF:*

$$\Pr_{\substack{\Lambda, \mathcal{W}, \\ \mathbf{t} \leftarrow \mathrm{U}(\mathbb{R}^n/\Lambda)}}[f_{\mathcal{W}}(\mathbf{t}) \leq x] = \frac{1}{2} + \frac{1}{2} \int_0^{\mathrm{GH}(n)} \mathrm{erf}\left(\frac{x - N \cdot E_{n,r_d}(r)}{\sqrt{2N \cdot V_{n,r_d}(r)}}\right) \frac{n r^{n-1}\,\mathrm{d}r}{\mathrm{GH}(n)^n}, \tag{4.9}$$

where the probability is taken over randomness from sampling Λ and obtaining N dual vectors $\mathcal{W} \leftarrow \textsc{Sieve}(\Lambda^\vee, r_{\mathrm{sat}}, f_{\mathrm{sat}})$ from a sieve algorithm.

This heuristic claim predicts the waterfall and floor shapes in the score distribution for uniform targets.

Waterfall shape: Suppose x is close to zero. If r is much smaller than $\mathrm{GH}(n)$, the integrand is negligible because the expected score $N \cdot E_{n,r_d}(r)$ is much higher than x. On the other hand, for $r \approx \mathrm{GH}(n)$, we have $N \cdot E_{n,r_d}(r) \approx 0$ and $N \cdot V_{n,r_d}(r) \approx N/2$. By using these approximations for $r > (1-\varepsilon)\,\mathrm{GH}(n)$ given some constant $\varepsilon > 0$, we may approximate the integral in Equation (4.9) by $(1-e^{-n/\varepsilon}) \cdot \mathrm{erf}(x/\sqrt{N})$. This approximation approaches $\mathrm{erf}(x/\sqrt{N})$ as n tends to infinity. Hence, the survival function has a waterfall shape around $x = 0$.

Floor shape: Let us consider the survival function of $f_{\mathcal{W}}(\mathbf{t})$:

$$\Pr_{\substack{\Lambda,\mathcal{W},\\ \mathbf{t}\leftarrow\mathrm{U}(\mathbb{R}^n/\Lambda)}}\left[f_{\mathcal{W}}(\mathbf{t}) > x\right] = \frac{1}{2}\int_0^{\mathrm{GH}(n)} \mathrm{erfc}\left(\frac{x - N \cdot E_{n,r_d}(r)}{\sqrt{2N \cdot V_{n,r_d}(r)}}\right)\frac{nr^{n-1}\,\mathrm{d}r}{\mathrm{GH}(n)^n}.$$

For large x, say $c\sqrt{N}$ for some $c > 10$, the biggest contribution to the integral comes from $r \approx r_0$ where r_0 is a solution of the equation $N \cdot E_{n,r_d}(r_0) = x$, because here erfc evaluates to ≈ 1. However, for $r \gg r_0$, the integrand drops to zero as erfc decreases much quicker than r^{n-1}, while for $r \ll r_0$, the integrand also drops to zero because r^{n-1} is simply too small.

4.4 Experiments

In this section, we provide further substantiation of the concrete predictions made in Section 4.3, in particular the predictions in Equations (4.2), (4.5), (4.6) and (4.9), with experimental support. By running experiments, we can verify whether the used heuristics lead to conclusions that precisely match practice.

Similar to Section 3.6, we take $\mathcal{W}$ to be the full output of a lattice sieve on $\Lambda^\vee$ in the experiments. We will compare the score distribution for three possible BDD target distributions with their respective prediction from Section 4.3: uniform from a sphere, uniform from a ball and Gaussian. In addition, we compare the score distribution for uniform targets with the predicted CDF of Equation (4.9).

4.4.1 Implementation details

In the experiments, we used the G6K software [ADH+19] to obtain the dual vectors. We used Python on a high-level together with a binding to some C/C++ code to speed up BDD sampling.

BDD targets

The script `bdd_sample.py` samples the three BDD score distributions by first sampling a random q-ary lattice from a matrix of dimension $n \times n/2$ (n is only even), then running a lattice sieve to acquire many dual vectors, and finally computing scores for many samples. To make the implementation easier, the sampled q-ary lattice is used as the dual lattice $\Lambda^\vee$, because we do not work with primal lattice vectors. The prime used is $q = 3329$.

For the experiments, we sample BDD targets from a distribution with norms concentrated around $f_{\mathrm{GH}} \cdot \mathrm{GH}(n)/\sqrt{q}$ for some parameter $f_{\mathrm{GH}} \in (0, 1]$, where Λ is the primal lattice. Note that the Gaussian heuristic predicts $\lambda_1(\Lambda) \approx \mathrm{GH}(n)/\sqrt{q}$ holds, because the dual lattice has determinant $q^{n/2}$.

More specifically, for a Gaussian sample, we simply sample $\mathbf{x} \leftarrow \mathcal{N}_n(0, \sigma^2)$ with $\sigma = \frac{f_{\mathrm{GH}} \cdot \mathrm{GH}(n)}{\sqrt{n \cdot q}}$. Note that we need a factor of $1/\sqrt{n}$ such that $\|\mathbf{x}\|$ is concentrated around the target length $f_{\mathrm{GH}} \cdot \mathrm{GH}(n)/\sqrt{q}$. For a sample $\mathbf{y}$ uniform on the sphere, we reuse the same Gaussian sample $\mathbf{x}$, and compute

$$\mathbf{y} = \frac{f_{\mathrm{GH}} \cdot \mathrm{GH}(n)}{\sqrt{q}} \cdot \frac{\mathbf{x}}{\|\mathbf{x}\|},$$

which is distributed uniformly on the sphere of radius $f_{\mathrm{GH}} \cdot \mathrm{GH}(n)/\sqrt{q}$. For a sample uniform in the ball, we take a sample uniformly from the $(n+1)$-dimensional sphere in $\mathbb{R}^{n+2}$ and drop the last two coordinates, making use of [BGMN05, Corollary 4]. We only generated 2 more Gaussian samples, and reused the n from $\mathbf{x}$ here. The script `bdd_sample.py` ran to collect 100 000 samples.

The script `bdd_predict.py` then uses this data to plot the experimental data, the prediction from Section 4.3 and the heuristic prediction. The predictions from Section 4.3 required evaluating the Bessel function and an integration, for which we used `hyp0f1` (see Remark 4.5) and `quad` respectively, both in the Python package `mpmath`.

For the integration, the Gaussian prediction was numerically more accurate after splitting the interval $(0, \infty)$ into two at the expected length. For more details, see 'Highly variable functions' in `quad`'s documentation. Predictions for uniform samples from the ball of radius r were obtained by integrating from $0.001r$ to r to prevent the singularity around 0.

Uniform targets

For targets uniform in the lattice, the scripts `unif_sample.py` and `unif_predict.py` were used similar to the BDD case. One should pay special attention to the integration errors caused by numerical integration when calculating the predictions of Heuristic Claim 4.18. Namely, the numerical integration should give an answer with a small *relative error*, because the output can be very small, as can be seen in Figure 4.3. Thus, for numerical integration we used the function `quad` from the Python package `SciPy` with parameters `epsabs=0` and `epsrel=0.001`.

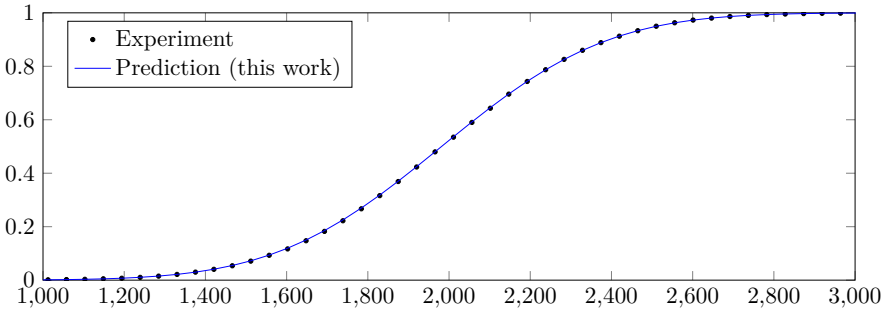
4.4.2 BDD targets

The experimental results for BDD targets can be found in Figure 4.1 and Figure 4.2. The former samples BDD targets at expected distance of $0.7\,\mathrm{GH}(n)$ from the lattice, while the latter samples at distance $0.5\,\mathrm{GH}(n)$ from the lattice. Here, a saturation ratio of $f_{\mathrm{sat}} = 0.99$ and saturation radius of $r_{\mathrm{sat}} = \sqrt{4/3}$ was used, so $N = \lceil \frac{1}{2} f_{\mathrm{sat}} \cdot {r_{\mathrm{sat}}}^{90} \rceil = 207419$ dual vectors were used.

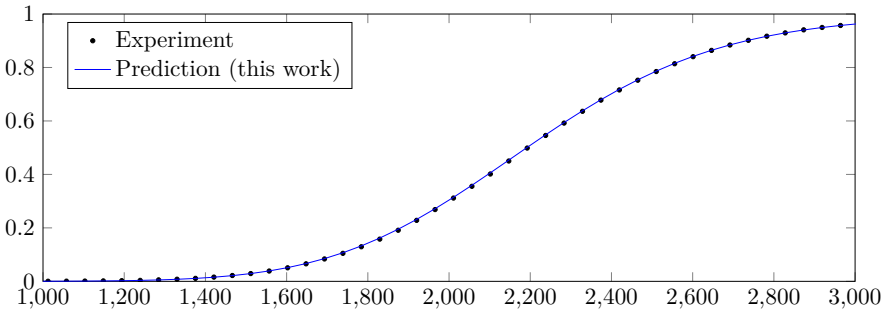
It is clear that the predictions made in Sections 4.3.2 and 4.3.3 give accurate estimates on the score distribution for BDD targets that are from a sphere, from a ball or Gaussian. Moreover, these experiments show that Heuristic 4.11 appears to be a reasonable heuristic to use in this regime.

4.4.3 Uniform targets

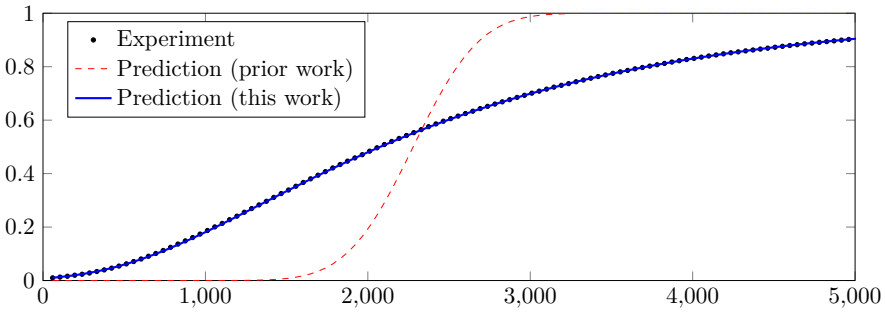
Figure 4.3 compares experimental data with the prediction of the score distribution for uniform targets. Here, the lattice sieve used a saturation ratio of $f_{\mathrm{sat}} = 0.9$ and saturation radius of $r_{\mathrm{sat}} = \sqrt{4/3} \approx 1.1547$. The experimental data for uniform targets is acquired independently of that in Section 3.6.2, which used $2N$ dual vectors. For each of the four experimental curves of Figure 4.3, we ran the script `code/unif_sample.py`, which took slightly more than one day on a server with 40 physical cores.



(a) Uniform from a sphere of radius $0.7\mathrm{GH}(n)$

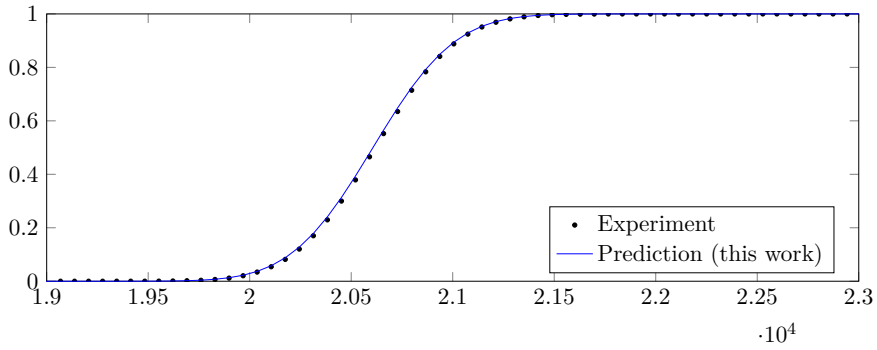


(b) Uniform from a ball of radius $0.7\mathrm{GH}(n)$

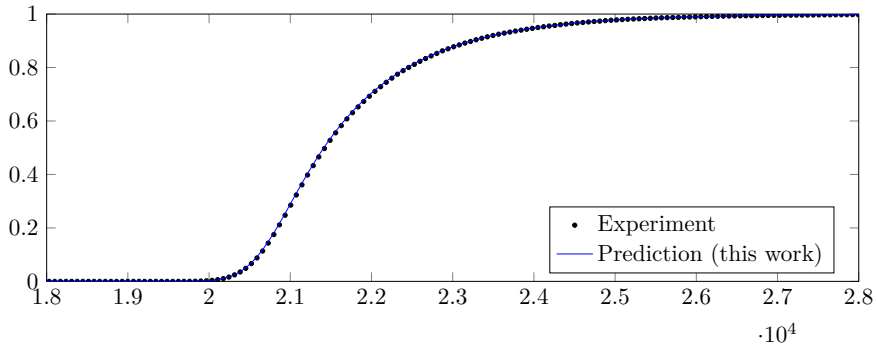


(c) Gaussian with $\sigma = 0.7\mathrm{GH}(n)\,/\sqrt{n}$

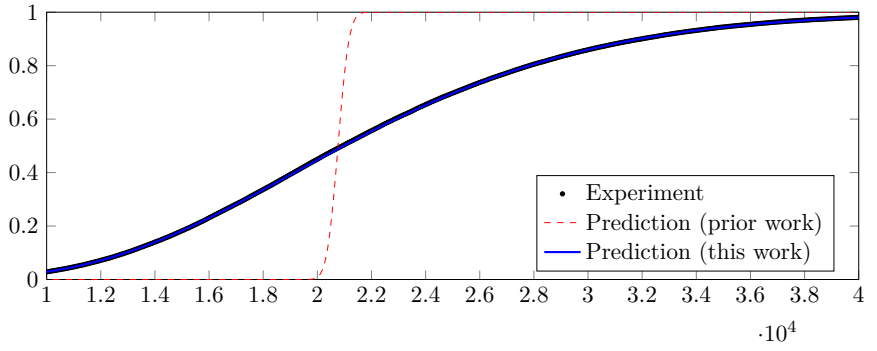
Figure 4.1: CDF of the score for three BDD target distributions in dimension $n = 90$. The prior prediction uses the independent score heuristic, while 'Prediction (this work)' is based on Sections 4.3.2 and 4.3.3. Each experimental curve contains 10^5 samples, which are obtained with `code/bdd_sample.py` and are listed in `data/predictions_n90_ghf0.7.csv` of the auxiliary files.



(a) Uniform from a sphere of radius $0.5\mathrm{GH}(n)$



(b) Uniform from a ball of radius $0.5\mathrm{GH}(n)$



(c) Gaussian with $\sigma = 0.5\mathrm{GH}(n)/\sqrt{n}$

Figure 4.2: CDF of the score for three BDD target distributions in dimension $n = 90$. The prior prediction uses the independent score heuristic, while 'Prediction (this work)' is based on Sections 4.3.2 and 4.3.3. Each experimental curve contains 10^5 samples, which are obtained with `code/bdd_sample.py` and are listed in `data/predictions_n90_ghf0.5.csv` of the auxiliary files.

Note that in dimension 50, 60 and 70, the uniform prediction seems to be very close to the experimental data. The right tail of the experimental data depends on extremely rare events (happening once in 2^{48} trials), so a few of the most-right data points can change a little bit in another run. We believe if more samples were taken, the experimental data would get closer to the prediction, but then the experiment would require a runtime of multiple days on our server.

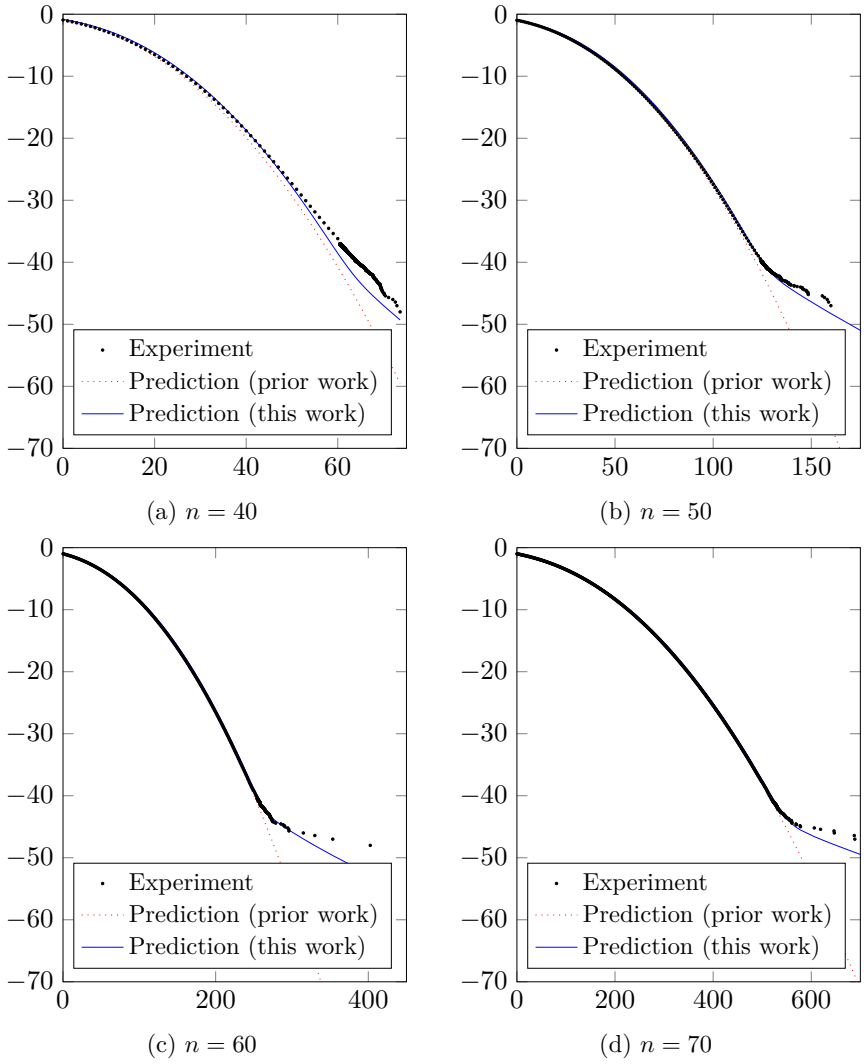
In dimension 40, it seems that the experiments give scores higher than expected by our prediction. However, the sieve provides only $N = \lceil \frac{1}{2} f_{\text{sat}} \cdot {r_{\text{sat}}}^{40} \rceil = 142$ dual vectors in this dimension, which may be too small for the heuristic prediction to be accurate, as Heuristic 2.3 might not apply with few dual vectors, but also the lengths of the dual vectors are small. Therefore, we believe that the predictions become more accurate when there are more dual vectors, which happens, e.g., in higher dimensions, or when sieving with a larger saturation radius.

To test the robustness of our prediction for the floor phenomenon, we also ran experiments with a lattice sieve using a different saturation radius r_{sat}, which produces more (and therefore a bit longer) dual vectors than using the standard saturation radius of $\sqrt{4/3} \approx 1.1547$. In these scenarios, the floor phenomenon is observable already at a larger failure probability. Thus, fewer samples are needed to observe the floor phenomenon in the experimental data. This eases the computational power required to get the experiments, but the analysis should of course still hold in a situation where the saturation radius is larger than $\sqrt{4/3}$. This motivates Figure 4.4, which shows the predictions for a larger saturation radius. In these cases the floor phenomenon can be seen after a decent computation time, and it shows that the new predictions match the experimental data very accurately.

4.5 Conclusion

After identifying the norm of the target $\|\mathbf{t}\|$ as a confounding variable in the individual scores, we propose alternative heuristics in which this confounding variable is fixed. This leads us to predict the score distributions for both BDD targets and uniform targets in the Dual-Sieve Attack.

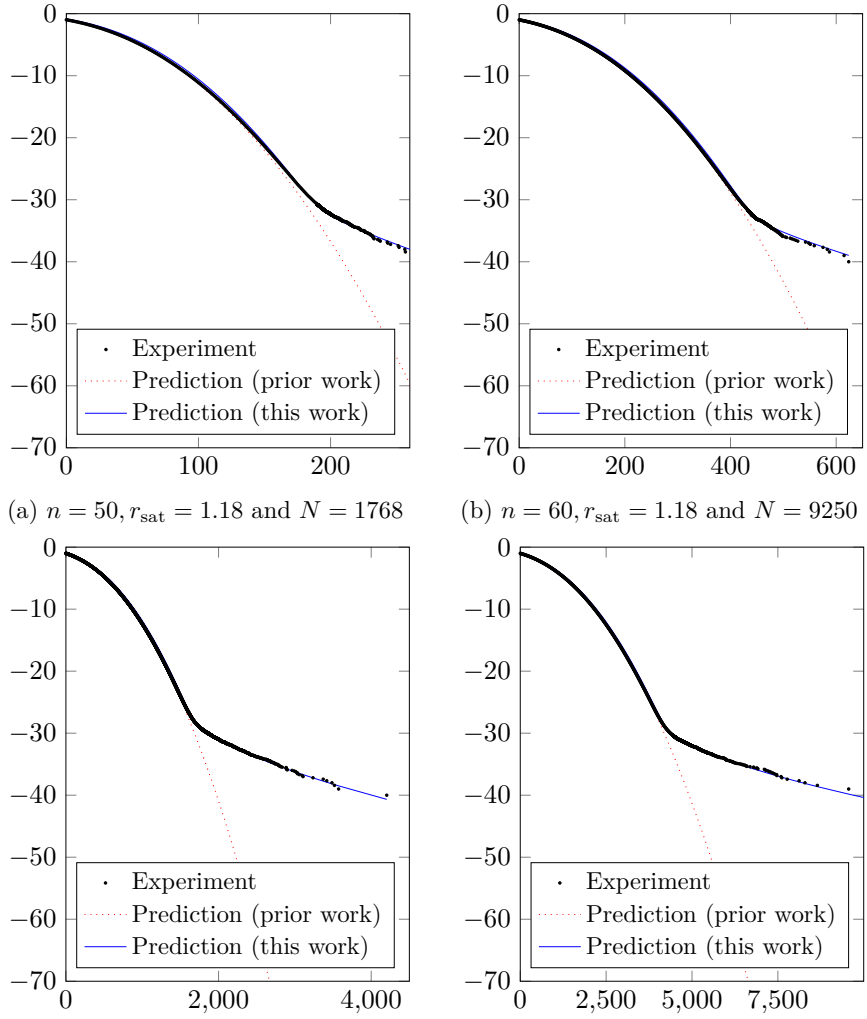
The newly derived heuristic predictions are tested in extensive experiments. Specifically, Heuristic Claim 4.15 and Heuristic Claim 4.18 experimentally show accurate predictions regarding both BDD and uniform



(a) $n = 40$ (b) $n = 50$

(c) $n = 60$ (d) $n = 70$

Figure 4.3: Binary logarithm ($\log_2$) of the survival function (SF) of the score distribution for uniform targets in various dimensions n. Each experimental curve contains 2^{48} samples, and ran a sieve with saturation ratio $f_{\text{sat}} = 0.9$ and saturation radius $r_{\text{sat}} = \sqrt{4/3}$. For $n = 40$, samples scoring below 60 are in buckets.



(a) $n = 50, r_{\text{sat}} = 1.18$ and $N = 1768$

(b) $n = 60, r_{\text{sat}} = 1.18$ and $N = 9250$

(c) $n = 70, r_{\text{sat}} = 1.20$ and $N = 157001$

(d) $n = 80, r_{\text{sat}} = 1.20$ and $N = 972103$

Figure 4.4: Binary logarithm ($\log_2$) of the survival function (SF) of the score distribution for uniform targets in various dimensions n, with a different *saturation radius*. Each experimental curve contains 2^{40} samples, and ran a sieve with saturation ratio $f_{\text{sat}} = 0.9$.

targets, compared to experimental data. In conclusion, the experiments strongly suggest that Heuristics 2.114, 4.11 and 4.17 may be used confidently in an analysis of the Dual-Sieve (with or without FFT) attack against BDD.

4.5.1 Future work

The effectiveness of the state-of-the-art dual attack remains as future work. In particular, note that existing dual attacks, e.g. [GJ21], will most likely require a different parametrisation to achieve the best runtime, while having a constant success probability, when the analysis will use the new model of Section 4.3. Moreover, there will be other aspects to the costing of the attack that may still require some attention, as highlighted in [DP23a, App. A].

On another note, theoretically it would be interesting to predict the scores for uniform targets from Section 4.3.4: instead of relying on a ball approximation of the Voronoi cell, it could be more satisfactory to adapt the Poisson model of [MT23, Assumption 8] to the lattice setting. One expects on average to have $\mathrm{Vol}_n(r\mathcal{B}^n)/\det(\Lambda)$ many lattice points at distance at most r from a target $\mathbf{t}$ sampled uniformly modulo a lattice. Whereas the code setting has a Poisson process for each weight $w = 0, 1, \ldots, n$, the situation is a bit more complex for lattices, as there is a continuum of processes to consider.

Part III
Lattice Isomorphism Problem

Chapter 5

Hawk Signature Scheme

This chapter is based on joint work with L. Ducas, E.W. Postlethwaite and W.P.J. van Woerden published at ASIACRYPT 2022 [DPPvW22a], and the third round specification document of Hawk in NIST's standardisation process of additional digital signature schemes [BBD+26].

This chapter describes the signature scheme Hawk in detail, of which security is based on a variant of the Lattice Isomorphism Problem (LIP). Hawk uses modules of rank 2 over a cyclotomic number field of order a power of 2, which allows for efficient computations. The best known attacks that can break Hawk perform not much better than attacks that forget the module structure.

The specific module lattice makes sampling extremely easy: the distributions can be precomputed and stored compactly, and floating-point numbers are not needed. These design features are desirable for constrained devices, in which it can be challenging to use FN-DSA, a.k.a. Falcon [PFH+22].

Hawk's implementation is much faster than Falcon, and its signatures are a bit more compact than those of Falcon.

5.1 Introduction

Making the transition to post-quantum alternatives as straightforward as possible, requires construction of signature schemes having multiple good characteristics. It is important that keys and signatures are small, because many protocols, such as TLS and DNSSEC, are designed with the small sizes of pre-quantum schemes in mind [MdJvH+20].

Currently, the NIST has provided standards for the following three signature schemes: FN-DSA, ML-DSA and SLH-DSA, which are based respectively on Falcon [PFH+22], CRYSTALS-Dilithium [LDK+22] and Sphincs+ [HBD+22]. Of these three, Falcon has the smallest signatures, has smaller public keys than CRYSTALS-Dilithium, and is one of the most efficient post-quantum signature schemes. Yet, the NIST recommends CRYSTALS-Dilithium for general use, because generating Falcon signatures securely and efficiently is difficult [AAC+22].

Namely, Discrete Gaussian Sampling (DGS) [Kle00, GPV08] is used in Falcon to prevent signatures from leaking the private key [NR09]. DGS has become more efficient since its introduction [DN12a, DDLL13, GM18], in particular by exploiting ideal or module structures [Pei10, DP16] such as those of NTRU lattices. Nonetheless, DGS remains particularly difficult to implement securely and efficiently, especially on constrained devices, and even more so when side-channel attacks are a concern. In particular, DGS involves high precision floating-point operations and evaluation of transcendental functions.

The difficulties of DGS can be avoided completely by using a simple lattice, such as $\mathbb{Z}^n$ [DvW22, BGPS23], and an efficient, simple sampler for such lattice. One can build cryptography by rotating the simple lattice, and making this so-called *isomorphic* lattice public, while keeping the rotation private. In this case, the security is based on the intractability of recovering an isomorphism between two isomorphic lattices i.e., the *Lattice Isomorphism Problem* (LIP).

5.1.1 Contributions

In this chapter, we consider a concrete instantiation of the LIP-based cryptography framework [DvW22]. In their introductory work on LIP, Ducas and van Woerden mostly focus on theoretical and asymptotic results [DvW22], whereas this chapter focuses on the concrete construction of a signature scheme by combining their work with simple module

lattices, and verifying its practicality.

An attractive choice would be to consider the most structured option, namely modules of rank one over number fields, i.e., ideal lattices. However, this restricted version of LIP is solvable in classical polynomial time [GS02, LS17]. Instead, we work with rank two modules. It was quickly noted that NTRUSign signatures [HHP+03] were leaking the Gram matrix of the private key; recovering the private key from this Gram matrix is precisely LIP. While the NTRUSign scheme was ultimately broken, it was only by exploiting a stronger form of leakage [NR09], not by solving LIP. In conclusion this module LIP problem is plausibly hard and is clear and simple to state, and therefore appears as a legitimate basis for cryptography.

We consider the ring $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, that is the ring of integers for some power of two cyclotomic field. While any ring of integers gets a lattice structure from its canonical embedding, this particular ring is viewed as an orthogonal lattice via the *coefficient embedding*, which allows for efficient computations. Moreover, LIP has already received some cryptanalytic attention in the case of an orthogonal lattice [Szy03].

To randomly generate a key pair, we must generate a basis for the module lattice R^2 following some distribution. We achieve this by mimicking NTRUSign key generation and setting the modulus q to 1, which allows us to use an efficient key generation technique [PP19]. Following the ideas presented by Ducas and van Woerden [DvW22], we are able to show that sampling our keys in this manner gives a worst case to average case reduction for module LIP. However, this reduction is limited to a large choice of the parameter that determines the sampling of the public key. In Hawk, we make more aggressive choices based on heuristic and experimental cryptanalysis.

The original design of Ducas and van Woerden [DvW22] hashes a message to $\left\{0, \frac{1}{q}, \ldots, \frac{q-1}{q}\right\}^{2n}$ for some q polynomial in the dimension n of the lattice. Another optimisation we propose is to hash the message to a smaller target space $\left\{0, \frac{1}{2}\right\}^{2n}$ to further simplify Gaussian sampling. For this variant we provide a reduction in the programmable random oracle model to a new problem: one more (approximate) SVP (omSVP). This reduction also requires a specific choice of parameters, and again Hawk makes more aggressive choices. This problem is similar to the recently introduced one more inhomogeneous short integer solution

Table 5.1: Performance of Falcon and Hawk for $n = 512, 1024$ on an Intel® Core™ i5-4590 @3.30GHz processor with TurboBoost disabled. Falcon and Hawk are compiled with `-O3` and `-O2`, respectively. Timings of Sign correspond to batch usage, see Section 5.4.3.

Scheme	Falcon-512	Hawk-512	Falcon-1024	Hawk-1024
AVX2 KeyGen	7.95 ms	4.25 ms	23.60 ms	17.88 ms
Ref. KeyGen	19.32 ms	13.14 ms	54.65 ms	41.39 ms
AVX2 Sign	193 µs	50 µs	382 µs	99 µs
Ref. Sign	2449 µs	168 µs	5273 µs	343 µs
AVX2 Verify	50 µs	19 µs	99 µs	46 µs
Ref. Verify	53 µs	178 µs	105 µs	392 µs

problem [AKSY22] used to design blind signature schemes from lattices.

We also propose efficient encodings for the public key and signatures of our scheme. Decoding the keys is cheap and recovering redundant parts is done efficiently with a few number theoretic transforms (NTT) or fast Fourier transforms (FFT). Moreover, we significantly compress the signature by dropping half of it, which is effectively computationally free. Decompressing a signature uses the Rounding-Off algorithm [Bab86]. This decompression uses public data during verification, so it is not a target for side-channel or statistical attacks and does not require masking. Its use of rounding also allows us to avoid the need for high precision floats.

Performance and comparison

We propose a reference implementation and an AVX2 optimised implementation. The reference implementation makes no use of floating-points and only emulates them during key generation. The AVX2 version uses floating-points.

Table 5.1 lists performance of Hawk and compares with Falcon. On CPUs supporting AVX2, Hawk-512 outperforms Falcon-512 by a factor of about 2 for key generation and verification and a factor of 4 for signing. The situation is similar for Hawk-1024. Without floats, Hawk signs 15 times faster than Falcon, because Hawk uses number

Table 5.2: Key and signature sizes in bytes of Falcon and Hawk for $n = 512, 1024$.

Scheme	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{Sig}\rvert$
Falcon-512 [PFH+22]	1281	897	652 ± 3
Hawk-512 (this work)	1153	1006 ± 6	542 ± 4
Falcon-1024 [PFH+22]	2305	1793	1261 ± 4
Hawk-512 (this work)	2561	2329 ± 11	1195 ± 6

theoretic transforms in signing while Falcon emulates floating-points. The verification contains a fast decompression that uses fixed-point arithmetic but uses two number theoretic transforms making it slightly slower than Falcon. Because Hawk uses a basis with smaller coefficients, its key generation is faster than Falcon.

Table 5.2 lists the key pair sizes and signature sizes of Falcon and Hawk. Regarding compactness, Hawk-512 signatures are about 110 bytes shorter, but public keys about 110 bytes larger, than Falcon-512; this puts Hawk-512 on par for certificate chain applications, and should be advantageous for other applications. Additionally, private keys are 128 bytes smaller. In Hawk-1024 we save a little on signatures, but our keys are larger. See Section 5.4.2 for the used encodings.

We also note that Hawk resists forgery attacks a little better than Falcon. This is a direct result of being able to use the private key to efficiently sample slightly smaller signatures in $\mathbb{Z}^{2n}$ than is possible in an NTRU lattice.

A recent variant of Falcon named Mitaka [EFG+22], also aims to make the signing procedure simpler and free from floating-point arithmetic. This is achieved with some loss in the signing quality compared to Falcon which makes signature forgeries somewhat easier, but the provided floating-point implementation signs twice as fast. In contrast, by using $\mathbb{Z}^{2n}$ we obtain an even simpler sampler while simultaneously improving the signing quality, efficiency and signature size.

Simplicity as a circuit

We claim that our signature scheme is simpler as a circuit than Falcon and therefore expect the performance gap to be larger on constrained

architectures. In fact, we hope that Hawk or a variant of Hawk may be simple enough to be implemented within a Fully Homomorphic Encryption scheme for applications such as blind or threshold signatures [ASY22]. It might also be easier to mask against side-channel attacks, similar to how the lack of floating-points in the sampler simplifies the masking of Mitaka$_{\mathbb{Z}}$ [EFG+22, Sec. 7.3].

Implementation and source code

The *isochronous* C implementation, experimental data, and auxiliary scripts are open source:

https://github.com/ludopulles/hawk-aux

5.2 Scheme

In this section we present Hawk. We first give a version of Hawk that performs no compression on its signatures for simplicity, we call this uncompressed Hawk. We then introduce (compressed) Hawk and discuss how the security of Hawk directly reduces to that of the uncompressed Hawk. The auxiliary script `code/hawk.sage` is a proof of concept SageMath implementation of Hawk.

5.2.1 Uncompressed Hawk

The uncompressed version of our signature scheme is based on the scheme presented in [DvW22, Sec. 6], but is adapted to number rings for efficiency. The scheme uses the number ring $R = \mathbb{Z}[X]/(X^n + 1)$ with $n \geq 2$ a power of two, the ring of integers of the number field $\mathbb{Q}(\zeta_{2n})$. We use the simplest rank 2 module lattice, $R^2 \cong \mathbb{Z}^{2n}$. We implicitly move between R^2 and $\mathbb{Z}^{2n}$ via the coefficient embedding. The private key is some basis $\mathbf{B} \in \mathrm{SL}_2(R)$ where $\mathbf{B}$ (resp. $\mathrm{Rot}(\mathbf{B})$) generates R^2 (resp. $\mathbb{Z}^{2n}$). In the context of [DvW22, Sec. 6] this matrix represents a basis transformation applied to the trivial basis $\mathbf{I}_2(\mathbb{K})$ of R^2. The public key is the Hermitian form $\mathbf{Q} = \mathbf{B}^\star \cdot \mathbf{B}$ associated to the basis $\mathbf{B}$. A signature on a message m is generated by first hashing m and a salt r to a point $\mathbf{h} = (h_0, h_1)^\mathsf{T} \in \{0,1\}^{2n}$. Applying the transformation $\mathbf{B}$ to $\frac{1}{2}\mathbf{h}$ gives us a target $\frac{1}{2}\mathbf{B} \cdot \mathbf{h}$. We then sample a short element $\mathbf{x}$ in the target's coset $R^2 + \frac{1}{2}\mathbf{B} \cdot \mathbf{h}$ via discrete Gaussian samples on $\mathbb{Z}$ and

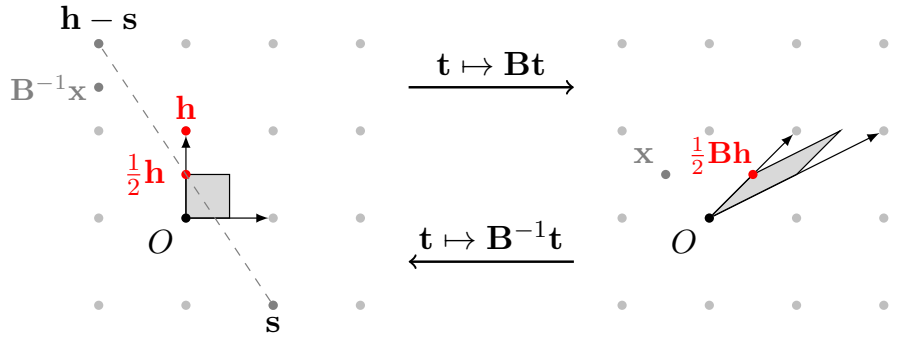


Figure 5.1: Illustration of signing. First $\mathbf{h}$ is sampled (left), then $\mathbf{B}$ is applied, a short lattice point $\mathbf{x}$ is sampled from a discrete Gaussian on $\mathbb{Z}^{2n} + \frac{1}{2}\mathbf{B} \cdot \mathbf{h}$ (right). Finally, $\mathbf{B}^{-1}$ applied to $\mathbf{x}$ is subtracted from $\frac{1}{2}\mathbf{h}$ to obtain a lattice point $\mathbf{s}$ close to $\mathbf{h}/2$ in $\|\cdot\|_{\mathbf{Q}}$, and we update $\mathbf{s}$ to $\mathbf{h} - \mathbf{s}$ to ensure SymBreak$(h_1 - 2s_1)$ is satisfied.

$\mathbb{Z} + 1/2$. By applying the inverse transformation $\mathbf{B}^{-1}$ we compute the signature $\mathbf{s} = \frac{1}{2}\mathbf{h} \pm \mathbf{B}^{-1}\mathbf{x} \in R^2$. This is close to $\frac{1}{2}\mathbf{h}$ with respect to $\|\cdot\|_{\mathbf{Q}}$, and the sign is chosen to prevent weak forgeries, see Algorithm 5.2 and below. See Figure 5.1 for a visualisation when $n = 1$. Verification checks if the distance $\left\|\frac{\mathbf{h}}{2} - \mathbf{s}\right\|_{\mathbf{Q}}$ between $\mathbf{s}$ and $\frac{1}{2}\mathbf{h}$ is not too large, which only requires the public key $\mathbf{Q} = \mathbf{B}^{\star}\mathbf{B}$ and not the private key $\mathbf{B}$. We have the following parameters:

1. σ_{pk}: controls the length of $(f, g)^{\mathsf{T}}$, the first basis vector of $\mathbf{B}$,
2. σ_{sec}: controls the lower bound on the acceptable length of $(f, g)^{\mathsf{T}}$,
3. σ_{sign}: controls the length of a short coset vector,
4. σ_{ver}: controls the acceptable distance from a signature to a halved hash,
5. saltlen: controls the probability of hash collisions.

For uncompressed Hawk we present KeyGen in Algorithm 5.1, Sign in Algorithm 5.2 and Verify in Algorithm 5.3. The security parameter n is a power of two, and we assume the internal parameters can be computed from it. We use previous work to generate the unimodular transformation $\mathbf{B}$ efficiently [PP19, Alg. 4], by sampling the first basis

Algorithm 5.1. KeyGen(1^n): Hawk key generation

Input: parameter n
Output: a Hermitian form $\mathbf{Q} = \mathbf{B}^\star\mathbf{B}$, and the corresponding (randomly generated) basis $\mathbf{B} \in \mathrm{SL}_2(R)$ for R^{2n}

1: sample $f, g \in R$ with coefficients from $D_{\mathbb{Z},\sigma_{\mathsf{pk}}}$
2: $q_{00} = f^\star f + g^\star g$
3: **if** $2 \mid \mathrm{N}(f)$ **or** $2 \mid \mathrm{N}(g)$ **or** $\|(f,g)\|^2 \le \sigma_{\mathsf{sec}}^2 \cdot 2n$ **then**
4: **restart**
5: $(F,G)^\mathsf{T} \leftarrow \mathrm{TowerSolveI}_{n,1}(f,g)$ [PP19, Alg. 4], **if** $\perp$, **restart**
6: $\begin{pmatrix} F \\ G \end{pmatrix} \leftarrow \begin{pmatrix} F \\ G \end{pmatrix} - \mathrm{ffNP}_R\left(\frac{f^\star F + g^\star G}{q_{00}}, \mathrm{ffLDL}_R^\star(q_{00})\right) \cdot \begin{pmatrix} f \\ g \end{pmatrix}$ [DP16]
7: $\mathbf{B} = \begin{pmatrix} f & F \\ g & G \end{pmatrix}$
8: $\mathbf{Q} = \begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix} = \mathbf{B}^\star \cdot \mathbf{B}$
9: **return** $(\mathsf{pk}, \mathsf{sk}) = (\mathbf{Q}, \mathbf{B})$

Algorithm 5.2. $\mathrm{Sign}_{\mathbf{B}}(\mathsf{m})$: Hawk signature generation

Input: private key $\mathbf{B}$ and message $\mathsf{m} \in \{0,1\}^*$
Output: a (randomly generated) valid signature $\mathsf{sig} \in \{0,1\}^{\mathsf{saltlen}} \times R^2$

1: $\mathsf{r} \leftarrow U(\{0,1\}^{\mathsf{saltlen}})$
2: $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{m} \| \mathsf{r})$
3: $\mathbf{t} \leftarrow \mathbf{B} \cdot \mathbf{h} \pmod 2$
4: $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^{2n} + \frac{1}{2}\mathbf{t}, \sigma_{\mathsf{sign}}}$
5: **if** $\|\mathbf{x}\|^2 > \sigma_{\mathsf{ver}}^2 \cdot 2n$ **then**
6: **restart** (optional, see Section 5.5.2)
7: $\mathbf{s} = (s_0, s_1)^\mathsf{T} \leftarrow \frac{1}{2}\mathbf{h} - \mathbf{B}^{-1}\mathbf{x}$ (parse $\mathbf{x} \in \frac{1}{2}R^2$ via Vec^{-1})
8: **if** $\mathrm{SymBreak}(h_1 - 2s_1)$ is `False` **then**
9: $\mathbf{s} \leftarrow \mathbf{h} - \mathbf{s}$
10: **return** $\mathsf{sig} = (\mathsf{r}, \mathbf{s})$

Algorithm 5.3. $\textsc{Verify}_{\mathbf{Q}}(\mathsf{m}, \mathsf{sig})$: Hawk signature verification

Input: public key $\mathbf{Q}$, message $\mathsf{m} \in \{0,1\}^*$ and signature sig
Output: 1 if sig is a valid signature on m, otherwise 0

1: $(\mathsf{r}, \mathbf{s}) \leftarrow \mathsf{sig}$
2: $\mathbf{h} \leftarrow \mathsf{H}(\mathsf{m} \| \mathsf{r})$
3: **return** $\left[\!\!\left[\begin{array}{l} \mathbf{s} \in R^2 \textbf{ and } \left\| \frac{\mathbf{h}}{2} - \mathbf{s} \right\|_{\mathbf{Q}}^2 \leq \sigma_{\mathsf{ver}}^2 \cdot 2n \\ \textbf{and } \textsc{SymBreak}(h_1 - 2s_1) \text{ is } \texttt{True} \end{array} \right]\!\!\right]$

vector $(f, g)^{\mathsf{T}}$, and, if possible, completing it with a second basis vector $(F, G)^{\mathsf{T}}$ such that $\det(\mathbf{B}) = 1$. We combine this with the fast Babai's Nearest-Plane reduction [DP16] to obtain a shorter second basis vector $(F, G)^{\mathsf{T}}$. In KeyGen checks are performed prior to completing the basis $\mathbf{B}$. In TowerSolveI it is necessary for $\mathrm{N}(f)$ or $\mathrm{N}(g)$ to be an odd integer [PP19]. We require both to be odd to use an optimised constant-time greatest common divisor algorithm, identical to the Falcon reference implementation. Also, we require the squared norm of $(f, g)^{\mathsf{T}}$ to be longer than $\sigma_{\mathsf{sec}}^2 \cdot 2n$ for our concrete cryptanalysis, see Section 5.3. Note that a signer also has easy access to $\mathbf{B}^{-1}$, because

$$\mathbf{B}^{-1} = \begin{pmatrix} G & -F \\ -g & f \end{pmatrix}, \tag{5.1}$$

follows from $fG - gF = \det(\mathbf{B}) = 1$.

In Sign and Verify we check the condition $\textsc{SymBreak}(h_1 - 2s_1)$, which is required for strong unforgeability. Without it $\mathsf{sig}' = (\mathsf{r}, \mathbf{h} - \mathbf{s})$, which can be constructed from public values, is another valid signature on m if $\mathsf{sig} = (\mathsf{r}, \mathbf{s})$ is. Given $e \in R$, we define $\textsc{SymBreak}(e)$ to be $\texttt{True}$ if and only if $e \neq 0$ and the first nonzero coefficient of $\textsc{Vec}(e)$ is positive. Checking this condition on $h_1 - 2s_1$ in Verify prevents a weak forgery attack.

Signature correctness

Assume Sign is called with message m and outputs $\mathsf{sig} = (\mathsf{r}, \mathbf{s})$. First, note $\mathbf{B}^{-1}\mathbf{x} \in R^2 + \frac{1}{2}\mathbf{h}$ and $\frac{1}{2}\mathbf{h} \pm \frac{1}{2}\mathbf{h} \in R^2$, so $\mathbf{s} = \frac{1}{2}\mathbf{h} \pm \mathbf{B}^{-1}\mathbf{x} \in R^2$. Second, suppose $\textsc{SymBreak}(h_1 - 2s_1)$ is not satisfied during verification.

By lines 8 and 9 of Algorithm 5.2, this means SymBreak$(h_1 - 2s_1)$ and SymBreak$(2s_1 - h_1)$ are both `False`, therefore $h_1 = 2s_1$. Since $\mathbf{h} \in \{0,1\}^{2n}$, this implies $h_1 = 0$, i.e., we have found a preimage of $(h_0, 0)^\mathsf{T}$ for H. By choosing a preimage resistant H or modelling it as a random oracle, this happens with $\mathsf{negl}(n)$ probability. We allow this failure probability to simplify (compressed) Hawk.

The signing algorithm terminates only if the condition on Line 5 is `False`. Therefore, $\|\mathbf{x}\|^2 \leq \sigma_{\mathsf{ver}}^2 \cdot 2n$. Thus, during verification $\|\frac{\mathbf{h}}{2} - \mathbf{s}\|_{\mathbf{Q}}^2 = \|\mathbf{B}\left(\frac{\mathbf{h}}{2} - \mathbf{s}\right)\|^2 = \|\pm\mathbf{x}\|^2 \leq \sigma_{\mathsf{ver}}^2 \cdot 2n$, with $-\mathbf{x}$ given by Line 9 of Sign.

Key storage

We now consider how to efficiently store pk and sk, that is,

$$\mathbf{Q} = \begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix} = \mathbf{B}^\star \cdot \mathbf{B} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} f & F \\ g & G \end{pmatrix}$$

respectively. For $\mathbf{B}$ it is sufficient to only store f and g, but this requires the computationally expensive recovery of F and G in Sign. We note that computing F, G is the most expensive part of KeyGen. Instead, one stores f, g and F since G can be recovered efficiently from $fG - gF = 1$. The coefficients of f, g and F are small so we use a simple encoding with constant-time decoding for them.

For $\mathbf{Q}$ by construction we have $q_{10} = q_{01}^\star$, so one may simply drop q_{10}. Moreover, since $\det(\mathbf{B}) = 1$ we have $q_{00}q_{11} - q_{01}q_{10} = \det(\mathbf{Q}) = \det(\mathbf{B}^\star)\det(\mathbf{B}) = 1$, therefore q_{11} can be dropped and reconstructed as $q_{11} = (1 + q_{01}^\star \cdot q_{01})/q_{00}$. In addition, q_{00} is self-adjoint and therefore only the first half of its coefficients need to be encoded. More details are given in Section 5.4.2.

5.2.2 (Compressed) Hawk

Hawk is obtained by dropping s_0 from a signature $\mathbf{s} = (s_0, s_1)^\mathsf{T}$ in Sign and then reconstructing it in Verify using public values $\mathbf{Q}$ and $\mathbf{h}$. There is a probability that s_0 is not correctly recovered, but it is kept small by rejecting 'bad' key pairs in KeyGen. In Verify, s_0 is recovered by finding a value that makes $\frac{1}{2}\mathbf{h} - (s_0, s_1)^\mathsf{T}$ short with respect to $\|\cdot\|_{\mathbf{Q}}$. Two ways to reconstruct s_0 are Rounding-Off and Babai's Nearest-Plane algorithm [Bab86]. Given that we work with respect to the norm induced

by $\mathbf{Q}$, we must adapt one of these algorithms to quadratic forms. Because of its simplicity and good performance, we use Rounding-Off for Hawk. Specifically, to reconstruct s_0, we use,

$$s_0' = \left\lceil \frac{h_0}{2} + \frac{q_{01}}{q_{00}} \left(\frac{h_1}{2} - s_1 \right) \right\rfloor , \tag{5.2}$$

where the rounding map $\lceil \cdot \rfloor \colon \mathbb{K} \to R$ acts coefficientwise, and satisfies $x - \lceil x \rfloor \in (-1/2, 1/2]$ for all $x \in \mathbb{R}$. Hence, Verify is adapted to read a signature $\mathsf{sig} = (\mathsf{r}, s_1)$ and reconstruct s_0' using Equation (5.2), before setting $\mathbf{s} = (s_0', s_1)^{\mathsf{T}}$. Observe that $s_0' = s_0$ holds if and only if

$$\left\lceil \frac{q_{00} \left(\frac{h_0}{2} - s_0 \right) + q_{01} \left(\frac{h_1}{2} - s_1 \right)}{q_{00}} \right\rfloor = 0.$$

The fraction inside $\lceil \cdot \rfloor$ is equal to $\frac{f^\star x_0 + g^\star x_1}{f^\star f + g^\star g}$ because we have $(q_{00}, q_{01}) = (f^\star, g^\star) \cdot \mathbf{B}$ and $\mathbf{B} \cdot \left(\frac{\mathbf{h}}{2} - \mathbf{s} \right) = (x_0, x_1)^{\mathsf{T}}$. Thus, we certainly recover the correct s_0 from $\mathbf{Q}$, $\mathbf{h}$ and s_1 if

$$\frac{f^\star x_0 + g^\star x_1}{f^\star f + g^\star g} \in \left(-\frac{1}{2}, \frac{1}{2} \right)^n . \tag{5.3}$$

Intuitively, when (f, g) is sampled such that the Euclidean norm of $(f^\star/q_{00}, g^\star/q_{00})$ is sufficiently small, this recovery works almost always. Note

$$\left\| \left(\frac{f^\star}{q_{00}}, \frac{g^\star}{q_{00}} \right) \right\|^2 = \frac{1}{n} \operatorname{Tr}\left(\frac{f^\star f + g^\star g}{q_{00}^2} \right) = \frac{1}{n} \operatorname{Tr}\left(q_{00}^{-1} \right) = \langle 1, q_{00}^{-1} \rangle. \tag{5.4}$$

Hence, we choose a bound ν_{dec} such that decompression works almost always for keys satisfying $\langle 1, q_{00}^{-1} \rangle < \nu_{\mathsf{dec}}$. We provide a computation and value for ν_{dec} in Section 5.4.1. In summary, Algorithms 5.1, 5.2 and 5.3 are changed as follows for Hawk.

1. In KeyGen, restart if $\langle 1, q_{00}^{-1} \rangle \geq \nu_{\mathsf{dec}}$.
2. In Sign, restart if $\mathbf{x} = (x_0, x_1)^{\mathsf{T}}$ does not satisfy Equation (5.3).
3. In Sign, return a signature as (r, s_1) instead of $(\mathsf{r}, \mathbf{s}) = (\mathsf{r}, (s_0, s_1)^{\mathsf{T}})$.
4. In Verify, given a signature (r, s_1) reconstruct s_0' using Equation (5.2) and set $\mathbf{s} = (s_0', s_1)^{\mathsf{T}}$.

Given the above, a reconstructed signature is correct. In practice, we choose ν_{dec} such that Equation (5.3) holds except with small probability and forego item 2. in the list, see Section 5.5.

Security relation to the uncompressed variant

Note that an adversary that can create a forgery (strong or weak) against Hawk can also create a forgery (strong or weak) against uncompressed Hawk. Indeed, if $\mathsf{sig} = (\mathsf{r}, s_1)$ is a forgery against Hawk then this implies $\mathsf{sig} = (\mathsf{r}, (s_0', s_1)^{\mathsf{T}})$ is a forgery against uncompressed Hawk. Only public quantities are required to recover s_0'. Therefore, throughout we consider the security of uncompressed Hawk. In Section 5.6.3 we further reduce the forgery security of uncompressed Hawk to an assumption called the one more short vector problem, or omSVP.

5.3 Cryptanalysis

In this section we provide a concrete cryptanalysis of Hawk. Whereas the formal security arguments we make in Sections 5.6.1 and 5.6.3 increase our confidence in the design of Hawk, our results here aid us in choosing parameter sets that are efficient. Throughout we consider uncompressed Hawk. We consider recovering the private key from public information and forging a new signature given at most $q_s = 2^{64}$ signatures. Furthermore, we report the various parameters, probabilities and block sizes output by our cryptanalysis in Table 5.3.

The constraints on various quantities in our scheme are expressed in terms of Gaussian parameters $\sigma_\bullet$, even if they are not quantities sampled from a distribution. This allows us to present necessary relationships between these quantities as in Figure 5.2. In particular for $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^d,\sigma}$, with σ above smoothing, $\mathbb{E}[\|\mathbf{x}\|] \approx \sigma\sqrt{d}$ [MR07, Sec. 4]. For example, the verification of signatures is determined by a distance, say ℓ. Instead of referring to ℓ, we use the shorthand $\sigma_{\mathsf{ver}} = \ell/\sqrt{d}$.

We stress that the relations of Figure 5.2 are necessary, under our experimental analysis, conditions for security – any selection must also satisfy the concrete cryptanalysis below. As a short introduction, σ_{sign} is our fundamental parameter, and we select it first. It ensures that our scheme does not suffer from learning attacks [NR09, DN12b] if an adversary is given access to signature transcripts. We then choose σ_{pk} which controls key generation in Algorithm 5.1. It must be large enough that recovering the private key is hard, and also that the cost of computing a sufficiently good basis to aid with signature forgeries is hard. To this end we heuristically estimate and verify experimentally

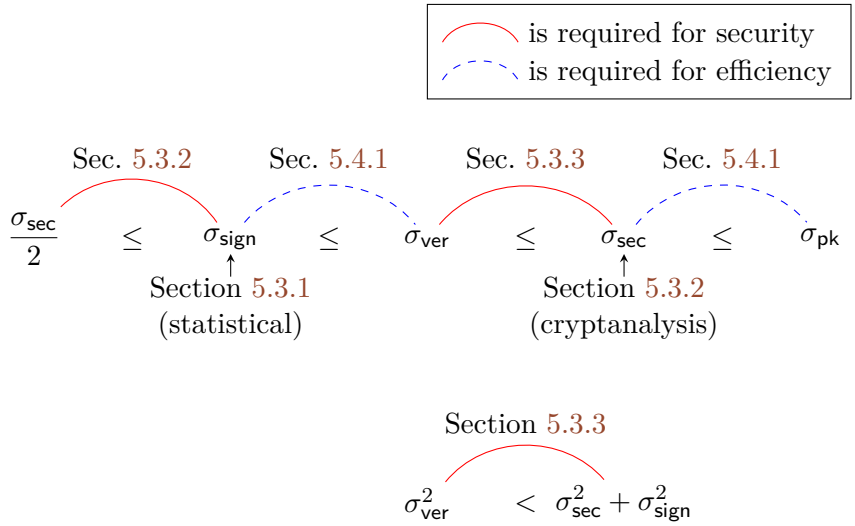


Figure 5.2: Summary of necessary relationships between various $\sigma_\bullet$

σ_{sec}, a parameter that represents the shortest a public basis can be before one recovers the private key. Finally, σ_{pk} and σ_{ver} are chosen to ensure various rejection steps, in key generation and signing respectively, do not occur too frequently, see Section 5.4.1. The condition $\sigma_{\mathsf{ver}}^2 < \sigma_{\mathsf{sec}}^2 + \sigma_{\mathsf{sign}}^2$ encodes the requirement for a good basis to not help with signature forgeries.

5.3.1 Signature sampling

We choose σ_{sign} large enough to avoid signatures leaking information about a private key. Following the methodology in Falcon [PFH+22, Sec. 2.6], for security parameter λ in the face of an adversary allowed q_s signatures, it is enough to set

$$\sigma_{\mathsf{sign}} \geq \frac{1}{\pi} \cdot \sqrt{\frac{\log(4n(1+1/\varepsilon))}{2}} \geq \eta_\varepsilon\left(\mathbb{Z}^{2n}\right),$$

for $\varepsilon = 1/\sqrt{q_s \cdot \lambda}$ to lose a small constant number of bits of security. We note that since we sample from $\mathbb{Z}^{2n}$ we may use the orthogonal basis $\mathbf{I}_{2n}(\mathbb{Z})$, and thus the above inequality is also sufficient for efficient sampling via Lemma 2.81. We ensure that, following the analysis of Falcon [PFH+22, Sec. 3.9.3], our probability distribution tables have

a Rényi divergence at order 513 from their ideal distributions of less than $1 + 2^{-79}$. These computations are done in the auxiliary script `code/generate_C_tables.py`.

5.3.2 Key recovery

In Hawk, the problem of recovering the private key $\mathbf{B} \in \mathrm{SL}_2(\mathcal{O}_\mathbb{K})$ from the public key $\mathbf{Q} = \mathbf{B}^\star \cdot \mathbf{B}$ is a (module) Lattice Isomorphism Problem. For the lattice $R^2 \cong \mathbb{Z}^{2n}$ it is equivalent to finding a $\mathbf{U} \in \mathrm{SL}_2(\mathcal{O}_\mathbb{K})$ such that $\mathbf{U}^\star \cdot \mathbf{Q} \cdot \mathbf{U} = \mathbf{I}_2(\mathbb{K})$, i.e., reducing (any lattice basis corresponding to) $\mathbf{Q}$ to an orthonormal basis. As mentioned in [DvW22], all known algorithms to solve LIP for modules of rank at least two require finding at least one shortest vector. Therefore, we assume that the best key recovery attack requires one to find a single shortest vector.

Unusual-SVP

The shortest vectors in R^2 have length 1, which is a factor of order $\Theta(\sqrt{n})$ shorter than predicted by the Gaussian heuristic. Recovering such 'unusually' short vectors is easier than generic shortest vectors, and can be achieved by running the BKZ lattice reduction algorithm with block size β much lower than the full dimension $2n$. Given that for current cryptanalysis there are no significant speed-ups for solving the structured variant of this unusual-SVP, we treat the problem by considering the unstructured version, i.e., as the form $\mathrm{Rot}(\mathbf{Q})$ or some rotation of $\mathbb{Z}^{2n}$. The problem of finding an unusually short vector has received much cryptanalytic attention. This has lead to accurate estimates for the required BKZ block size, see [AD21] for a survey. As an estimate, given that our lattice has unit volume, and we search for a vector of length one, we require a block size β such that

$$\sqrt{\beta/d} \approx \delta_\beta^{2\beta-d-1}, \tag{5.5}$$

where $\delta_\beta \approx (\beta/2\pi e)^{1/2(\beta-1)}$. Asymptotically this is satisfied for some $\beta \in d/2 + o(d)$. Concrete estimates also simulate the Gram–Schmidt profile, use probabilistic models for the lengths of projected vectors and account for the presence of multiple shortest vectors [CN11, BSW18, DDGR20, PV21].

In Figure 5.3 we plot the estimate given by Equation (5.5) where the $o(d)$ term is concretised to some constant, the estimate given by the leaky-LWE-estimator [DDGR20], which applies the concrete improvements mentioned above, and experimental data. These experiments apply the BKZ 2.0 algorithm with lattice point enumeration [CN11] as implemented in fplll [FPL24] to the public form $\mathbf{Q}$, reporting the BKZ block size required to find a shortest vector. For dimensions which are not powers of two, the experimental data uses a form sampled by [DvW22, Alg. 1], the unstructured generation procedure upon which our key generation is based. For some small power of two dimensions we generate bases via Algorithm 5.1. We see that below approximately dimension 80 instances can be solved with LLL reduction [LLL82], and that afterwards the required block size approximately increments by one when the dimension increases by two, as Equation (5.5) would suggest. Moreover, we see that above approximately block size 70 the model of Dachman-Soled et al. [DDGR20] appears especially accurate. Therefore, we use this model to determine β_{key} in Table 5.3. We use a simple progressive strategy where the block size increments by one after each tour, which we expect to require a block size perhaps two or three larger than a more optimal progressive strategy.

Minimal basis randomisation

For the experiments of Figure 5.3 we took a large σ_{pk} as an attempt to find a ground truth. We would like to minimise σ_{pk} to minimise the size of our keys and the complexity of computing with them, but without significantly reducing security. To this end we perform a similar experiment where we fix a set of dimensions and reduce forms of these dimensions using various $\sigma_{\mathsf{pk}} < 20$. The results of these experiments are presented in Figure 5.4. For σ_{pk} below a certain threshold, instances can be solved by LLL. Then as σ_{pk} increases past this threshold the instances become harder, before reaching an empirical 'maximum hardness' (at least with respect to these experiments) where further increases in σ_{pk} appear to give no extra security.

When running BKZ one encounters shorter vectors as β grows. In a random unit-volume lattice, one expects to encounter vectors of length $\delta_\beta^{d-1} \in \Theta(d)$ when $\beta = d/2 + o(d)$, but for $\mathbb{Z}^d$ this is also the moment that a vector of length 1 is found. In fact, the model of [DDGR20] predicts that we suddenly jump from finding vectors of length $\ell_0 = \Theta(d)$

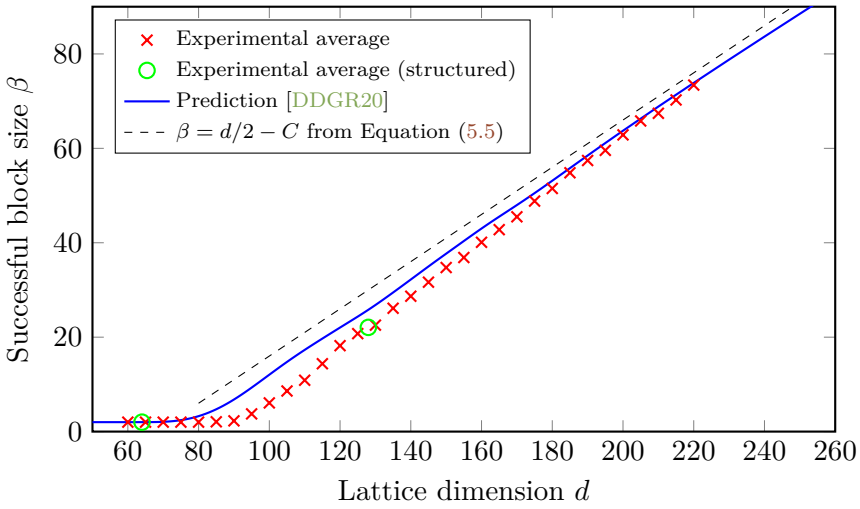


Figure 5.3: Block size required to recover a shortest vector via lattice reduction as a function of dimension d. We ran progressive BKZ (one tour per block size) over $\mathbb{Z}^d$ using an input form generated with $\sigma_{\mathsf{pk}} = 20$ and report the average successful β that recovered a length one vector over 40 instances. We used the BKZ simulator and probabilistic model of Dachman-Soled et al. [DDGR20], accounting for the d target solutions. See auxiliary scripts `code/BKZ_simulator.sage` and `code/exp_varying_n.sage`.

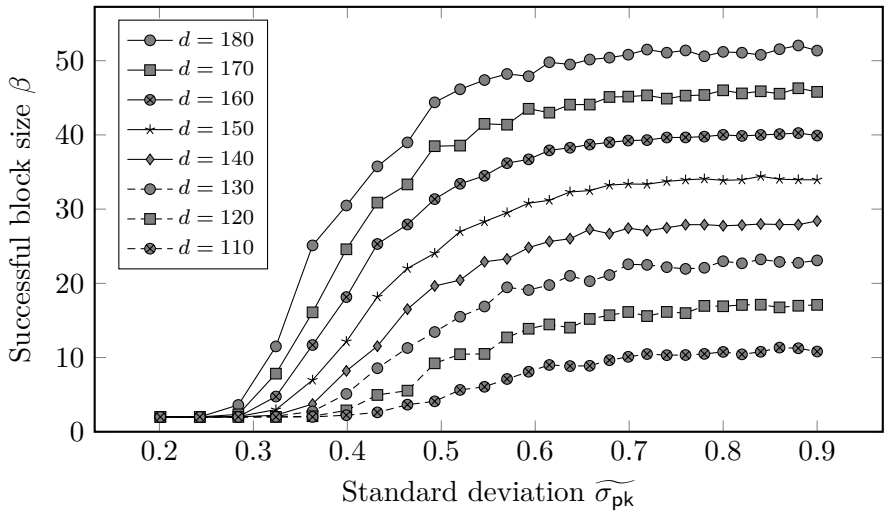


Figure 5.4: Block size required to recover a shortest vector via lattice reduction as a function of the standard deviation $\widetilde{\sigma_{\mathsf{pk}}}$. We ran progressive BKZ (one tour per block size) over $\mathbb{Z}^d$ using an input form generated with various σ_{pk} and report the average successful β that recovered a length one vector over 80 instances. Note that the range of σ_{pk} includes values below smoothing, for which the actual standard deviation $\widetilde{\sigma_{\mathsf{pk}}}$ can be significantly lower than the Gaussian parameter σ_{pk}. See auxiliary script `code/exp_varying_sigma.sage`.

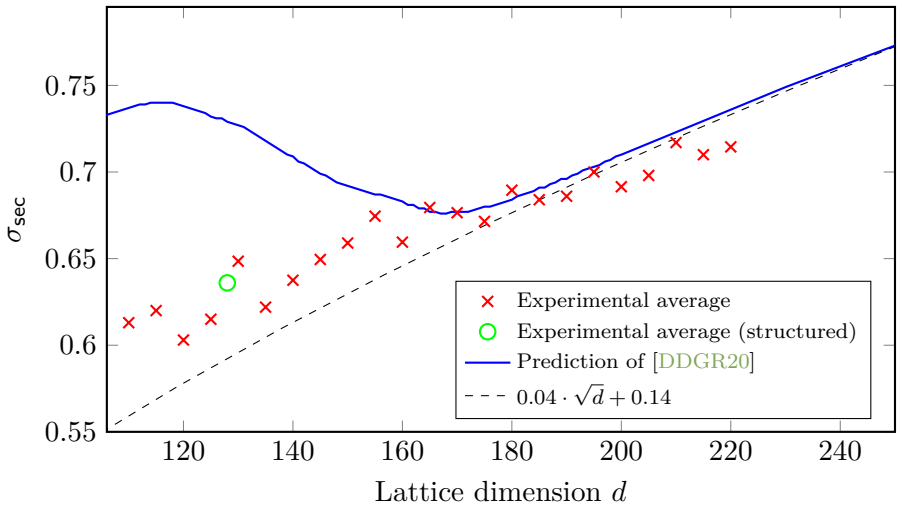


Figure 5.5: Shortest basis vector length before the recovery of a length one vector, given in terms of σ_{sec}. We ran progressive BKZ (one tour per block size) over $\mathbb{Z}^d$ using an input form generated with $\sigma_{\mathsf{pk}} = 20$ and report the average σ_{sec} determined by the shortest basis vector in the penultimate BKZ tour over 40 instances. See auxiliary scripts `code/exp_varying_n.sage` and `code/predict_varying_n.sage`.

to finding a shortest vector of length 1. This threshold behaviour was observed and discussed in [BGPS23, Sec. 6.2]. In our notation the authors observe the threshold effect once vectors of length approximately $\sqrt{d}/2$ are discovered. Our model and experiments suggest the threshold length is $\Theta(d)$ but with a constant smaller than 1. We verified this behaviour for $\mathbb{Z}^d$ experimentally. In Figure 5.5 we plot $\sigma_{\mathsf{sec}} = \ell_0/\sqrt{d}$ where ℓ_0 is the length of the shortest basis vector after the penultimate tour concludes.

We see that for large enough dimensions the behaviour matches the unusual-SVP predictions, and we obtain $\sigma_{\mathsf{sec}} = \Theta(\sqrt{d})$. For Table 5.3 we take σ_{sec} as the output from the prediction of [DDGR20]. We assume the value σ_{sec} represents a lower bound for σ_{pk} such that our public forms exhibit maximum hardness. In practice, we take $\sigma_{\mathsf{pk}} > \sigma_{\mathsf{sec}}$ and reject keys where the length of $(f, g)^{\mathsf{T}}$ is shorter than ℓ_0. If we allow shorter $(f, g)^{\mathsf{T}}$ then the public key may give information to an adversary that she would not have unless she had already recovered the private key.

We note that the prediction of [DDGR20] in Figure 5.5 is inaccurate for $d \leq 180$, similarly (but more noticeably) to Figure 5.3. One can

improve the accuracy of estimates for these dimensions by using the geometric series assumption, and performing several tours so that basis profiles match it, for small block sizes (say up to $\beta = 20$). Since our estimates converge in the range of feasible experiments, we choose simplicity instead.

Note that even if the statistical arguments of Section 5.3.1 allow it, we cannot take $\sigma_{\mathsf{sign}} < \sigma_{\mathsf{sec}}/2$. Indeed, $2 \cdot (\frac{1}{2}\mathbf{h} - \mathbf{s}) \in \mathbb{Z}^{2n}$ and if $\sigma_{\mathsf{sign}} < \sigma_{\mathsf{sec}}/2$ then $\|2 \cdot (\frac{1}{2}\mathbf{h} - \mathbf{s})\|_{\mathbf{Q}} = 2\|\mathbf{x}\| \approx 2\sigma_{\mathsf{sign}}\sqrt{d} < \sigma_{\mathsf{sec}}\sqrt{d}$. Therefore, doubling a public quantity given by a signature may describe a shorter lattice vector than those seen just before private key recovery.

5.3.3 Signature forgery

Strong forgery

We consider the general problem of forging a signature on some unsigned message. Specifically, given a target $\frac{1}{2}\mathbf{h}$ for some $\mathbf{h} \in \{0,1\}^{2n}$, return an $\mathbf{s} \in \mathbb{Z}^{2n}$ such that $\|\frac{1}{2}\mathbf{h} - \mathbf{s}\|_{\mathbf{Q}} \leq \sigma_{\mathsf{ver}}\sqrt{d}$. We use the heuristic that solving such an approximate CVP instance is at least as hard as solving an approximate SVP instance with the same approximation factor over the same lattice. We determine the expected block size β using the BKZ simulator [DDGR20] such that our first basis vector has norm less than $\sigma_{\mathsf{ver}}\sqrt{d}$, and report it as β_{forge} in Table 5.3. Note that since we use the BKZ simulator of [DDGR20] to estimate both β_{key} and β_{forge}, if $\sigma_{\mathsf{ver}} = \sigma_{\mathsf{sec}}$ then $\beta_{\mathrm{key}} = \beta_{\mathrm{forge}}$. Our approach mandates that $\sigma_{\mathsf{ver}} \leq \sigma_{\mathsf{sec}}$. We make this design decision because in our model it means an adversary should not be able to produce a strong forgery unless the private key is recovered. Indeed, when $\sigma_{\mathsf{ver}} \leq \sigma_{\mathsf{sec}}$ our assumption on approximate CVP says forging a signature is as hard as finding a vector as short as those found just before key recovery.

Weak forgery

We consider a weak forgery attack consisting of adding a short lattice vector to an existing signature on some message, and hoping that it remains a valid signature on the same message. This vector might come from the public key, or from lattice reduction effort on it. Its length is assumed to be at least $\ell_0 = \sigma_{\mathsf{sec}} \cdot \sqrt{d}$, see Section 5.3.2. We give arbitrarily many such length ℓ_0 vectors to the adversary for free. Using the function

`forgeProbability` from the auxiliary script `code/find_params.sage`, we estimate the probability the attack succeeds, i.e., that $\|\mathbf{x}+\mathbf{v}\|^2 \leq d \cdot \sigma_{\mathsf{ver}}^2$ for $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^{2n},\sigma_{\mathsf{sign}}}$ and $\mathbf{v}$ of length ℓ_0. If $\mathbf{x}$ were sampled from a spherical continuous Gaussian, then considering any such $\mathbf{v}$ would give the same distribution of squared lengths. We examine the distribution of $\|\mathbf{x}+\mathbf{v}\|^2$ for two 'extremal' choices of $\mathbf{v}$; the first has all its weight in one coordinate, $\mathbf{v} = (\lfloor \ell_0 \rfloor, 0, \ldots, 0)$, and the second is as balanced as possible, e.g. $\mathbf{v} = (1, \ldots, 1, 2, \ldots, 2)$ for $\|\mathbf{v}\| = \lfloor \ell_0 \rfloor$. Note that the distribution of $\|\mathbf{x}+\mathbf{v}\|^2$ is invariant under signed permutations of $\mathbf{v}$. We report our estimate for the success probability of this attack as Pr[weak forgery] in Table 5.3.

This attack implies the requirement $\sigma_{\mathsf{ver}}^2 < \sigma_{\mathsf{sec}}^2 + \sigma_{\mathsf{sign}}^2$. Even if a vector from a reduced public key is orthogonal to a given signature, then if $\sigma_{\mathsf{ver}}^2 \geq \sigma_{\mathsf{sec}}^2 + \sigma_{\mathsf{sign}}^2$ adding it will likely be sufficient for a weak forgery.

Comparison with Falcon

Falcon uses a different cryptanalytic model to determine the block sizes reported in Table 5.3. Our model makes use of recent improvements [DDGR20] and enjoys the experimental evidence above. The unusually short vectors in $\mathbb{Z}^d$ are a factor about 1.17 shorter than the shortest vector in the NTRU lattice of Falcon, after appropriate renormalisation. Thus, key recovery for Hawk will be slightly easier than for Falcon in either model. On the other hand, our verification bound σ_{ver} is a factor about 1.15 (Hawk-512) to 1.06 (Hawk-1024) shorter than Falcon after renormalisation, and thus obtaining strong forgeries is slightly harder than for Falcon in either model. In both Hawk-512 and Falcon-512 key recovery is harder than signature forgery, and thus hardening the latter, as we do, gives a slightly more secure scheme overall. For Hawk-1024 and Falcon-1024 key recovery is easier than signature forgery, and in the Falcon model we obtain a slightly less secure scheme overall. However, we note that part of the key recovery methodology of Falcon is overconservative [DPPvW22b, App. D].

5.4 Parameters and performance

In Table 5.3 we list parameters and the output of the auxiliary script `code/find_params.sage` using our concrete cryptanalysis for Hawk.

Section 5.4.1 explains how these parameters were chosen. We explain the encoding used for public keys and signatures, and the simple encoding used for private keys, in Section 5.4.2. Finally, Section 5.4.3 contains the details behind Table 5.1. Note that the formal reductions in Section 5.6 require different parametrisations.

5.4.1 Parameter selection

In Hawk we set $\mathsf{saltlen} = \lambda + \log_2 q_s$, where $q_s = 2^{64}$ is the limit on the signature transcript size. The probability of a hash collision is then less than $q_s \cdot 2^{-\lambda}$ [PFH+22, Section 2.2.2]. Allowing $\mathsf{saltlen}$ to depend on λ implies one must know λ before computing $\mathsf{H}(\mathsf{m}\|\mathsf{r})$, which is commonly the case. For simplicity Falcon choose a fixed salt length of 320 bits. This is not optimal for $\lambda = 128$. Here Hawk-512 saves 16 bytes on signatures, though this saving is also available to Falcon-512.

The value of σ_{pk} for Hawk-512 listed in Table 5.3 is such that the probability of $\|(f,g)\|^2 > \sigma_{\mathsf{sec}}^2 \cdot 2n$ is greater than 99.5% for f, g both with odd algebraic norm and sampled as in KeyGen. For Hawk-1024, the probability that $\|(f,g)\|^2 > \sigma_{\mathsf{sec}}^2 \cdot 2n$ holds for similar f, g is greater than 80%.

There are two failures that may occur during signing in Hawk. Firstly, $\|\mathbf{x}\|^2$ may be too large. Secondly, Equation (5.3) may be violated, i.e., decompressing $\mathsf{sig} = (\mathsf{r}, s_1)$ may return $s_0' \neq s_0$, the original first component of the signature. We choose parameters σ_{ver} and ν_{dec} to make such failures unlikely.

For Hawk-512, $\mathbf{x}$ is too large with a probability of around 2^{-22}, determined by convolving the necessary distributions together, see the function `failProbability` in the auxiliary script `code/find_params.sage`. To obtain a strict upper bound on this probability one can use a looser tail bound analysis via Lemma 2.79 and Lemma 2.84, with $\varepsilon = 1/\sqrt{q_s \lambda}$ and $\tau = \sigma_{\mathsf{ver}}/\sigma_{\mathsf{sign}}$, which gives 2^{-17}. Similarly for Hawk-1024, $\mathbf{x}$ is too large with a probability of around 2^{-128} and the tail bound gives a probability of at most 2^{-121}.

Given a fixed private key $\mathsf{sk} = \mathbf{B}$, we provide a heuristic upper bound on the probability that decompression using Equation (5.2) gives $s_0' \neq s_0$, which is upper bounded by the probability that Equation (5.3) does not hold. This also upper bounds the probability a compressed signature is correct although $s_0 \neq s_0'$. Heuristically, we assume that x_0, x_1 are independently sampled from a normal distribution on $\mathbb{R}^n$ with mean $\mathbf{0}$

Table 5.3: Parameter sets for Hawk and their estimated security. The dimension, bit security and transcript size are used to determine σ_{sign}. Other standard deviations are determined in Section 5.3. We then estimate the probability that a signature fails for being too long. The estimated required block sizes β for BKZ reduction to achieve key recovery and signature forgery are then given. Finally, we give the estimated probability of finding a weak forgery via the attack in Section 5.3.3.

	Hawk-256	Hawk-512	Hawk-1024
Targeted security	Challenge	NIST-1	NIST-5
Dimension $d = 2n$	512	1024	2048
Bit security λ	64	128	256
Transcript size limit q_s	2^{32}	2^{64}	2^{64}
Signature std. dev. σ_{sign}	1.010	1.278	1.299
Verif. std. dev. σ_{ver}	1.040	1.425	1.572
Key Recovery std. dev. σ_{sec}	1.042	1.425	1.974
Key Gen. std. dev. σ_{pk}	1.1	1.5	2
Salt Length (bits)	112	192	320
$\log_2(\Pr[\text{sign fail}])$	-2	-22	-128
Key Recovery (BKZ) β_{key}	211	452	940
Strong Forgery (BKZ) β_{forge}	211	452	1009
$\log_2(\Pr[\text{weak forgery}])$	-83	-143	-683

and standard deviation σ_{sign}. Following Section 5.2.2 the decompression succeeds if Equation (5.3) holds, or equivalently,

$$\frac{f^\star}{q_{00}} \cdot x_0 + \frac{g^\star}{q_{00}} \cdot x_1 \in \left(-\frac{1}{2}, \frac{1}{2}\right)^n.$$

Since $\mathbf{B}$ is fixed, each coefficient in the left hand side above is normally distributed with mean 0 and variance $\|(f^\star/q_{00}, g^\star/q_{00})\|^2 \cdot \sigma_{\mathsf{sign}}^2 = \langle 1, q_{00}^{-1}\rangle \cdot \sigma_{\mathsf{sign}}^2$, using Equation (5.4). Hence, the probability that a particular coefficient is outside $(-\frac{1}{2}, \frac{1}{2})$ is given by

$$\operatorname{erfc}\left(\frac{1/2}{\sigma_{\mathsf{sign}}\sqrt{2 \cdot \langle 1, 1/q_{00}\rangle}}\right).$$

Using a union bound, the probability that decompression fails is heuristically bounded from above by $n \cdot \operatorname{erfc}\Big(1/(\sigma_{\mathsf{sign}}\sqrt{8 \cdot \langle 1, 1/q_{00}\rangle})\Big)$. By rejecting keys having $\langle 1, q_{00}^{-1}\rangle \geq \nu_{\mathsf{dec}}$, signature decompression fails for any $\mathbf{B}$ heuristically with probability at most,

$$n \cdot \operatorname{erfc}\Big(1/\left(\sigma_{\mathsf{sign}}\sqrt{8\nu_{\mathsf{dec}}}\right)\Big).$$

If we take $\nu_{\mathsf{dec}} = 1/1000$ in Hawk-512, this probability is upper bounded by 2^{-105}. This condition on (f, g) in KeyGen fails in about 9% of cases. We empirically determined this by sampling f and g with odd algebraic norm 100 000 times. Combining this with the small probability that $\|(f, g)\|$ is too small, one can efficiently sample (f, g) until all requirements, before TowerSolveI is invoked, are met.

In Hawk-1024 we take $\nu_{\mathsf{dec}} = 1/3000$ and decompression fails on a signature with probability less than 2^{-315} for a key satisfying $\langle 1, q_{00}^{-1}\rangle < \nu_{\mathsf{dec}}$. This condition fails in about 0.9% of the cases during sampling of (f, g) inside KeyGen.

5.4.2 Encoding

In Hawk both the private key and $\mathbf{x}$ in Sign are sampled from a discrete Gaussian. As a consequence, the coefficients of the public key and the signature roughly follow a normal distribution. Therefore, it is beneficial to use the Golomb–Rice encoding [Gol66]. This encoding is used for the signatures in Falcon [PFH+22].

For encoding, we use an altered absolute value,

$$|\cdot|' \colon \mathbb{Z} \to \mathbb{N}, \qquad x \mapsto \begin{cases} x & \text{for } x \geq 0 \\ -x-1 & \text{for } x < 0. \end{cases}.$$

The map that sends x to its sign and $|x|'$ gives a bijection $\mathbb{Z} \to \{0,1\} \times \mathbb{N}$. Given a quantity that is sampled from a discrete Gaussian distribution with (an above smoothing) parameter σ, take an integer k close to $\log_2(\sigma)$. To encode a value $x \in \mathbb{Z}$, first output the sign of x and the lowest k bits of $|x|'$ in binary. Then output $\lfloor |x|'/2^k \rfloor$ in unary, i.e., $\lfloor |x|'/2^k \rfloor$ zeros followed by a one. Note that Falcon uses $|\cdot|$ where we use $|\cdot|'$, but their decoding fails when negative zero is encountered to ensure unique encodings [PFH+22, Section 3.11.2]. An advantage of this altered absolute value is that it sometimes saves one bit and is easy to implement: $|\mathtt{x}|'$ is the XOR of `x` and `-(x >> 15)` when `x` has 16 bits.

We use the Golomb–Rice encoding on `pk` and s_1. In particular, for `pk` the coefficients of q_{01} and coefficients 1 up to and including $n/2-1$ of q_{00} are encoded, with $k = 9$ and $k = 5$ respectively for Hawk-512 and $k = 10$ and $k = 6$ for Hawk-1024. The constant coefficient of q_{00} is output with 16 bits as its size is much larger than the other coefficients. The second half of q_{00} can be deduced from its self-adjointedness. For s_1 we use $k = 8$ and $k = 9$ in our implementation for Hawk-512 and Hawk-1024 respectively.

For the private key, we note the sampler in our implementation of Hawk-512 generates coefficients for f and g with an absolute value at most $13 < 2^4$, we encode these with 5 bits, one of which is the sign. Likewise, for the remaining polynomial F of the private key, we encode with one byte per coefficient (recall G can be reconstructed). We use this simple encoding and decoding for our private key as it is constant-time. Hawk-1024 requires 6 bits for coefficients of f and g since the sampler generates values of absolute value at most $18 < 2^5$.

By using the Asymmetric Numeral Systems (ANS) encoding [Dud14, DTGD15] instead of Golomb–Rice encoding, signature sizes of Falcon shrink by 7%–14% [ETWY22]. Making use of the open-source rANS implementation by Fabien Giesen at https://github.com/rygorous/ryg_rans in the encoding and decoding of Hawk public keys, we are able to reduce public key sizes by 15 and 30 bytes for Hawk-512 and Hawk-1024 respectively. Because this only saves less than 2% on the public key

sizes but adds significantly to the code complexity, we did not use this encoding in the final version.

5.4.3 Performance

We report on the performance of our implementation of Hawk-512 and compare it to that of Falcon-512 in Table 5.1. Hawk was compiled with `gcc` version 12.1.0 using compilation flag `-O2` (and `-mavx2` for AVX2). Optimisation flag `-O2` gave better performance than `-O3`. The code for Falcon was taken from the 'Extra' folder in the Round 3 submission package https://falcon-sign.info/falcon-round3.zip, and was compiled with the same `gcc` using compilation flags `-O3 -march=native`.

Memory usage

The reference implementation of Hawk-512 uses 24 kB, 15 kB and 18 kB of RAM for KeyGen, Sign and Verify respectively, versus 16 kB, 40 kB and 4 kB for Falcon-512 respectively. Hawk requires more RAM for KeyGen compared to Falcon to execute Line 6 of Algorithm 5.1. RAM usage of Falcon's KeyGen is more than reported on https://falcon-sign.info as we took the RAM usage of the API functions, which takes sizes of decoded keys into account. The AVX2 optimised implementation of Hawk requires 27 kB and 24 kB for Sign and Verify respectively, prioritising speed over memory usage. For Hawk-1024 and Falcon-1024 memory usage roughly doubles.

Batched versus dynamic signing

For consistency with the Falcon report [PFH+22], Table 5.1 reports signing speeds for batched usage, that is, after some precomputation expanding the private key (called `expand_seckey`). If one needs to start from the private key without expanding it, the performance of Falcon is significantly affected while the precomputation is much lighter for Hawk. The timings for dynamic signing are given in Table 5.4.

5.5 Implementation details

We have written a reference and an AVX2 optimised implementation for Hawk-512 and Hawk-1024 in the C programming language. We have

Table 5.4: Performance of dynamic signing for Falcon and Hawk on same PC with same compilation flags as in Table 5.1.

Scheme	AVX2 Sign	Reference Sign
Falcon-512 [PFH+22]	320 µs	5427 µs
Hawk-512 (this work)	58 µs	168 µs
Speed-up (Falcon/Hawk)	↓ 5.5×	↓ 32×
Falcon-1024 [PFH+22]	656 µs	11 868 µs
Hawk-1024 (this work)	114 µs	343 µs
Speed-up (Falcon/Hawk)	↓ 5.8×	↓ 35×

reused a significant portion from the public implementation of Falcon, because Hawk and Falcon are algorithmically rather similar. This section provides details on the design choices that were made in these implementations.

All the implementations are *isochronous*, i.e., the runtime of KeyGen and Sign gives no information about the used private data. Verification is trivially isochronous as it only uses public information. Our implementations are not *constant-time* because of two reasons: KeyGen and Sign sometimes restart depending on internal probabilistic events, and time varies to encode the public quantities pk and sig.

Almost all processors that support AVX2 have a floating-point unit, and plenty of RAM. Therefore, the AVX2 optimised implementation makes extensive use of built-in floating-point numbers and FFTs for multiplications or divisions in a number ring. On the other hand, the reference implementation uses the NTT for multiplication in the number ring and only uses an FFT with fixed point precision to decompress signatures during verification. As floating-points are required in TowerSolveI during key generation, the reference implementation emulates floating-points (IEEE-754 'binary64' format) in key generation, and is entirely free of them in signing and verification.

Sampling and pseudorandomness

The extendable output function SHAKE256 is used to seed a pseudorandom number generator (PRNG) based on ChaCha20 [Ber08] during

sampling in KeyGen and Sign. As the Gaussian parameters used in the scheme are fixed, DGS can be performed efficiently with precomputed probability tables and this PRNG.

Fast polynomial arithmetic

We want all computations to be performed in time $O(n \log n)$. Addition and subtraction of two polynomials in $R = \mathbb{Z}[X]/(X^n + 1)$ are linear operations, but naïve multiplication in R requires $O(n^2)$ integer multiplications. There are two ways to achieve $O(n \log n)$ via the specific structure of the used number ring. First, one can use the fast Fourier transform (FFT) to perform a multiplication in $O(n \log n)$. Alternatively, one can use the number theoretic transform (NTT), which works with a sufficiently large prime modulus $p \equiv 1 \pmod{2n}$ and an element $\omega \in \mathbb{F}_p^\times$ of order $2n$. As $\mathbb{F}_p^\times$ is a cyclic group of order $p - 1$, the NTT computes $f(\omega^i) \in R$ for all $i \in (\mathbb{Z}/2n\mathbb{Z})^\times$ in time $O(n \log n)$.

When polynomials are transformed using FFT or NTT, multiplication is coefficientwise. The resulting polynomial in R is then obtained via the inverse transformation. The NTT gives the correct result $a(X) \cdot b(X)$, if the used prime p is at least twice as large as the absolute value of any coefficient of $a(X)$, $b(X)$ or $a(X) \cdot b(X)$. We only use the NTT in the reference implementation.

Multiplications in R could also be implemented with Karatsuba or Toom–Cook style multiplication. For certain applications such as masking this may have comparable performance for the given parameters.

The FFT in our implementation uses double precision floating-point numbers (`double`). When a CPU supports AVX2, the FFT is faster than the NTT, but also requires more RAM. Like Falcon, we use the 'bit reversal' ordering of the roots in the FFT representation, and only half the embeddings are stored since the other half are their complex conjugates. Moreover, of these n fixed-point or floating-point numbers, the first $n/2$ are the real parts and the last $n/2$ are the imaginary parts, giving performance benefits [PFH+22, Sec. 4.2.1]. The advantage of this ordering is that self-adjoint polynomials, like q_{00} and q_{11}, only have real embeddings and thus require only half the memory. An implementer is free to choose their own ordering as long as it is consistent within the implementation.

Divisions and signature decompression

When decoding the private and public key, we require a polynomial division in $\mathbb{K}$ to recover $G = (1 + gF)/f$ and $q_{11} = (1 + q_{10}q_{01})/q_{00}$ respectively. Since these exact divisions have output in R, they can be computed using FFT or NTT, by performing a division coefficientwise in the transformed domain. In KeyGen, it should be checked that all NTT coefficients of f and q_{00} are nonzero.

5.5.1 Key generation

Key generation is the same for both implementations, as the majority of the work is the TowerSolveI procedure. This requires floating-points, so we emulate them in the reference implementation. At the time TowerSolveI was invented [PP19, Sec. 6], the problem of having a vector reduction in TowerSolveI using fixed point arithmetic was left open. If one is able to solve this problem, one can remove all floating-point emulation from the reference implementation of Hawk.

In key generation, f, g are sampled in R with coefficients from $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pk}}}$. This sampling is performed using a precomputed cumulative distribution table with 63 bits of precision and produces values of absolute value at most 13 for Hawk-512 and at most 18 for Hawk-1024. The kth index of the table (with $k = 0, 1, \ldots, 12$ for Hawk-512) contains the integer

$$\left\lceil 2^{63} \cdot \Pr_{x \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pk}}}} [|x| \geq k + 1] \right\rfloor,$$

so the statistical distance between $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pk}}}$ and the implemented sampling procedure is smaller than 2^{-64}.

We then check five conditions, before calling TowerSolveI.

1. $\mathrm{N}(f)$ and $\mathrm{N}(g)$ are both odd, i.e., both the sum of coefficients of f and the sum of coefficients of g are odd,
2. $\|(f, g)\|^2 > \sigma_{\mathsf{sec}}^2 \cdot 2n$,
3. $f(X)$ is invertible in $\mathbb{Z}[X]/(X^n + 1, 2^{16} + 1)$, i.e., the NTT of f modulo $2^{16} + 1$ has no entry equal to zero,
4. $ff^\star + gg^\star$ is invertible in $\mathbb{Z}[X]/(X^n + 1, 2147473409)$, i.e., the NTT of q_{00} modulo 2147473409 has no entry equal to zero,

5. $\langle 1, q_{00}^{-1} \rangle < \nu_{\mathsf{dec}}$.

The primes $2^{16} + 1$ and 2147473409 are both 1 (mod $2n$) and will be motivated in sign and verify respectively. The conditions in 3. and 4. are required only in the reference implementation. However, we check these also in the AVX2 implementation as a signer should be able to switch to the reference implementation without changing the key pair and a verifier should be able to use the reference implementation.

If any of the checks fail, we reject this sample and restart key generation. We remark that in Falcon if the sum of coefficients of e.g. $\mathrm{N}(f)$ is even, then the final coefficient is resampled until its parity changes. As this slightly alters the key distribution we choose caution and resample both f and g completely.

The time to find a valid pair (f, g) is much smaller than the time to run TowerSolveI. The implementation of the TowerSolveI algorithm is taken from Falcon setting $q = 1$. Since the standard deviation of the coefficients of f and g is smaller for Hawk, the upper bound on the number of bits required to represent intermediate polynomials in the tower of fields is a little smaller.

The TowerSolveI algorithm fails if $\mathrm{N}(f)$ and $\mathrm{N}(g)$ are not coprime over $\mathbb{Z}$, or the returned F, G have a coefficient with absolute value larger than 128, or F, G do not satisfy the sanity check,

$$fG - gF = 1 \pmod{2147473409},$$

which is rather unlikely. The sanity check catches rare cases where the intermediate coefficients of normed down f, g, F or G may be larger than is supported, see § Coefficient Sizes [PFH+22, Sec. 4.4.3]. If TowerSolveI fails, we restart key generation. All these checks are also in the reference implementation of Falcon.

Otherwise, the public key is computed, and the program checks if the encoded public key is not larger than the allowed maximum size. The encoded public key first contains a header byte where the four most significant bits are set to zero and the remaining four bits contain the value $\log_2(n)$ for the n used in KeyGen (512 or 1024). After the header byte follows the used encoding of q_{00} and q_{01}.

Experimentally the average public key size is 1006.5 bytes (including its header byte) with a standard deviation of 6.1. To be able to reserve memory in advance for encoding a public key, we enforce the encoding of a public key to be of size at most 1043 bytes in Hawk-512, which is

6 standard deviations above the average, altering the key distribution only a little. In Hawk-1024, an average public key is of size 2329.2 bytes with a standard deviation of 11.2 and therefore the encoding of a public key can be at most 2396 bytes long. These averages were obtained by generating 10^5 keys.

Moreover, given the small σ_{pk}, we know the constant coefficient of q_{00} fits in an `int16_t`. It is checked for all the other coefficients of the quadratic form if these lie in a certain range. For example, in Hawk-512 the absolute values must be (strictly) smaller than $2^9, 2^{12}$ and 2^{15} for q_{00}, q_{01} and q_{11} respectively. These bounds provide handles in verification when these elements are multiplied by a signature.

Babai reduction

In KeyGen, after TowerSolveI outputs relatively short $(F, G)^{\mathsf{T}}$, we perform another reduction step using ffNP [DP16, Alg. 4] to further shrink the size of $(F, G)^{\mathsf{T}}$. TowerSolveI returns an element $(F, G)^{\mathsf{T}}$ whose projection onto the module lattice $M = (f, g)^{\mathsf{T}} \cdot R$, i.e., the rank
5 n real lattice with basis $\mathbf{C} = \text{Rot}((f, g)^{\mathsf{T}})$, lies in the fundamental parallelepiped defined by $\mathbf{C}$. If this projection is uniformly distributed here, the expectation of its squared norm is $\frac{n}{12} \cdot 2n\sigma_{\mathsf{pk}}^2$.

However, Line 6 in Algorithm 5.1 implies the projection of $(F, G)^{\mathsf{T}}$ onto M lies in the fundamental domain generated by the Gram–Schmidt orthogonalisation of the rotations of $(f, g)^{\mathsf{T}}$ in bit reversed order. The ith vector $(i = 1, 2, \ldots, n)$ has an expected norm of $\sqrt{\frac{2n+1-i}{2n}} \cdot \|(f, g)\|$ [Pre15, Lem. 6.9]. Therefore, the expected squared norm of a point sampled uniformly from this fundamental domain will be

$$\frac{1}{12} \cdot \frac{(3n+1)n/2}{2n} \cdot 2n\sigma_{\mathsf{pk}}^2 = \frac{3n+1}{48} \cdot 2n\sigma_{\mathsf{pk}}^2 \leq \frac{n}{12} \cdot 2n\sigma_{\mathsf{pk}}^2.$$

We observe that the squared norm of (F, G) is reduced by a factor of 4/3 by Line 6 of Algorithm 5.1. This shrinks the average public key size in Hawk-512 from 1027 bytes to 1006. We note that using ffNP in Falcon would not lead to smaller public key sizes, and only leads to a slightly better trapdoor sampler.

Our implementation of ffNP is written in a memory friendly manner, overwriting certain polynomials when they are no longer used.

Remark 5.1. Since the initial work on Hawk [DPPvW22a], multiple changes have been made to KeyGen. Namely, the following things were modified when Hawk was originally submitted to NIST [BBD+23].

- The centred binomial distribution (CBD) is used to sample coefficients for f and g, which only slightly changes the size distribution of public keys and signatures.
- Line 6 in Algorithm 5.1 is omitted, which changes reduction of (F, G) to a structured variant of Rounding-Off [Bab86].
- A new implementation of TowerSolveI is used [Por23] that only uses fixed-points and no floating-points anymore.
- The private key is essentially encoded as

$$(\mathsf{kgseed}, F \bmod 2, G \bmod 2),$$

  where kgseed is the seed used to sample a completable (f, g) in KeyGen. During signing, Line 3 of Algorithm 5.2 computes the target modulo 2, so it is sufficient to know $(F, G) \pmod 2$. Moreover, s_1 can be computed on Line 7 without (F, G) by Equation (5.1). The private key consists of 184 and 360 bytes for Hawk-512 and Hawk-1024 respectively.
- We apply a light version by Pornin and Stern [PS05] of the BUFF transform to Hawk [CDF+21, ADM+24]. Namely, the hash function on Line 2 of Algorithm 5.2 gets as input the message, salt and a hash $\mathsf{H}(\mathbf{Q})$ of the public key. Now, KeyGen computes $\mathsf{H}(\mathbf{Q})$ once to store it in the private key, so Sign can use it.

Subsequently, when Hawk advanced to the second round [BBD+24], there was one more modification. Vector reduction inside TowerSolveI was adjusted to reduce F with respect to f using Rounding-Off, and then update G accordingly. Namely, a multiple of (f, g) is subtracted from (F, G) such that $\lceil F/f \rfloor$ holds. This reduction often gives the same reduced vector (F, G) as calling Rounding-Off algorithm with (F, G) and (f, g) [Por25]. This allows for an easier implementation of KeyGen that uses only 12 kB of RAM for Hawk-512.

5.5.2 Signature generation

A message m is hashed to some element $h \in \{0,1\}^{2n}$ using SHAKE256 on the bit string $\mathsf{m}\|r$, with $r \in \{0,1\}^{320}$ a random salt. Whenever a check fails in signing, a new random salt is generated. Note the message is not hashed again as it is enough to recover the internal state of SHAKE256 after consuming the message. This approach is only possible because the salt comes last.

In the case of Falcon, we do not see how reusing the salt is compatible with the security argument of Gentry et al. [GPV08]. Reusing the salt may lead to a statistical leak for Falcon, though it may be hard to exploit as failures in signing are rare. Nevertheless, we choose to be cautious when it comes to statistical leaks.

First, there is a check if a signature is short enough, which is Line 5 in Algorithm 5.2. Moreover, it is checked that all coefficients of s_1 are not too large and the encoded signature cannot be too large as well, similar to the encoded public key.

In signing the two implementations differ: the AVX2 version uses
5 double precision floating-points inside the FFT while the reference implementation uses the NTT with prime $p = 2^{16} + 1$.

AVX2 optimised implementation

During AVX2 signing we transform the whole basis $\mathbf{B}$ into FFT representation. Since G is not stored in the private key, $G = (1 + gF)/f$ is calculated in FFT representation after f, g, F and 1 are put into FFT representation. Note that the Fourier transform of 1 is a vector with a one in every entry. Then, the hash $\mathbf{h} = (h_0, h_1)^\mathsf{T} \in R^2$ is also put into FFT representation after which we calculate $\mathbf{t} = \mathbf{B} \cdot \mathbf{h}$ in FFT representation and then invert the FFT. From this, the point $\mathbf{x} = (x_0, x_1)^\mathsf{T} \leftarrow \mathcal{D}_{2\mathbb{Z}^{2n}+\mathbf{t},2\sigma_{\mathsf{sign}}}$ is sampled. This sampled point then overwrites the initial target and is converted to FFT representation. With the basis still in FFT representation we can easily compute the second component of $\mathbf{B}^{-1}\mathbf{x}$ by computing $-gx_0 + fx_1$ using Equation (5.1). Now the signature is computed using

$$s_1 = \frac{h_1 - (-gx_0 + fx_1)}{2}.$$

Note here that h_1 does not need to be converted to FFT representation.

If one adds a compilation flag `-DHAWK_RECOVER_CHECK` explicitly during compilation, it is checked whether the vector $\mathbf{x}$ satisfies Equation (5.3)

during signing. In batched signing, the basis $\mathbf{B}$ is precomputed in FFT representation and therefore cannot be altered. If the code is compiled with the flag `-DHAWK_RECOVER_CHECK`, the expanded private key also contains $1/(f^\star f + g^\star g)$ in FFT representation to make the check faster.

Note that the target $\mathbf{B} \cdot \mathbf{h}$ only needs to be known modulo 2 to sample $\mathbf{x}$. However, since f and g need to be in FFT representation to compute s_1 with $\mathbf{B}^{-1}$, as far as we know the best approach is to compute $\mathbf{Bh}$ exactly and then reduce it modulo 2. Having a fast multiplication in the ring $\mathbb{Z}[X]/(2, X^n + 1) = \mathbb{Z}[X]/(2, (X + 1)^n)$ may result in a faster signing algorithm and would reduce the memory usage of the current implementation, making this interesting future work.

Reference implementation

In the reference implementation we use the Fermat prime $p = 2^{16} + 1$ which is $1 \pmod{2048}$. The coefficients in $\mathbf{B} \cdot \mathbf{h}$ are distributed around zero with a standard deviation of less than 400 for Hawk-1024. Since the prime used in signing is much larger than this standard deviation, the inverse NTT provides the correct lifting from $\mathbb{Z}/p\mathbb{Z}$ to $\mathbb{Z}$. Using the special property of the Fermat prime $2^{16} + 1$, multiplication is done using $2^{16} \equiv -1 \pmod{2^{16} + 1}$. Inverting an element $x \in \mathbb{Z}/p\mathbb{Z}$ is done by calculating $x^{p-2} \pmod{p}$, and can be done for this prime with an addition chain requiring 19 multiplications, which is faster than the naïve exponentiation by squaring. This inverse is only used to reconstruct G using $G = (1 + gF)/f$, and since the third condition in KeyGen holds, f is invertible modulo p.

One can reduce the memory usage of signing from 15 kB down to 8 kB for Hawk-512 by using a prime $p < 2^{16}$ for example $p = 12289$ or $p = 18433$ as then values can be stored in 16 bits rather than 32 bits. Here to have constant-time multiplications, one can use Montgomery modular reduction [Mon85] with radix 2^{16}, which also has better performance than the built-in modulo operator. Interestingly, there are addition chains for $p = 12289$ and $p = 18433$ both requiring 18 multiplications, which is much less than the 26 needed for exponentiation by squaring. Since a division is as costly for $p = 12289$ as for $p = 18433$, it is preferable to use the latter prime as larger integers are supported without incorrect lifting.

The NTT version of signing is essentially obtained by replacing all FFT invocations with NTT invocations. One of the few differences is

that the NTT version calculates s_1 using

$$s_1 = (-g, f) \cdot \frac{\mathbf{t} - \mathbf{x}}{2},$$

where the division by two is performed in $\mathbb{Z}$ before passing to the NTT representation. Here, s_1 is calculated correctly with the NTT when the absolute value of the coefficients of the actual s_1 are all smaller than $p/2$, as otherwise an incorrect lift from $\mathbb{Z}/p\mathbb{Z}$ to $\mathbb{Z}$ may be chosen when inverting the NTT. If the equivalent equation from the AVX2 implementation is used, these coefficients must be bounded by $p/4$ in absolute value.

In the reference implementation, one cannot supply the compilation flag `-DHAWK_RECOVER_CHECK` to perform the recovery check as in Equation (5.3), because this check can only be done with little overhead when basis and $\mathbf{x}$ are already in FFT representation. To still have this check without the use of emulated floating-point numbers, one could simply run the verification algorithm inside signing, although this check almost never fails, see Section 5.4.1.

5

Failure checks

By default, we catch invalid signatures before they are issued. The first failure check is Line 5 of Algorithm 5.2. A decompression may also output an incorrect $s_0' \neq s_0$. To catch this we could check with an FFT if Equation (5.3) holds, and restart if not. We remove this decompression check, because it is extremely unlikely that it restarts over the lifetime of a key (see Section 5.4.1).

Omitting the first check may be necessary when implementing Hawk as a circuit inside an FHE scheme, where `while` loops are impractical. This comes at the cost of a rare but nonnegligible probability (see Section 5.4.1) of an invalid signature, which might be mitigated by reparametrising the scheme.

Discrete Gaussian sampling

We implement DGS for $\mathbb{Z} + \frac{1}{2}c$ with $c \in \{0, 1\}$ that is constant-time over input c and that uses 80 bits of randomness, sufficient for Section 5.3.1. Almost half of the time of Sign is spent on sampling.

This sampler uses a precomputed reverse cumulative distribution table scaled by a factor of 2^{78} similar to the case of Falcon [PFH+22,

Section 3.9.3]. To have a constant-time sampler, independent of the centre c and the outcome, the program reads the two cumulative distribution tables fully. Algorithm 5.4 explains the sampler in pseudocode. The algorithm uses a table where `RCDT[c][i]` contains the value

$$\left\lceil 2^{78} \cdot \Pr_{x \leftarrow \mathcal{D}_{2\mathbb{Z}+c,2\sigma_{\mathsf{sign}}}} [|x| \geq 2i+2] \right\rfloor.$$

Algorithm 5.4. SamplerZ(c): sampler for $2\mathbb{Z}+c$ with std. dev. $2\sigma_{\mathsf{sign}}$

Input: `RCDT` containing two reverse cumulative distribution tables for support $2\mathbb{Z}+c$ with $c \in \{0,1\}$
Output: a sample taken from a distribution close to $\mathcal{D}_{2\mathbb{Z}+c,2\sigma_{\mathsf{sign}}}$

$u \leftarrow \text{UniformBits}(78)$
$z_0 \leftarrow 0$
for `i` $= 0, 1, \ldots, 12$ **do**
    $z_0 \leftarrow z_0 + [\![u < (1-c) \cdot \texttt{RCDT[0][i]} + c \cdot \texttt{RCDT[1][i]}]\!]$
$z_0 \leftarrow 2 \cdot z_0 + c$
$z_0 \leftarrow z_0 - 2z_0 \cdot \text{UniformBits}(1)$
return z_0

As computers do not have built-in support for 78-bit numbers, the tabulated values are split in the implementation into a high part of type `uint16_t` and a low part of type `uint64_t`. The high part contains 15 bits and the low part 63 bits. We get faster constant-time code by not having the highest bit set in both parts, because checking $a < b$ can be done by looking at the sign bit of $a - b$ for numbers a, b not having their highest bit set, making comparisons easier and more efficient. Moreover, after the first 5 entries in the table, the high parts are all zero and are thus omitted from the table to save memory. More importantly this makes the comparison of 78-bit numbers easier for $i \in \{5, \ldots, 12\}$.

5.5.3 Signature verification

The implementation for signature verification follows Algorithm 5.3 closely, but computes the norm with respect to $\mathbf{Q}$ efficiently.

Moreover, a signature s_1 is decompressed, i.e., s_0 is recovered. This requires a division with rounding to the closest integral point in R,

which the FFT can do efficiently. One can even do this using fixed-point arithmetic: it is highly unlikely that the numerical error yields an incorrect rounding. Especially, when we require that the quantity in Equation (5.3) has to be in $(-0.49, 0.49)^n$, an absolute error of 0.01 is tolerated.

AVX2 optimised implementation

For the AVX2 version, one first computes q_{11} via the FFT and the equation $q_{11} = (1+q_{10}q_{01})/q_{00}$, making use of the fact that numerator and denominator are both self-adjoint and thus saving some multiplications.

In the recovery of s_0 using Equation (5.2), note that h_0 does not need to be in FFT representation. Moreover, $t_1 = h_1 - 2s_1$ is calculated in FFT representation, and after recovering s_0 one sets $t_0 = h_0 - 2s_0$ in FFT representation, after which we check $\|(t_0, t_1)^{\mathsf{T}}\|_{\mathbf{Q}}^2 \leq 8n \cdot \sigma_{\mathsf{ver}}^2$. Thus, in total there are four FFT calls for t_0, t_1, q_{00} and q_{01} and an inverse FFT call on $t_1 \cdot q_{01}/q_{00}$ needed to calculate s_0.

To compute the norm with respect to $\mathbf{Q}$, we use $\|\mathbf{t}\|_{\mathbf{Q}}^2 = \frac{1}{n}\operatorname{Tr}(\mathbf{t}^\star\mathbf{Q}\mathbf{t})$ as we have $\operatorname{Tr}(x) = \sum_\sigma \sigma(x)$ for $x \in R$, where σ ranges over all the field embeddings of $\mathbb{Q}(\zeta_{2n})$ into $\mathbb{C}$. Recall that the FFT representation of x holds all the $\sigma(x)$ up to complex conjugation, making the trace easily computable once x is in FFT representation. Moreover, the norm is a real number so we only need the real part of $\mathbf{t}^\star\mathbf{Q}\mathbf{t}$ under the embeddings. For example, we have

$$\left\|\begin{pmatrix} t_0 \\ 0 \end{pmatrix}\right\|_{\mathbf{Q}}^2 = \frac{1}{n}\sum_\sigma \sigma\left(t_0^\star q_{00} t_0\right) = \frac{1}{n}\sum_\sigma \sigma(q_{00})|\sigma(t_0)|^2,$$

where the contribution of some σ is equal to that of its conjugate embedding. A benefit of calculating the geometric norm using t_0, t_1, q_{00}, q_{01} and q_{11} in FFT representation is that it requires no extra memory. When checking if the norm does not exceed the norm bound, first we check if the outcome fits in an `int32_t` and is positive. It is then rounded to an integer before being compared to the norm bound.

Reference implementation

In the reference implementation, we still recover s_0 using FFTs and Equation (5.2), but using fixed-point arithmetic instead of floating-point arithmetic. As there are bounds on coefficients of s_1 and q_{00}, q_{01} from

signing and key generation respectively, it is possible to determine bounds on the absolute value of the numbers for all layers inside the FFT. The constant coefficient of q_{00} has a different bound to the other coefficients of q_{00}, so this one is excluded via setting it to zero before transforming q_{00} to the FFT representation. The FFT representation of the constant coefficient of q_{00} is a constant-valued vector and can thus be easily added afterwards.

Initially we scale up the coefficients of s_1, q_{00} and q_{01} by a power of two such that they require at most 29 bits (excluding the sign bit at the beginning). Then, after each layer of computations within the FFT, all numbers are divided by two, dropping the least significant bit after the fixed point. It is then readily proven that no overflow happens during the FFT computation.

Although the computation here uses less precision than the AVX2 version, the precision errors before rounding in Equation (5.2) are so small that rounding to an incorrect s_0 is highly unlikely. By the parameters in Section 5.4.1, it is extremely unlikely (probability less than 2^{-105}) that s_0 is recovered incorrectly with Equation (5.2). By small adaptations to this analysis, it can be shown that heuristically for a fixed key pair in Hawk-512 with probability at least $1 - 2^{-100}$, all the coefficients in Equation (5.2) lie in $\mathbb{Z} + (-0.49, 0.49)$ before being rounded to the nearest integer. This illustrates that the fixed-point computation can handle even 'large' precision errors of 0.01 (which are rare) as such almost never affect the recovery of s_0.

Experimentally, out of $5 \cdot 10^8$ valid generated signatures, where we generate 100 signatures for a key pair before sampling a new key pair, none failed with this fixed-point FFT recovery of s_0 both for Hawk-512 and Hawk-1024. Note that fixed-point arithmetic improves the performance of the reference implementation significantly: using floating-points, verification is done in 945 µs and 2095 µs for Hawk-512 and Hawk-1024 respectively.

For the norm calculation, q_{11} needs to be calculated from q_{00} and q_{01} using the NTT with the 31-bit prime. Here, the modular inverse of q_{00} is needed. We use the extended Euclidean algorithm for this, since verification is trivially isochronous, i.e., it operates only on public information and so there is no requirement for it to be constant-time. The extended Euclidean algorithm has better performance than exponentiation.

For calculating the norm modulo a prime p, we exploit a property

of the NTT, namely that $\mathrm{Tr}(x)$, the sum of FFT coefficients of x, is congruent to the sum of NTT coefficients of x modulo p for any element $x \in R$. The verification calculates the norm modulo two primes p_1, p_2, say n_1, n_2 with $n_i \in \{0, 1, \ldots, p_i - 1\}$ and then accepts the signature when $n_1 = n_2$ and n_1 is smaller than the norm bound. By theoretically determining the maximum norm possible with bounds on the signature coefficients and the public key coefficients, if this norm is known to be smaller than $p_1 \cdot p_2$ we know by the Chinese remainder theorem that no wrap around has happened, so all invalid signatures will be rejected.

First, denote the infinity norm of $f = f_0 + \cdots + f_{n-1}X^{n-1} \in R$ by

$$\|f_0 + f_1 X + \cdots + f_{n-1}X^{n-1}\|_\infty = \max_{0 \le i < n} \|f_i\|,$$

and write $\mathbf{t} = \mathbf{h} - 2\mathbf{s}$.

For Hawk-1024, the nonconstant coefficients of q_{00} and q_{11} are at most 2^{10} and 2^{17} in absolute value respectively, while $\|q_{01}\|_\infty \le 2^{14}$. The encoding of signatures requires $\|s_1\|_\infty < 2^{10}$. In addition, for bounding the norm, we require that after recovering s_0, we must have $\|s_0\|_\infty < 2^{14}$. Thus, $\|t_0\|_\infty < 2^{15}$ and $\|t_1\|_\infty < 2^{11}$.

We will now use that for polynomials $a(X), b(X) \in \mathbb{Z}[X]/(X^n+1)$, we have $\|a(X) \cdot b(X)\|_\infty \le \|a(X)\|_\infty \cdot \|b(X)\|_\infty \cdot n$. However, to use this bound, we handle the contribution of the constant terms of q_{00} and q_{11} separately as these are larger than the bounds mentioned above. First, KeyGen required that $\|F\|_\infty, \|G\|_\infty < 128$ and therefore, $\langle 1, q_{11} \rangle < 2n \cdot 128^2 = 2^{25}$. On the other hand, $\langle 1, q_{00} \rangle / \sigma_{\mathsf{pk}}^2$ behaves as a χ^2-distribution with $2n$ degrees of freedom, so by setting $D = 2n$ and $x = D/4$ in [LM00, (4.3)], the probability that this quantity is larger than $5n$ is at most $\exp(-n/2)$, which is miniscule for $n = 512$ and $n = 1024$. Therefore, we can safely say $\langle 1, q_{00} \rangle < 5n\sigma_{\mathsf{pk}}^2 < 2^{15}$. Now combining this with the bound on s_0 and s_1, the constant terms give a contribution of at most

$$2^{15} \cdot n \cdot \|t_0\|_\infty^2 + 2^{25} \cdot n \cdot \|t_1\|_\infty^2 \le n \cdot (2^{15+2\cdot 15} + 2^{25+2\cdot 11}) < 2^{58}.$$

These bounds on $\mathbf{Q}$ and $\mathbf{s}$ then imply that,

$$\begin{aligned} \|\mathbf{h} - 2\mathbf{s}\|_{\mathbf{Q}}^2 = \|\mathbf{t}\|_{\mathbf{Q}}^2 &\le n^2 \cdot \left(2^{2\cdot 15+10} + 2 \cdot 2^{15+11+14} + 2^{2\cdot 11+17}\right) + 2^{58} \\ &\le 2^{20}\left(2^{40} + 2^{41} + 2^{39}\right) + 2^{58} = 15 \cdot 2^{58}. \end{aligned}$$

Now given primes $p_1 = 2147473409$ and $p_2 = 2147389441$, one can see that $15 \cdot 2^{58} < p_1 p_2$. Of course, this also shows that Hawk-512 can

calculate the norm with these two primes, as the bounds on all numbers are smaller. In both cases, this shows that invalid signatures are always rejected as the geometric norm cannot be above $p_1 p_2$ for a signature that passes this NTT version of verification. On the other hand, the extra restriction on $\|s_0\|_\infty$ does reject some valid signatures, but this can be shown to happen with miniscule probability. Namely, the coefficients for recovered s_0, averaged over key pairs and valid signatures s_1, follow roughly a Gaussian distribution with $\sigma \approx 354$ and $\sigma \approx 962$ for Hawk-512 and Hawk-1024 respectively. Hence, a simple union bound yields $\Pr[\|s_0\|_\infty \geq 2^{13}] < 2^{-382}$ for Hawk-512 and $\Pr[\|s_0\|_\infty \geq 2^{14}] < 2^{-203}$ for Hawk-1024.

Moreover, we choose to work with 31-bit primes as the (Montgomery) modular multiplication requires 64-bit numbers and if one takes a prime of e.g. 62 bits, this requires working with 128-bit integers which is not supported in the C language.

5.6 Formal reductions

In this section, we consider two formal reductions related to Hawk. These reductions provide confidence in the design of Hawk, whereas the cryptanalysis in Section 5.3 provides confidence in security of Hawk when using the parameters from Section 5.4.1.

Section 5.6.1 discusses a worst case to average case reduction, and its relation to Hawk. Section 5.6.2 explains why Hawk does not fit in the PSF framework [GPV08]. Section 5.6.3 reduces the strong signature forgery security of Hawk's design to so-called omSVP.

5.6.1 Module LIP self reduction

Our goal in this section, is to sketch a worst case to average reduction for the search module Lattice Isomorphism Problem (smLIP), which underlies private key recovery in Hawk, see 5.3.2. However, this reduction will use a different average case distribution than the distribution of $\mathbf{Q}$ output by Algorithm 5.1, and requires for an efficient reduction that σ_{pk} grows exponentially in n, which is larger than what is chosen in Section 5.4.1.

Specifically, in this section, we will consider a key generation algorithm with the following changes compared to Algorithm 5.1:

- certain conditions in KeyGen that cause restarts, are omitted;
- instead of TowerSolveI on Line 5, we use HermiteSolve, which successfully completes a basis if and only if it is possible to do so;
- we pick $\sigma_{\mathsf{pk}} = 2^{\Theta(n)}$ to make the reduction efficient. This value is larger than in Section 5.4.1.

The goal of this section is to define the worst case and average case versions of smLIP, and to show that an adversary against the average case can be transformed into a similar adversary against the worst case. From this, one may conclude that smLIP is as hard for a worst-case instance as an average case instance. However, the average case distribution does not match the output distribution of KeyGen, because it requires σ to be much larger than the chosen σ_{pk} and its support is much larger.

Worst-case search module LIP

In this section, we generalise LIP to module lattices.

Recall, for a CM-field $\mathbb{K}$ and $\ell \in \mathbb{Z}_{\geq 1}$, the set of matrices representing Hermitian positive definite forms, as

$$\mathcal{H}_\ell^{>0}(\mathbb{K}) = \left\{ \mathbf{Q} \in \mathbb{K}^{\ell\times\ell} \;\middle|\; \mathbf{Q} = \mathbf{Q}^\star \text{ and } \forall \mathbf{v} \in \mathbb{K}^\ell \setminus \{\mathbf{0}\}\colon \operatorname{Tr}(\mathbf{v}^\star \mathbf{Q} \mathbf{v}) > 0 \right\}.$$

This set is the generalisation of quadratic forms to the module setting. Quadratic forms $\mathbf{Q}, \mathbf{Q}'$ are considered equivalent if there exists $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{Q}' = \mathbf{U}^\mathsf{T}\mathbf{Q}\mathbf{U}$. The natural translation to Hermitian forms becomes equivalence under the action of $\mathrm{GL}_\ell(\mathcal{O}_\mathbb{K})$. However, for simplicity we restrict ourselves to equivalence under the action of $\mathrm{SL}_\ell(\mathcal{O}_\mathbb{K})$, and we denote the equivalence class by

$$[\mathbf{Q}]_{\mathrm{sl}} = \left\{ \mathbf{Q}' \in \mathcal{H}_\ell^{>0}(\mathbb{K}) \;\middle|\; \exists \mathbf{U} \in \mathrm{SL}_\ell(\mathcal{O}_\mathbb{K})\colon \mathbf{Q}' = \mathbf{U}^\star \cdot \mathbf{Q} \cdot \mathbf{U} \right\}.$$

Throughout we implicitly restrict to $\mathbb{K}$ that are CM fields, e.g. all cyclotomic fields. Using the Gentry–Szydlo algorithm [GS02] and its generalisations [Kir16, LS17], solving LIP under both actions is equivalent for such fields. Namely, if we have $\mathbf{Q}' = \mathbf{U}^\star\mathbf{Q}\mathbf{U}$, the Gentry–Szydlo algorithm can find $u \in \mathcal{O}_\mathbb{K}$ such that $u^\star u = \det(\mathbf{Q}')/\det(\mathbf{Q})$. Then, we have $\mathbf{Q}' \in [\mathbf{V}^\star\mathbf{Q}\mathbf{V}]_{\mathrm{sl}}$ where $\mathbf{V} = \begin{pmatrix} u & 0 \\ 0 & \mathbf{I}_{\ell-1} \end{pmatrix}$, so equivalence under $\mathrm{GL}_\ell(\mathcal{O}_\mathbb{K})$ reduces to equivalence under $\mathrm{SL}_\ell(\mathcal{O}_\mathbb{K})$.

Definition 5.2 (Worst-case smLIP)**.** Given rank $\ell \in \mathbb{Z}_{\geq 2}$, a CM-field $\mathbb{K}$, and $\mathbf{Q} \in \mathcal{H}_\ell^{>0}(\mathbb{K})$, an instance of $\text{wc-smLIP}^{\mathbf{Q}}_{\mathbb{K},\ell}$, or the worst-case search module Lattice Isomorphism Problem, is any $\mathbf{Q}' \in [\mathbf{Q}]_{\text{sl}}$. A solution is any $\mathbf{U} \in \text{SL}_\ell(\mathcal{O}_\mathbb{K})$ for which we have

$$\mathbf{Q}' = \mathbf{U}^\star \mathbf{Q} \mathbf{U}.$$

Note a solution is unique up to automorphism of $\mathbf{Q}$.

Average case search module LIP

We now define an average case version of smLIP relevant to Hawk. It is less general than the worst-case version in that we implicitly fix $\ell = 2$ and consider only power of two cyclotomic fields $\mathbb{K} = \mathbb{Q}(\zeta_{2n})$ having conductor $2n$, where ζ_{2n} satisfies $(\zeta_{2n})^n + 1 = 0$.

We define our average case distribution for any class $[\mathbf{Q}]_{\text{sl}}$, but note that the average case distribution over $[\mathbf{I}_2(\mathbb{K})]_{\text{sl}}$ relates to our key generation in Algorithm 5.1. Finally, we define our average case distribution $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}})$ algorithmically, see Algorithm 5.7. This algorithm takes as input a particular form $\mathbf{Q}$ and a parameter σ that controls an internal discrete Gaussian sampling procedure, and outputs a sample from $[\mathbf{Q}]_{\text{sl}}$. One can think of $\sigma = \sigma_{\mathsf{pk}}$ in the case of Hawk.

Algorithm 5.5. HermiteSolve$(\mathbb{K}, f, g)$: basis completion (if possible)

Input: cyclotomic field $\mathbb{K}$, and $f, g \in \mathcal{O}_\mathbb{K}$
Output: completion $F, G \in \mathcal{O}_\mathbb{K}$ s.t. $fG - gF = 1$ if these exist, else $\perp$

1: $\mathbf{X} \leftarrow [\text{Rot}(f), \text{Rot}(g)]$
2: compute $\mathbf{U} \in \text{GL}_{2n}(\mathbb{Z})$ such that $\mathbf{X} \cdot \mathbf{U}$ is in HNF
3: **if** $\mathbf{X} \cdot \mathbf{U} \neq [\mathbf{I}_n(\mathbb{Z}), \mathbf{0}^{n \times n}]$ **then**
4: **return** $\perp$
5: parse first column of $\mathbf{U}$ as $\begin{pmatrix} \text{Vec}(G) \\ -\text{Vec}(F) \end{pmatrix}$ with $F, G \in \mathcal{O}_\mathbb{K}$
6: **return** $\begin{pmatrix} F \\ G \end{pmatrix}$

Algorithm 5.5, upon input a vector $\mathbf{b}_1 \in \mathcal{O}_\mathbb{K}^2$, computes and outputs a vector $\mathbf{b}_2 \in \mathcal{O}_\mathbb{K}^2$ such that $\det([\mathbf{b}_1, \mathbf{b}_2]) = 1$ holds whenever this is

possible. Otherwise, the algorithm should output $\perp$, indicating it failed. This Algorithm is a subroutine of Algorithm 5.1 to complete a basis with a second basis vector after its first basis vector is sampled.

We also define Reduce with respect to the form $\mathbf{Q}$ in Algorithm 5.6. This is a simpler, but less efficient, version of ffNP used in Algorithm 5.1. It serves two purposes; from a theoretical perspective it ensures that the distribution is well-defined, i.e., that the distribution of the output form is independent of the input form being used to sample, and from a practical perspective it ensures the second column of the completed basis is relatively short.

Algorithm 5.6. Reduce$(\mathbf{Q}, \mathbf{y}_1, \mathbf{y}_2)$: size reduction of $\mathbf{y}_2$ w.r.t. $\mathbf{Q}$ and $\mathbf{y}_1$

Input: cyclotomic field $\mathbb{K}$, and $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{O}_{\mathbb{K}}$, $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$
Output: canonical $\mathbf{y}_2'$ size reduced with respect to $\mathbf{Q}$ and $\mathbf{y}_1$

1: $\nu \leftarrow \left\lceil \frac{\mathbf{y}_1^\star \mathbf{Q} \mathbf{y}_2}{\mathbf{y}_1^\star \mathbf{Q} \mathbf{y}_1} \right\rfloor \quad \in \mathcal{O}_{\mathbb{K}}$
2: **return** $\mathbf{y}_2 - \nu \mathbf{y}_1$

The reduction in Reduce is canonical in the following way.

Lemma 5.3. *Let* $\mathbf{Q}' = \mathbf{U}^\star \mathbf{Q} \mathbf{U}$ *for* $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$ *and* $\mathbf{U} \in \mathrm{SL}_2(\mathcal{O}_{\mathbb{K}})$*, and suppose* $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_2' \in \mathcal{O}_{\mathbb{K}}^2$ *satisfy* $\det([\mathbf{y}_1, \mathbf{y}_2]) = \det([\mathbf{y}_1', \mathbf{y}_2']) = 1$*, where* $\mathbf{y}_1' = \mathbf{U}^{-1}\mathbf{y}_1$*. Then, we have*

$$\textsc{Reduce}(\mathbf{Q}', \mathbf{y}_1', \mathbf{y}_2') = \mathbf{U}^{-1} \cdot \textsc{Reduce}(\mathbf{Q}, \mathbf{y}_1, \mathbf{y}_2).$$

Proof. First, let us write $\begin{pmatrix} f & F \\ g & G \end{pmatrix} = \mathbf{U}^{-1} [\mathbf{y}_1, \mathbf{y}_2]$ and $\begin{pmatrix} F' \\ G' \end{pmatrix} = \mathbf{y}_2'$. Note that we have $fG - gF = fG' - gF' = 1$. One readily verifies $F' = F + \lambda \cdot f$ and $G' = G + \lambda \cdot g$ holds for $\lambda = GF' - FG' \in \mathcal{O}_{\mathbb{K}}$. Indeed,

$$F + \lambda f = F(\underbrace{1 - fG'}_{-gF'}) + GfF' = F'(\underbrace{-gF + fG}_{1}) = F',$$

and similarly $G' = G + \lambda g$.

We have $\mathbf{y}_2' = \mathbf{U}^{-1}\mathbf{y}_2 + \lambda \mathbf{y}_1'$, so $\mathbf{y}_2'$ and $\mathbf{U}^{-1}\mathbf{y}_2$ are size reduced to the same output, say $\mathbf{z}' = \textsc{Reduce}(\mathbf{Q}', \mathbf{y}_1', \mathbf{y}_2') = \textsc{Reduce}(\mathbf{Q}', \mathbf{y}_1', \mathbf{U}^{-1}\mathbf{y}_2)$. By writing $\mathbf{z} = \textsc{Reduce}(\mathbf{Q}, \mathbf{y}_1, \mathbf{y}_2)$, we see that $\mathbf{U}^{-1}\mathbf{z}$ is size reduced

with respect to $\mathbf{Q}'$ and $\mathbf{y}'_1$. Indeed,

$$\left\lceil \frac{\mathbf{y}_1'^{\star} \cdot \mathbf{Q}' \cdot (\mathbf{U}^{-1}\mathbf{z})}{\mathbf{y}_1'^{\star} \cdot \mathbf{Q}' \cdot \mathbf{y}'_1} \right\rfloor = \left\lceil \frac{(\mathbf{U}^{-1}\mathbf{y}_1)^{\star} \cdot \mathbf{Q}' \cdot (\mathbf{U}^{-1}\mathbf{z})}{(\mathbf{U}^{-1}\mathbf{y}_1)^{\star} \cdot \mathbf{Q}' \cdot (\mathbf{U}^{-1}\mathbf{y}_1)} \right\rfloor = \left\lceil \frac{\mathbf{y}_1^{\star} \cdot \mathbf{Q} \cdot \mathbf{z}}{\mathbf{y}_1^{\star} \cdot \mathbf{Q} \cdot \mathbf{y}_1} \right\rfloor = 0.$$

We conclude $\mathbf{z}' = \mathbf{U}^{-1}\mathbf{z}$. □

Algorithm 5.7. $\text{AC}_{\mathbb{K},\sigma}(\mathbf{Q})$: sampler for $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}})$

Input: cyclotomic field $\mathbb{K}$, parameter σ and $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$
Output: $(\mathbf{R}, \mathbf{Y}) \in [\mathbf{Q}]_{\text{sl}} \times \text{SL}_2(\mathcal{O}_{\mathbb{K}})$, where $\mathbf{R} = \mathbf{Y}^{\star}\mathbf{Q}\mathbf{Y}$

1: $\begin{pmatrix} f \\ g \end{pmatrix} = \mathbf{y}_1 \leftarrow D_{\mathbf{Q},\sigma}$ where $f, g \in \mathcal{O}_{\mathbb{K}}$
2: $\mathbf{y}_2 \leftarrow \text{HermiteSolve}(\mathbb{K}, f, g)$
3: **if** $\mathbf{y}_2 = \perp$ **then**
4: **restart**
5: $\mathbf{y}_2 \leftarrow \text{Reduce}(\mathbf{Q}, \mathbf{y}_1, \mathbf{y}_2)$
6: $\mathbf{Y} \leftarrow [\mathbf{y}_1, \mathbf{y}_2]$
7: $\mathbf{R} \leftarrow \mathbf{Y}^{\star} \cdot \mathbf{Q} \cdot \mathbf{Y}$
8: **return** $(\mathbf{R}, \mathbf{Y})$

The following lemma ensures that a sample $\mathbf{R} \in [\mathbf{Q}]_{\text{sl}}$ output by Algorithm 5.7 only depends on the class $[\mathbf{Q}]_{\text{sl}}$, and not on the input representative $\mathbf{Q}$. As a result the distribution $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}})$ is well-defined.

Lemma 5.4. *For any power of two cyclotomic* $\mathbb{K}$, $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$, $\mathbf{Q}' \in [\mathbf{Q}]_{\text{sl}}$ *and* $\sigma > 0$ *the distributions of* $(\mathbf{R}, \cdot) \leftarrow \text{AC}_{\mathbb{K},\sigma}(\mathbf{Q})$ *and* $(\mathbf{R}, \cdot) \leftarrow \text{AC}_{\mathbb{K},\sigma}(\mathbf{Q}')$ *are equal.*

Proof. For $\mathbf{Q}' \in [\mathbf{Q}]_{\text{sl}}$ there exists a $\mathbf{U} \in \text{SL}_2(\mathcal{O}_{\mathbb{K}})$ such that $\mathbf{Q}' = \mathbf{U}^{\star} \cdot \mathbf{Q} \cdot \mathbf{U}$. It is sufficient to show that for any $\mathbf{Y}$ created during $\text{AC}_{\mathbb{K},\sigma}(\mathbf{Q})$, $\mathbf{Y}' = \mathbf{U}^{-1} \cdot \mathbf{Y}$ is created within $\text{AC}_{\mathbb{K},\sigma}(\mathbf{Q}')$ with the same probability. Having shown this, since $\mathbf{Y}^{\star} \cdot \mathbf{Q} \cdot \mathbf{Y} = (\mathbf{Y}')^{\star} \cdot \mathbf{Q}' \cdot \mathbf{Y}'$, the two distributions for $\mathbf{R}$ are equal. Firstly, letting $\mathbf{y}'_1 = \mathbf{U}^{-1} \cdot \mathbf{y}_1$ we see that $\rho_{\mathbf{Q}',\sigma}(\mathbf{y}'_1) = \rho_{\mathbf{Q},\sigma}(\mathbf{y}_1)$. Given that the normalisation constant for a given σ will be equal over all forms in $[\mathbf{Q}]_{\text{sl}}$, the probability of sampling $\mathbf{y}'_1 \leftarrow D_{\mathbf{Q}',\sigma}$ is equal to the probability of sampling $\mathbf{y}_1 \leftarrow D_{\mathbf{Q},\sigma}$.

Now, since $\mathbf{U}^{-1}\mathbf{y}_2$ is a way to complete $\mathbf{y}'_1$, Algorithm 5.5 will succeed and complete to some $\mathbf{y}'_2$. After Algorithm 5.6 has reduced $\mathbf{y}'_2$, Lemma 5.4 shows that $\mathbf{y}'_2 = \mathbf{U}^{-1}\mathbf{y}_2$. Hence, $\mathbf{Y}' = \mathbf{U}^{-1} \cdot \mathbf{Y}$. □

We can now define the average case version of module LIP.

Definition 5.5 (Average case smLIP)**.** Given some power of two cyclotomic $\mathbb{K}$ and $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$ an instance of $\text{AC-SMLIP}^{\mathbf{Q}}_{\mathbb{K},\sigma}$, the average case search module Lattice Isomorphism Problem, is given by $\mathbf{Q}$ and an element $\mathbf{Q}' \in [\mathbf{Q}]_{\text{sl}}$ sampled from $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}})$. A solution is $\mathbf{U} \in \text{SL}_2(\mathcal{O}_{\mathbb{K}})$ such that $\mathbf{Q}' = \mathbf{U}^\star \cdot \mathbf{Q} \cdot \mathbf{U}$.

Reducing worst to average case

We expect that $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}})$ becomes harder as the parameter σ increases. In fact, following [DvW22], if σ is large enough we have equivalence with the corresponding worst-case problem, assuming HermiteSolve does not fail too often.

Lemma 5.6 (Worst case to average case)**.** *Given a machine that can solve* $\text{AC-SMLIP}^{\mathbf{Q}}_{\mathbb{K},\sigma}$ *in time* T *with probability* $\varepsilon > 0$*, for some* $\sigma \geq 2^{\Theta(n)} \cdot \lambda_{2n}([\text{Rot}(\mathbf{Q})])$*, one may solve* $\text{WC-SMLIP}^{\mathbf{Q}}_{\mathbb{K},\ell}$ *in expected time*

$$T + \mathbb{E}_{\mathsf{samples}} \cdot \mathsf{poly}(n, \log \sigma)$$

with probability ε*. Here* $\mathbb{E}_{\mathsf{samples}}(n, \sigma) \geq 1$ *is the expected number of times* $(f, g)^{\mathsf{T}}$ *is (re)sampled in Algorithm 5.7.*

Proof. As input, we receive $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$ and a worst-case instance $\mathbf{Q}' \in [\mathbf{Q}]_{\text{sl}}$. By first LLL reducing $\text{Rot}(\mathbf{Q}) \in \mathbb{Q}^{2n \times 2n}$, we can sample efficiently from $D_{\mathbf{Q},\sigma}$ via Lemma 2.81. Thus, we can sample $(\mathbf{Q}'', \mathbf{U}'') \leftarrow \text{AC}_{\mathbb{K},\sigma}(\mathbf{Q})$ in time $\mathbb{E}_{\mathsf{samples}} \cdot \mathsf{poly}(n, \sigma)$ by Algorithm 5.7. The sample $\mathbf{Q}''$ is distributed as $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}]_{\text{sl}}) = \mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{Q}']_{\text{sl}})$, and we have $\mathbf{Q}'' = \mathbf{U}''^\star \mathbf{Q} \mathbf{U}''$. Now $\mathbf{Q}''$ is an average case instance of $\text{AC-SMLIP}^{\mathbf{Q}'}_{\mathbb{K},\sigma}$, which the machine can solve in time T with probability ε. On success, we obtain some $\mathbf{U}' \in \text{SL}_2(\mathcal{O}_{\mathbb{K}})$ such that $\mathbf{Q}'' = \mathbf{U}'^\star \mathbf{Q}' \mathbf{U}'$, and $\mathbf{U} = \mathbf{U}''(\mathbf{U}')^{-1}$ gives a solution to the worst-case instance, i.e., $\mathbf{Q}' = \mathbf{U}^\star \mathbf{Q} \mathbf{U}$. □

Following the same argument as [DvW22, Lem 3.10] we may reduce σ in the reduction at the expense of an additive loss in runtime from the cost of stronger lattice reduction. There is a heuristic explanation and matching experimental evidence for why HermiteSolve fails with probability around 25% [DPPvW22b, App. A], which gives $\mathbb{E}_{\mathsf{samples}} \approx 4/3$, and thus the reduction above is in fact efficient.

Relation to Hawk key generation

There are multiple algorithmic differences between the average case distribution output by $\mathrm{AC}_{\mathbb{K},\sigma}(\mathbf{I}_2)$ and the public key output by KeyGen.

First, KeyGen uses a structured variant of Babai's Nearest-Plane algorithm to size reduce (F, G), whereas $\mathrm{AC}_{\mathbb{K},\sigma}(\mathbf{I}_2)$ uses a variant of Rounding-Off. This poses no notable issues, as the reduction on a Hermitian form $\mathbf{R}$ output by KeyGen can be reduced using Rounding-Off, and vice versa. Explicitly, given a Hermitian form $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$, the quadratic form $\mathbf{Q}' = \mathbf{U}^\star\mathbf{Q}\mathbf{U}$ is size reduced (using Rounding-Off), where

$$\mathbf{U} = \begin{pmatrix} 1 & -\lceil \mathbf{Q}_{01}/\mathbf{Q}_{00} \rfloor \\ 0 & 1 \end{pmatrix} .$$

Second, KeyGen in Algorithm 5.1 has more restart conditions because it uses TowerSolveI instead of HermiteSolve. Let p_r denote the probability that Algorithm 5.1 restarts when it internally samples a completable $(f, g)^\mathsf{T}$. If $\mathcal{A}$ is an adversary against Hawk's KeyGen using $\sigma = \sigma_{\mathsf{pk}}$, i.e., a machine that can return $\mathbf{B} \in \mathrm{SL}_2(\mathcal{O}_\mathbb{K})$ such that $\mathbf{B}^\star\mathbf{B} = \mathbf{Q}$ for a Hawk public key $\mathbf{Q}$ in time t and with probability ε, then $\mathcal{A}$ is an adversary for $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{I}_2(\mathcal{O}_\mathbb{K})]_{\mathrm{sl}})$ that runs in time t with probability at least $(1 - p_r) \cdot \varepsilon$. There is experimental and theoretical evidence that the component of p_r, caused by the use of TowerSolveI and the requirement on $\mathrm{N}(f), \mathrm{N}(g)$ to be odd in Algorithm 5.1, is a constant [DPPvW22b, App. A]. Although this reduction from $\mathsf{AC}_{\mathbb{K},\sigma}([\mathbf{I}_2]_{\mathrm{sl}})$ to Hawk's KeyGen is achievable for $\sigma = \sigma_{\mathsf{pk}}$, note that the composed reduction from worst case to Hawk's KeyGen requires σ_{pk} to be larger than chosen in Section 5.4.1. Therefore, there is no efficient reduction possible from worst-case smLIP to the average case key recovery of Hawk KeyGen.

Last, as Remark 5.1 indicates, Hawk uses a centred binomial distribution (CBD) to sample f, g. In this case, the average case output by $\mathrm{AC}_{\mathbb{K},\sigma}(\mathbf{I}_2)$ differs significantly from the public keys output by KeyGen. For example, while the discrete Gaussian sampler assigns the same probability mass to (f, g) of the same ℓ_2 norm, CDB does not. We therefore cannot claim any worst case to average case reduction for the problem of recovering the secret key of Hawk from the public key.

5.6.2 Inapplicability of PSF framework

In this section, we introduce the framework of Preimage Sampleable Functions (PSF) [GPV08, Sec. 5.2]. Falcon uses this framework to show it is SUF-CMA secure, assuming it is hard to find an inhomogeneous short integer solution (ISIS) in an NTRU lattice [PFH+22, Sec. 2.2]. We then explain why Hawk does not fit in the framework.

Definition 5.7. The *PSF framework*, or *trapdoor collision-resistant hash functions with preimage sampling*, is a tuple of probabilistic polynomial-time algorithms (TrapGen, SampleDom, SamplePre) satisfying the following properties.

Trapdoor generation: The output $(a, t) \leftarrow \text{TrapGen}(1^n)$, consists of a (public) description a that allows one to efficiently compute a function $f_a \colon D_n \to R_n$, where D_n and R_n are some efficiently recognisable domain and range, respectively, and t is (private) trapdoor information for f_a.

Domain sampling: Let $(a, t) \leftarrow \text{TrapGen}(1^n)$. Then, output from $\text{SampleDom}(1^n)$ follows a distribution on D_n, not necessarily uniform, such that $f_a(x)$ for $x \leftarrow \text{SampleDom}(1^n)$ is the uniform distribution on R_n.

Trapdoor preimage sampling: Let $(a, t) \leftarrow \text{TrapGen}(1^n)$. For all $y \in R_n$, output from $\text{SamplePre}(t, y)$ follows a distribution on $f_a^{-1}(y)$ that is statistically indistinguishable from $x \leftarrow \text{SampleDom}(1^n)$ such that $f_a(x) = y$.

Moreover, for a given $y \in R_n$ the min-entropy of $x \leftarrow \text{SampleDom}(1^n)$ such that $f_a(x) = y$ holds, should be sufficient, and it should be hard to find a collision for f_a, i.e., distinct $x_1, x_2 \in D_n$ satisfying $f_a(x_1) = f_a(x_2)$.

We now consider the example of a hash-and-sign signature scheme that is based on the hardness of ISIS for q-ary lattices.

Hash-and-sign for q-ary lattices

We obtain a hash-and-sign signature scheme [GPV08, Sec. 6] based on the hardness of ISIS by using Ajtai's algorithm [Ajt99], for trapdoor generation: $(a, \mathbf{T}) \leftarrow \text{TrapGen}(1^n)$, where $a = (\mathbf{B}, \mathbf{A}) \in \mathbb{Z}^{m \times m} \times \mathbb{Z}^{n \times m}$. Here, $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is a good trapdoor basis for the q-ary lattice $\Lambda =$

$\mathcal{L}_q^\perp(\mathbf{A})$ using Equation (2.5), and $\mathbf{B}$ is a bad basis for Λ. The domain is $D_n = \mathbb{Z}^m \cap (\sigma\sqrt{2\pi m} \cdot \mathcal{B}^m)$, and the range is $R_n = (\mathbb{Z}/q\mathbb{Z})^n$. We have

$$f_a \colon D_n \to R_n, \qquad \mathbf{x} \mapsto \mathbf{x}\mathbf{A} \pmod q \ ,$$

and $\textsc{SampleDom}(1^n) = \mathcal{D}_{\mathbb{Z}^m,\sigma}$. The algorithm $\textsc{SamplePre}(\mathbf{T}, \mathbf{y})$ first constructs arbitrary $\mathbf{x}_0 \in R_n$ such that $\mathbf{x_0}\mathbf{A} = \mathbf{y}$, and then outputs a sample from $\mathcal{D}_{\mathbf{T}\mathbb{Z}^m+\mathbf{x}_0,\sigma}$ using $\mathbf{T}$. The hash-and-sign signature scheme then uses a hash function $\mathsf{H} \colon \{0,1\}^* \to R_n$, and a valid signature on a message m is an element $\mathbf{x} \in D_n$ such that $f_a(\mathbf{x}) = \mathsf{H}(\mathsf{m})$.

Intuitively, it is easy for the public to generate $\mathbf{y}$ being the sum of a lattice point $\mathbf{x} \in \Lambda$ and a small error $\mathbf{e} \in D_n$, whereas not knowing the good basis $\mathbf{T}$ makes it hard to decode a particular $\mathbf{y} = \mathbf{x} + \mathbf{e}$ to its nearest lattice point $\mathbf{x}$.

The private key leakage from a signature transcript [NR09, DN12b] does not apply to this hash-and-sign signature scheme, because the list $(x_i)_{i=1}^{q_s}$ acquired from a signature transcript $(\mathsf{m}_i, x_i)_{i=1}^{q_s}$ is statistically indistinguishable from i.i.d. samples $x_i \leftarrow \textsc{SampleDom}(1^n)$ by definition of trapdoor preimage sampling.

Note, however, "for the security reduction to work, the signer must give out *at most one* preimage of a given point" [GPV08, Sec. 6]. Namely, if this is not the case, an adversary may obtain two *distinct* signatures $\mathbf{x}_1, \mathbf{x}_2$ on message m, i.e.,

$$f_a(\mathbf{x}_1) = f_a(\mathbf{x}_2) = \mathsf{H}(\mathsf{m}) \ ,$$

and subsequently, they will learn a short lattice vector $\mathbf{x}_1 - \mathbf{x}_2$ of norm at most $\sigma\sqrt{8\pi m}$, which solves SIS, the homogeneous variant of ISIS. The mentioned condition can be achieved either by adding sufficient so-called salt to messages before hashing, or by letting the signing algorithm return a cached signature when asked to sign a message for the second time.

Hash-and-sign for Hawk

The signature distribution is fundamentally different for Hawk. If one has the public key $\mathbf{Q}$, one can publicly sample from $\mathcal{D}_{\mathbf{Q},\mathbb{Z}^{2n},\sigma}$ when

$$\sigma \geq \sigma_{\mathsf{pk}}\sqrt{2n} \cdot (1/\pi) \cdot \sqrt{\log(2n+4)/2},$$

using Lemma 2.81. An average sample $\mathbf{x} \leftarrow \mathcal{D}_{\mathbf{Q},\mathbb{Z}^{2n},\sigma}$ has a squared $\mathbf{Q}$-norm of $2n\sigma^2 \geq (2n)^2\sigma_{\mathsf{pk}}^2 \log(2n+4)/(2\pi^2)$. On the other hand,

valid signatures provide lattice points of much smaller $\mathbf{Q}$-norm. Indeed, if anyone with a public key $\mathbf{Q}$ obtains a valid signature $\mathbf{s}$ on hashed message $\mathbf{h} = \mathsf{H}(\mathsf{m}) \in \{0,1\}^{2n}$, then they learn the lattice point $\mathbf{x} = \mathbf{h} - 2\mathbf{s} \in \mathbb{Z}^{2n}$ of average squared $\mathbf{Q}$-norm $8n\sigma_{\mathsf{sign}}^2$, which is a factor at least n smaller than the public samples for the parametrisations in Section 5.4.1.

In particular, the natural analogue of the PSF framework for q-ary lattices to Hawk is setting as domain $D_n = \{\mathbf{x} \in \mathbb{Z}^{2n} : \|\mathbf{x}\|_{\mathbf{Q}} \leq \sigma_{\mathsf{sign}}\sqrt{8n}$, and range $R_n = \{0,1\}^{2n}$, and $f_a \colon \mathbf{x} \mapsto \mathbf{x} \bmod 2$. However, there is no way to instantiate SampleDom in this case, as one cannot sample short enough vectors. Moreover, it appears that sampling from D_n is hard. This is why we introduce a new hardness assumption to base security on.

5.6.3 One more shortest vector problem

The one more SVP problem, henceforth omSVP, is the problem upon which we base the forgery security of our signatures. Informally we define an average case omSVP instance that samples a $\mathbf{Q}$ from a distribution over some $\mathcal{H}_2^{>0}(\mathbb{K})$ and gives Gaussian samples according to this $\mathbf{Q}$. If one can find some nonzero $\mathbf{x}$ that is sufficiently short with respect to $\mathbf{Q}$, and that is in some sense nontrivially new, then one solves the problem. We show that for certain parameters, if one can forge a signature against Hawk, then in the programmable random oracle model one can solve an instance of this problem. We then discuss parametrisations of our omSVP problem that we expect to be hard, with reference to Hawk parameters.

Security definitions

We first define the ROM-SUF-CMA game for signature schemes. Here Sign and Verify are given access to a random oracle $\mathsf{RO} \colon \{0,1\}^* \to \{0,1\}^{\ell(n)}$, the output length of which depends on the security parameter n. The security notion we consider here is strong unforgeability under chosen message attacks in the random oracle model, or ROM-SUF-CMA, see Figure 5.6.

Definition 5.8 (ROM-SUF-CMA security)**.** We say that a signature scheme $\Pi = (\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Verify})$ is $(t, \varepsilon, q_s, q_h)$-*ROM-SUF-CMA secure*, or strongly unforgeable under chosen message attacks in the

$\text{ROM-SUF-CMA}_{\Pi,\mathcal{A}}(1^n)$

1: $\mathcal{L}_{\text{Sign}} \leftarrow \emptyset$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \text{KeyGen}(1^n)$
3: $(\mathsf{m}^\star, \mathsf{sig}^\star) \leftarrow \mathcal{A}^{\text{OSign}(\cdot),\text{RO}(\cdot)}(\mathsf{pk})$
4: **return** $\left[\!\left[\text{Verify}^{\text{RO}(\cdot)}(\mathsf{pk}, \mathsf{m}^\star, \mathsf{sig}^\star) = 1 \ \wedge \ (\mathsf{m}^\star, \mathsf{sig}^\star) \notin \mathcal{L}_{\text{Sign}}\right]\!\right]$

$\text{OSign}(\mathsf{m})$

1: $\mathsf{sig} \leftarrow \text{Sign}^{\text{RO}(\cdot)}(\mathsf{sk}, \mathsf{m})$
2: $\mathcal{L}_{\text{Sign}} \leftarrow \mathcal{L}_{\text{Sign}} \cup \{(\mathsf{m}, \mathsf{sig})\}$
3: **return** sig

$\text{RO}(\mathsf{x})$

1: **if** $T[\mathsf{x}] = \perp$ **then** // initially $T[\mathsf{x}] = \perp$ for all $\mathsf{x} \in \{0,1\}^*$
2: $\quad T[\mathsf{x}] \leftarrow \text{U}(\{0,1\}^{\ell(n)})$
3: **return** $T[\mathsf{x}]$

Figure 5.6: ROM-SUF-CMA game

random oracle model, if for any adversary $\mathcal{A}$ running in time at most t, making at most q_s queries to its signing oracle, and making at most q_h queries to its random oracle, we have

$$\Pr[\text{ROM-SUF-CMA}_{\Pi,\mathcal{A}} = 1] \leq \varepsilon.$$

Trivial automorphisms

In this section, we define a set of automorphisms of $\text{Vec}(R^2)$, which give rise to 'trivial' wins in our omSVP game. For example, negation preserves the inner product with respect to any Hermitian form: $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{Q}} = \langle -\mathbf{x}, -\mathbf{y} \rangle_{\mathbf{Q}}$.

Definition 5.9. Given a number field $\mathbb{K}$, let $\mu_{\mathbb{K}}$ be the set of roots of unity.

Lemma 5.10. *For $\alpha \in R$ we have $\alpha^\star \alpha = 1 \iff \alpha \in \mu_{\mathbb{K}}$.*

Proof. If $\alpha^\star \alpha = 1$ then for any field embedding σ_i we have $|\sigma_i(\alpha)| = 1$ and therefore $|\sigma_i(\alpha)^j| = 1$ for all $j \in \mathbb{Z}$. Assume α is not a root of unity,

so no such $\sigma_i(\alpha)^j = 1$, then $\{\sigma_i(\alpha)^j\}_{j\in\mathbb{Z}}$ is dense in the unit circle of $\mathbb{C}$, a contradiction. The converse follows from $\sigma_i(\alpha^\star\alpha) = |\sigma_i(\alpha)|^2$ and noting that $\alpha \in \mu_{\mathbb{K}}$ must have $|\sigma_i(\alpha)| = 1$ else no power of it will equal 1. □

Lemma 5.11. *For any $\ell \in \mathbb{Z}_{\geq 1}$ some $\alpha \in R$ has $\|\alpha\mathbf{x}\|_{\mathbf{Q}} = \|\mathbf{x}\|_{\mathbf{Q}}$ for all $\mathbf{x} \in \mathbb{K}^\ell$ and $\mathbf{Q} \in \mathcal{H}^{>0}_\ell(\mathbb{K})$ if and only if $\alpha \in \mu_{\mathbb{K}}$.*

Proof. First note if $\alpha \in \mu_{\mathbb{K}}$ then by Lemma 5.10,

$$\|\alpha\mathbf{x}\|_{\mathbf{Q}} = \mathrm{Tr}(\alpha^\star\alpha\mathbf{x}^\star\mathbf{Q}\mathbf{x})/n = \mathrm{Tr}(\mathbf{x}^\star\mathbf{Q}\mathbf{x})/n = \|\mathbf{x}\|_{\mathbf{Q}}.$$

Conversely, if $\|\alpha\mathbf{x}\|_{\mathbf{Q}} = \|\mathbf{x}\|_{\mathbf{Q}}$ for all $\mathbf{x}$ and $\mathbf{Q}$ then it must be true for $\mathbf{x} = (1, 0, \ldots, 0)^\mathsf{T}$ and $\mathbf{Q} = \mathbf{I}_\ell(\mathbb{K})$, which by unrolling the definitions tells us $\sum_{i=1}^n \sigma_i(\alpha^\star\alpha) = n$. Similarly, it must be true for $\mathbf{x} = (\alpha, 0, \ldots, 0)^\mathsf{T}$ and $\mathbf{Q} = \mathbf{I}_\ell(\mathbb{K})$, which by unrolling the definitions tells us $\sum_{i=1}^n \sigma_i(\alpha^\star\alpha)^2 = n$. Note that for a set of real numbers $\{y_i\}_{i=1}^n$ such that $\sum_{i=1}^n y_i = \sum_{i=1}^n y_i^2 = n$ we have $y_1 = \cdots = y_n = 1$. Therefore, $\sigma_i(\alpha^\star\alpha) = 1$ for all i, and $\alpha^\star\alpha = 1$, then $\alpha \in \mu_{\mathbb{K}}$ by Lemma 5.10. □

In short, the above lemmata tell us that multiplying a solution in our omSVP game by a root of unity should be considered a trivial win, and disallowed.

Lemma 5.12. *A power of two cyclotomic field $\mathbb{K}$ of conductor $2n$ has $\mu_{\mathbb{K}} = \{X^i : i = 0, \ldots, 2n-1\}$.*

Proof. Recall $R = \mathbb{Z}[X]/(X^n + 1)$ and if $\alpha = \sum_{i=0}^{n-1} \alpha_i X^i$ then $\alpha^\star = \alpha_0 - \sum_{i=1}^{n-1} \alpha_i X^{n-i}$. From $\alpha^\star\alpha = 1$ we therefore have $\sum_{i=0}^{n-1} \alpha_i^2 = 1$ for $\alpha_i \in \mathbb{Z}$. □

Here $i = 0, n$ represent the identity and negation, respectively.

Identifying X with the matrix $X \cdot \mathbf{I}_2 \in \mathrm{GL}_2(R)$, note that the automorphism X of the Hermitian form $\mathbf{I}_2(R)$ provides the following automorphism of the standard basis $\textsc{Rot}(\mathbf{I}_2(R)) = \mathbf{I}_{2n}(\mathbb{Z})$:

$$\begin{pmatrix} x_0 \\ \vdots \\ x_{2n-1} \end{pmatrix} \mapsto (-x_{n-1}, x_0, x_1, \ldots, x_{n-2},\ -x_{2n-1}, x_n, x_{n+1}, \ldots, x_{2n-2})^\mathsf{T}\ .$$

In fact, X is an automorphism for every Hermitian form, similar to how -1 is an automorphism for every lattice.

Luo, Jiang, Pan and Wang observed that given $\mathbf{Q}$ there is another efficiently computable automorphism $\omega_{\mathbf{Q}}$ of $\textsc{Rot}(\mathbf{Q})$ [LJPW24], which we will now explain. Let us write $\overline{\mathbf{B}} = (\mathbf{B}^\star)^\mathsf{T}$ given a matrix (or vector) $\mathbf{B} \in \mathbb{K}^{k\times\ell}$, which compared to Definition 2.66, applies complex conjugation coefficientwise to $\mathbf{B}$ (not $\mathbf{B}^\mathsf{T}$). First, let

$$\mathbf{J}_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

and observe $\mathbf{J}_2^2 = -\mathbf{I}_2(R)$. Via Equation (5.1) we relate the primal and dual $\mathbf{D} = (\mathbf{B}^{-1})^\star$ bases

$$\mathbf{D} = (\mathbf{B}^{-1})^\star = (-\mathbf{J}_2\mathbf{B}^\mathsf{T}\mathbf{J}_2)^\star = -\mathbf{J}_2 \cdot \overline{\mathbf{B}} \cdot \mathbf{J}_2. \tag{5.6}$$

Now, suppose it is publicly known that some vector $\mathbf{x}$ is short with respect to $\mathbf{Q}$, i.e., $\|\mathbf{x}\|_{\mathbf{Q}}$ is small. Then, the vector $\mathbf{z} = \mathbf{B}\cdot\mathbf{x}$ is short, i.e., $\|\mathbf{z}\| = \|\mathbf{B}\mathbf{x}\| = \|\mathbf{x}\|_{\mathbf{Q}}$ is small. Observe that conjugation and multiplication by $\mathbf{J}_2$ preserve inner products. Indeed, $\mathbf{z}' = \mathbf{J}_2 \cdot \overline{\mathbf{z}}$ satisfies $\|\mathbf{z}'\| = \|\mathbf{z}\|$. Using Equation (5.6), we get

$$\mathbf{z}' = \mathbf{J}_2 \cdot \overline{\mathbf{B}\mathbf{x}} = \mathbf{J}_2 \cdot \overline{\mathbf{B}} \cdot \overline{\mathbf{x}} = \mathbf{D} \cdot \mathbf{J}_2 \cdot \overline{\mathbf{x}}.$$

As $\mathbf{B}$ is not known, $\mathbf{z}$ and $\mathbf{z}'$ are not publicly known, but it is known that $\mathbf{J}_2\overline{\mathbf{x}}$ is a short vector with respect to the dual Hermitian form $\mathbf{Q}^{-1}$. Exploiting self-duality of the underlying unstructured lattice $\mathbb{Z}^{2n}$, allows us to find a short vector in terms of $\mathbf{Q}$. We express $\mathbf{z}'$ in terms of the primal basis $\mathbf{B}$:

$$\mathbf{z}' = \mathbf{B} \cdot \underbrace{\mathbf{B}^{-1}\mathbf{D}}_{\mathbf{Q}^{-1}} \cdot \mathbf{J}_2\overline{\mathbf{x}} = \mathbf{B} \cdot \mathbf{Q}^{-1} \cdot \mathbf{J}_2\overline{\mathbf{x}}.$$

Therefore, $\mathbf{x}' = \mathbf{Q}^{-1} \cdot \mathbf{J}_2\overline{\mathbf{x}}$ is also a publicly known lattice point of same $\mathbf{Q}$-norm $\|\mathbf{x}'\|_{\mathbf{Q}} = \|\mathbf{x}\|_{\mathbf{Q}}$. This motivates the following definition, which uses that $\mathbb{Z}^{2n}$ is a symplectic lattice [GHN06].

Definition 5.13 (Symplectic automorphism)**.** The so-called *symplectic automorphism* is the additive bijective map,

$$\omega\colon R^2 \to R^2, \qquad \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} y^\star \\ -x^\star \end{pmatrix} .$$

Given a Hermitian form $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$, the *symplectic automorphism with respect to* $\mathbf{Q}$ is the additive, bijective map

$$\omega_{\mathbf{Q}} \colon R^2 \to R^2, \qquad \mathbf{x} \mapsto \mathbf{Q}^{-1}\omega(\mathbf{x}) = \mathbf{Q}^{-1}\mathbf{J}_2 \cdot \overline{\mathbf{x}} \ .$$

Note that $\omega_{\mathbf{Q}}$ is not an automorphism of $\mathbf{Q}$, because it is not R-linear: $\omega_{\mathbf{Q}}(\alpha\mathbf{x}) = \alpha^\star\omega_{\mathbf{Q}}(\mathbf{x})$ holds for all $\mathbf{x} \in R^2$ and $\alpha \in R$. It is, however, an automorphism of $\textsc{Rot}(\mathbf{Q})$ by considering its representation as a matrix of order $2n$. Therefore, we have $\|\mathbf{x}\|_{\mathbf{Q}} = \|\omega_{\mathbf{Q}}(\mathbf{x})\|_{\mathbf{Q}}$ for all $\mathbf{x} \in R^2$. Using Equation (5.6), we see for any basis $\mathbf{B} \in \mathrm{GL}_2(R)$ and $\mathbf{x} \in R^2$ that

$$\mathbf{B} \cdot \omega_{\mathbf{B}^\star\mathbf{B}}(\mathbf{x}) = (\mathbf{B}^{-1})^\star\mathbf{J}_2\overline{\mathbf{x}} = \mathbf{J}_2\overline{\mathbf{B}\mathbf{x}} = \omega(\mathbf{B}\mathbf{x}). \tag{5.7}$$

This motivates the definition of the following automorphism.

Definition 5.14 (Group of trivial automorphism)**.** Given a Hermitian form $\mathbf{Q} \in \mathcal{H}_2^{>0}(\mathbb{K})$, the *group of trivial automorphisms with respect to* $\mathbf{Q}$ is given by

$$\mathcal{G}_{\mathbf{Q}} = \left\langle X \cdot \mathbf{I}_2(\mathbb{K}), \omega_{\mathbf{Q}} \right\rangle = \bigcup_{i=0}^{2n-1} \left\{ X^i \, \mathbf{I}_2(\mathbb{K}), X^i\omega_{\mathbf{Q}} \right\} \quad \subset \mathrm{O}(\textsc{Rot}(\mathbf{Q})).$$

We simply write $\mathcal{G} = \mathcal{G}_{\mathbf{Q}}$ for $\mathbf{Q} = \mathbf{I}_2(R)$.

This group is isomorphic to the generalised quaternion group Q_{4n}, which has order $4n$. Note, $\omega^2 = -\mathbf{I}_2(R)$. Using Equation (5.7), we get

$$\omega_{\mathbf{Q}}(\omega_{\mathbf{Q}}(\mathbf{x})) = \mathbf{B}^{-1}\omega(\omega(\mathbf{B}\mathbf{x})) = -\mathbf{x}.$$

Average case distribution

We are now ready to define omSVP in its full generality, for certain parameters that will become clear later. Although we focus on rank 2 as $\mathcal{G}_{\mathbf{Q}}$ is only defined for that rank, omSVP could be generalised to other ranks.

Definition 5.15 (Average case omSVP)**.** An *average case omSVP instance* is the pair $\textsc{ac-omSVP} = (\mathsf{Init}, \mathsf{samp})$. On input 1^n, Init returns a form $\mathbf{Q}$ sampled from some distribution over some $\mathcal{H}_2^{>0}(\mathbb{K})$, the roots of unity $\mu_{\mathbb{K}}$ for $\mathbb{K}$, a length bound L_n, and a Gaussian parameter σ_n. On input $\mathbf{Q}$, samp returns a sample from $D_{\mathbf{Q},\sigma_n}$.

$\text{SAMPLE}_{\text{AC-omSVP},\mathcal{A}}(1^n)$

1: $\mathcal{L}_{\mathsf{samples}} \leftarrow \{\mathbf{0}\}$
2: $(\mathbf{Q}, \mu_{\mathbb{K}}, L, \sigma) \leftarrow \mathsf{Init}(1^n)$
3: $\mathbf{w}^\star \leftarrow \mathcal{A}^{\mathsf{samp}(\mathbf{Q})}(\mathbf{Q})$
4: **return** $[\![\|\mathbf{w}^\star\|_{\mathbf{Q}} \leq L \ \wedge \ \mathbf{w}^\star \notin \mathcal{L}_{\mathsf{samples}}]\!]$

$\mathsf{samp}(\mathbf{Q})$

1: $\mathbf{w} \leftarrow D_{\mathbf{Q},\sigma}$
2: $\mathcal{L}_{\mathsf{samples}} \leftarrow \mathcal{L}_{\mathsf{samples}} \cup \{\alpha(\mathbf{w})\}_{\alpha \in \mathcal{G}_{\mathbf{Q}}}$
3: **return** $\mathbf{w}$

Figure 5.7: SAMPLE game

The adversary $\mathcal{A}$ wins the game in Figure 5.7 whenever it can use the form $\mathbf{Q}$ and the samples it receives from samp to return some nontrivially new element of R^2 that is short enough.

Definition 5.16 (SAMPLE security)**.** Let AC-omSVP $= (\mathsf{Init}, \mathsf{samp})$ be an average case omSVP instance. We say AC-omSVP is (t, ε, q_o)-*SAMPLE secure*, if for any adversary $\mathcal{A}$ running in time at most t, and making at most q_o queries to samp, we have

$$\Pr[\text{SAMPLE}_{\text{AC-omSVP},\mathcal{A}} = 1] \leq \varepsilon.$$

Smoothing signatures

In the following, we will implicitly use the particular section $s\colon R/2R \to R$ of the natural surjection $R \to R/2R$ given by $s(h + 2R) = h_0 + h_1 X + \cdots h_{n-1} X^{n-1}$ with $(h_0, \ldots, h_{n-1}) \in \{0,1\}^n$. Thus, reduction modulo 2 can also be considered as a map $R \to R$ via s.

Definition 5.17. Let $\mathbf{B} \in \mathrm{GL}_2(R)$ and $\mathbf{Q} = \mathbf{B}^\star\mathbf{B}$. For every $\sigma > 0$ and $\mathbf{h} \in (R/2R)^2 \subset R^2$ we consider the following three $\mathbf{Q}$-dependent distributions.

- $\mathcal{D}_{\mathbf{Q},\sigma}$, the discrete Gaussian distribution over R^2 under $\|\cdot\|_{\mathbf{Q}}$ given by

$$\mathcal{D}_{\mathbf{Q},\sigma}\colon R^2 \to \mathbb{R}, \quad \mathbf{x} \mapsto \frac{1}{\sum_{\mathbf{y} \in R^2} \rho_{\mathbf{Q},\sigma}(\mathbf{y})} \rho_{\mathbf{Q},\sigma}(\mathbf{x}).$$

- $\widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}[\mathbf{h}]$, the discrete Gaussian distribution over $\mathbf{h} + 2R^2 \subset R^2$ under $\|\cdot\|_{\mathbf{Q}}$ given by

$$\widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}[\mathbf{h}] \colon \mathbf{h} + 2R^2 \to \mathbb{R}, \quad \mathbf{x} \mapsto \frac{1}{\sum_{\mathbf{y} \in \mathbf{h}+2R^2} \rho_{\mathbf{Q},\sigma}(\mathbf{y})} \rho_{\mathbf{Q},\sigma}(\mathbf{x}).$$

- $\widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}$, the distribution implied by sampling a uniform $\mathbf{h} \leftarrow (R/2R)^2$ and returning a sample from $\widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}[\mathbf{h}]$.

Given $\mathbf{B}$ we may sample the above distributions by sampling according to $\mathcal{D}_{\mathbf{I}_2(R),\sigma}$, in the coset defined by $\mathbf{t} = \mathbf{B}\mathbf{h}$ if required, and multiplying by $\mathbf{B}^{-1}$ as in Hawk signing, see Algorithm 5.2. Note that for every $\mathbf{h} \in (R/2R)^2$ and any outcome $\mathbf{w} \leftarrow \widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}[\mathbf{h}]$ we have $\mathbf{w} \equiv \mathbf{h} \pmod 2$. A direct consequence of Lemma 2.84 is the following result, which tells us that $\mathcal{D}_{\mathbf{Q},2\sigma}$ assigns a similar probability to each coset $\mathbf{h} + 2R^2$.

Lemma 5.18. *Let $\sigma \geq \eta_\varepsilon(\mathbb{Z}^{2n})$ for some $\varepsilon \in (0,1)$. Then for all fixed $\mathbf{h} \in (R/2R)^2$ we have*

$$\Pr_{\mathbf{w} \leftarrow \mathcal{D}_{\mathbf{Q},2\sigma}} [\mathbf{w} \equiv \mathbf{h} \pmod 2] \leq 2^{-2n} \cdot \frac{1+\varepsilon}{1-\varepsilon}.$$

We will use the Rényi divergence of $\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma}$ from $\mathcal{D}_{\mathbf{Q},2\sigma}$, which can be computed using following lemma.

Lemma 5.19. *Let $\sigma > 0$, $\mathbf{Q} = \mathbf{B}^\star\mathbf{B}$ for some $\mathbf{B} \in \mathrm{GL}_2(R)$ and $a \in (1,\infty)$ be an order. Furthermore, let*

$$\alpha = \frac{\rho_{2\sigma}(\mathbb{Z})}{2\,\rho_\sigma(\mathbb{Z})}, \quad \text{and} \quad \beta = \frac{\rho_{2\sigma}(\mathbb{Z})}{2\,\rho_\sigma(\mathbb{Z} + \frac{1}{2})}.$$

Then

$$\mathrm{R}_a\Big(\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma} \,\Big\|\, \mathcal{D}_{\mathbf{Q},2\sigma}\Big) = \left(\frac{\alpha^{a-1} + \beta^{a-1}}{2}\right)^{\frac{2n}{a-1}}.$$

Proof. Since each $\mathbf{w} \in R^2$ lies in $2R^2 + \mathbf{h}$ for exactly one $\mathbf{h} \in (R/2R)^2$ we have $\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma}(\mathbf{w}) = 2^{-2n}\,\rho_{\mathbf{Q},2\sigma}(\mathbf{w}) / \rho_{\mathbf{Q},2\sigma}(2R^2 + \mathbf{h})$. Similarly, we have $\mathcal{D}_{\mathbf{Q},2\sigma}(\mathbf{w}) = \rho_{\mathbf{Q},2\sigma}(\mathbf{w}) / \rho_{\mathbf{Q},2\sigma}(R^2)$. Hence, by definition of the Rényi

divergence,

$$\mathrm{R}_a\Big(\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma} \,\Big\|\, \mathcal{D}_{\mathbf{Q},2\sigma}\Big)^{a-1} = \sum_{\substack{\mathbf{h}\in(R/2R)^2 \\ \mathbf{w}\in 2R^2+\mathbf{h}}} \widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma}(\mathbf{w})^a / \mathcal{D}_{\mathbf{Q},2\sigma}(\mathbf{w})^{a-1}$$

$$= \sum_{\mathbf{h}\in(R/2R)^2} \frac{1}{2^{2n}} \left(\frac{\rho_{\mathbf{Q},2\sigma}(R^2)}{2^{2n}\,\rho_{\mathbf{Q},2\sigma}(2R^2+\mathbf{h})}\right)^{a-1} \sum_{\mathbf{w}\in 2R^2+\mathbf{h}} \frac{\rho_{\mathbf{Q},2\sigma}(\mathbf{w})}{\rho_{\mathbf{Q},2\sigma}(2R^2+\mathbf{h})} \ .$$

Note that $\sum_{\mathbf{w}} \rho_{\mathbf{Q},2\sigma}(\mathbf{w}) / \rho_{\mathbf{Q},2\sigma}(2R^2+\mathbf{h}) = 1$, simplifying the equation to

$$\mathrm{R}_a\Big(\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma} \,\Big\|\, \mathcal{D}_{\mathbf{Q},2\sigma}\Big)^{a-1} = \frac{1}{2^{2n}} \sum_{\mathbf{h}\in(R/2R)^2} \left(\frac{\rho_{\mathbf{Q},2\sigma}(R^2)}{2^{2n}\,\rho_{\mathbf{Q},2\sigma}(2R^2+\mathbf{h})}\right)^{a-1}$$

$$= \frac{1}{2^{2n}} \sum_{\mathbf{h}\in(R/2R)^2} \left(\frac{\rho_{2\sigma}(R^2)}{2^{2n}\,\rho_{2\sigma}(2R^2+\mathbf{B}\cdot\mathbf{h})}\right)^{a-1} ,$$

where the second equality relies on $\mathbf{B} \in \mathrm{GL}_2(R)$ and therefore $\mathbf{B}R^2 = R^2$. Note $\left\{2R^2+\mathbf{Bh} : \mathbf{h}\in(R/2R)^2\right\} = \left\{2R^2+\mathbf{t} : \mathbf{t}\in(R/2R)^2\right\}$, so we may sum over $\mathbf{t}\in(R/2R)^2$. We also pass to the unstructured setting, noting that $\rho_{2\sigma}(R^2) = \rho_{2\sigma}(\mathbb{Z}^{2n})$ and that, since $\mathbb{Z}^{2n}$ has an orthogonal basis $\mathbf{I}_{2n}(\mathbb{Z})$, $\rho_{2\sigma}(\mathbb{Z}^{2n}) = \rho_{2\sigma}(\mathbb{Z})^{2n}$. Similarly, we have $\rho_{2\sigma}(2R^2+\mathbf{t}) = \rho_{2\sigma}(2\mathbb{Z}^{2n}+\mathbf{t}) = \prod_{i=0}^{2n-1}\rho_{2\sigma}(2\mathbb{Z}+t_i)$, with $\mathbf{t}\in\{0,1\}^{2n}$ after the first equality. Moreover, $\rho_{2\sigma}(2\mathbb{Z}+b) = \rho_\sigma(\mathbb{Z}+b/2)$ holds ($b\in\{0,1\}$). Thus,

$$\mathrm{R}_a\Big(\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma} \,\Big\|\, \mathcal{D}_{\mathbf{Q},2\sigma}\Big)^{a-1} = \frac{1}{2^{2n}} \sum_{\mathbf{t}\in\{0,1\}^{2n}} \left(\frac{\rho_{2\sigma}(\mathbb{Z}^{2n})}{2^{2n}\,\rho_{2\sigma}(2\mathbb{Z}^{2n}+\mathbf{t})}\right)^{a-1}$$

$$= \frac{1}{2^{2n}} \sum_{\mathbf{t}\in\{0,1\}^{2n}} \prod_{i=0}^{2n-1} \left(\frac{\rho_{2\sigma}(\mathbb{Z})}{2\,\rho_\sigma(\mathbb{Z}+\frac{t_i}{2})}\right)^{a-1} .$$

By swapping these finite sums and products, and considering the contribution of a single t_i, half of which are zero and half of which are one over $\mathbf{t}\in\{0,1\}^{2n}$ for any fixed index i, we have

$$\mathrm{R}_a\Big(\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma} \,\Big\|\, \mathcal{D}_{\mathbf{Q},2\sigma}\Big)^{a-1} = \frac{1}{2^{2n}} \prod_{i=0}^{2n-1} \left[\left(\frac{\rho_{2\sigma}(\mathbb{Z})}{2\,\rho_\sigma(\mathbb{Z})}\right)^{a-1} + \left(\frac{\rho_{2\sigma}(\mathbb{Z})}{2\,\rho_\sigma(\mathbb{Z}+\frac{1}{2})}\right)^{a-1}\right]$$

$$= \left(\frac{\alpha^{a-1}+\beta^{a-1}}{2}\right)^{2n} ,$$

using the definitions of α and β. □

As σ grows the Rényi divergence decreases towards one. When σ is not too big, α and β in the above lemma can be efficiently numerically computed to any desired precision via a tail cut.

The following lemma, which is an adaptation of [DPPvW22b, Lem. 10] and [FH23, Lem. 1] to the symplectic automorphism, proves that a forgery in the SUF-CMA game of Hawk is with overwhelming probability a nontrivially new solution to ac-omSVP.

Lemma 5.20. *Let n be a power of 2, $\varepsilon \in (0,1)$, $\sigma \geq \eta_\varepsilon(\mathbb{Z}^{2n})$. For a sample $\mathbf{w} \leftarrow \mathcal{D}_{\mathbf{Q},2\sigma}$ we have*

$$\Pr_{\mathbf{w}}\left[\exists \alpha \in \mathcal{G}_{\mathbf{Q}} \setminus \{1,-1\} \colon \frac{\mathbf{h}+\alpha(\mathbf{w})}{2} \in R^2\right] \leq \frac{n+1}{2^n} \cdot \frac{1+\varepsilon}{1-\varepsilon}\ , \tag{5.8}$$

where $\mathbf{h} = \mathbf{w} \bmod 2$. For a fixed $\mathbf{h}^\circ \in (R/2R)^2$ we have

$$\Pr_{\mathbf{w}}\left[\exists \alpha \in \mathcal{G}_{\mathbf{Q}} \colon \frac{\mathbf{h}^\circ+\alpha(\mathbf{w})}{2} \in R^2\right] \leq \frac{2n}{2^{2n}} \cdot \frac{1+\varepsilon}{1-\varepsilon}\ . \tag{5.9}$$

Proof. To prove Equation (5.8), we split $\mathcal{G}_{\mathbf{Q}} = \mu_{\mathbb{K}} \cup \{\alpha\omega_{\mathbf{Q}} \mid \alpha \in \mu_{\mathbb{K}}\}$ in two sets and use a union bound. Moreover, for $\alpha \in \mathcal{G}_{\mathbf{Q}}$, note $(\mathbf{h}+\alpha(\mathbf{w}))/2 \in R^2$ happens if and only if $\mathbf{w} \equiv \alpha(\mathbf{w}) \pmod 2$ holds, as -1 acts trivially modulo $2R^2$.

First, if we have $\mathbf{w} \equiv \alpha\mathbf{w} \pmod 2$ for $\alpha \in \mu_{\mathbb{K}} \setminus \{1,-1\}$, then this statement is also true when replacing α by $\alpha^2, \alpha^4, \ldots, \alpha^{2n} = 1$. Now, Lemma 5.12 implies $\mathbf{w} \equiv X^{n/2}\mathbf{w} \pmod 2$. Note this event is rare as 2^n vectors $\mathbf{h} \in (R/2R)^2$ satisfy $\mathbf{h} \equiv X^{n/2}\mathbf{h} \pmod 2$. Hence, using Lemma 5.18 and a union bound over all such $\mathbf{h}$, we get

$$\Pr_{\mathbf{w}}\left[\exists \alpha \in \mu_{\mathbb{K}} \setminus \{1,-1\} \colon \frac{\mathbf{h}+\alpha\mathbf{w}}{2} \in R^2\right] \leq 2^n \cdot 2^{-2n}\frac{1+\varepsilon}{1-\varepsilon}.$$

Second, suppose we have $\mathbf{w} \equiv \alpha(\mathbf{w}) \pmod 2$ for some $\alpha \in \mathcal{G}_{\mathbf{Q}} \setminus \mu_{\mathbb{K}}$, i.e., for some $i \in \{0,1,\ldots,n-1\}$ we have $\mathbf{w} \equiv X^i \cdot \omega_{\mathbf{Q}}(\mathbf{w}) \pmod 2$ by Lemma 5.12. By a similar union bound and Lemma 5.18, now we have

$$\Pr_{\mathbf{w}}\left[\exists \beta \in \mu_{\mathbb{K}} \colon \frac{\mathbf{h}+\beta\omega_{\mathbf{Q}}(\mathbf{w})}{2} \in R^2\right] \leq 2^{-2n}\frac{1+\varepsilon}{1-\varepsilon} \cdot \sum_{i=0}^{n-1}|C_i|,$$

where $C_i = \{\mathbf{h} \in (R/2R)^2 \mid \mathbf{h} \equiv X^i \omega_{\mathbf{Q}}(\mathbf{h}) \pmod 2\}$. By taking $\mathbf{z} = (z_0, z_1)^\mathsf{T} = \mathbf{B}\mathbf{h}$, where $\mathbf{Q} = \mathbf{B}^\star\mathbf{B}$, we have both $z_0 \equiv X^i z_1^\star \pmod 2$ and $z_1 \equiv X^i z_0^\star \pmod 2$. Note that both conditions are equivalent, and any choice for $z_0 \in R/2R$ uniquely determines z_1. Hence, $|C_i| = 2^n$ for all i, which implies

$$\Pr_{\mathbf{w}}\left[\exists \alpha \in \mathcal{G}_{\mathbf{Q}} \setminus \mu_{\mathbb{K}} \colon \frac{\mathbf{h} + \alpha(\mathbf{w})}{2} \in R^2\right] \leq \frac{n}{2^n} \cdot \frac{1+\varepsilon}{1-\varepsilon}.$$

To prove Equation (5.9), observe $(\mathbf{h}^\circ + \alpha(\mathbf{w}))/2 \in R^2$ holds if and only if $\mathbf{w} \in \alpha^{-1}(\mathbf{h}^\circ) + 2R^2$. Since multiplication by -1 acts trivially modulo 2, there are at most $2n$ cosets of $2R^2$ to consider for $\mathbf{w}$ such that $\frac{1}{2}(\mathbf{h}^\circ + \alpha(\mathbf{w})) \in R^2$ holds for some $\alpha \in \mathcal{G}_{\mathbf{Q}} \setminus \mu_{\mathbb{K}}$. Together with Lemma 5.18, this proves Equation (5.9). □

Reduction

We now present the theorem that any ROM-SUF-CMA adversary against Hawk can be used as a subroutine for a SAMPLE adversary against an instance of ac-omSVP, while only losing a constant number of bits. There is also a reduction in the quantum ROM [FH23].

Below, let $\Pi_{\text{Hawk}}^{\text{tables}} = (\text{KeyGen}, \text{Sign}', \text{Verify})$ be Hawk as in Section 5.2.2, where Sign′ generates compressed signatures using precomputed tables like in the implementation, see Section 5.5.2. Moreover, $\kappa \in \mathbb{Z}_{\geq 1}$ denotes the maximum number of samples from samp needed to produce a valid Hawk signature. As invalid signatures are rare, one may take $\kappa = \lambda + 67$.

Theorem 5.21 (omSVP to SUF-CMA reduction)**.** *Let $\lambda \in \{128, 256\}$, $q_s \leq 2^{64}$ and $q_h \leq 2^\lambda$. Let $\mathcal{A}$ be a (quantum) adversary against the* (Q)SUF-CMA *game against $\Pi_{\text{Hawk}}^{\text{tables}}$ that runs in time at most $t_{\mathcal{A}} \geq 1$, makes (q_s, q_h) queries to $(\mathsf{OSign}, \mathsf{H})$, respectively, and has success probability*

$$\varepsilon_{\mathcal{A}} = \mathsf{Adv}^{\text{SUF-CMA}}_{\Pi_{\text{Hawk}}^{\text{tables}}, \mathcal{A}},$$

such that $t_{\mathcal{A}}/\varepsilon_{\mathcal{A}} < 2^\lambda$. In particular $\lambda = 128$ corresponds to Hawk-512 *and $\lambda = 256$ corresponds to* Hawk-1024*. Moreover, assume* $\mathsf{saltlen}$ *is taken arbitrarily large (in particular, at least as large as listed in Table 5.3).*

Then, there exists a (quantum) adversary $\mathcal{B}$ against the SAMPLE *game with the appropriate $\mathsf{Init}(1^n)$ running in time at most $t_{\mathcal{B}} = t_{\mathcal{A}} +$*

$\mathsf{Overhead}(\kappa \cdot q_s, q_h)$. *This overhead consists of making* $\kappa \cdot q_s$ *queries to* samp *and simulating* q_h *queries to* H. *The success probability of* $\mathcal{B}$ *satisfies*

$$\mathsf{Adv}^{\text{SAMPLE}}_{\text{AC-OMSVP},\mathcal{B}} > \frac{1}{2}\varepsilon_{\mathcal{A}}.$$

Proof. See Section 6 of https://hawk-sign.info/hawk-spec.pdf. □

Remark 5.22. We do not expect the reduction to be bidirectional. Intuitively even if one can find a $\mathbf{w}^\star$ that wins the SAMPLE game, one must also find a message and salt that hashes into a particular coset to make this a successful signature forgery.

Discussion

The AC-OMSVP instances implicit in our reduction from omSVP to Hawk are trivially solvable. In particular, without making any queries to the samp oracle, an adversary can submit $\mathbf{w}^\star = (1, 0)^\mathsf{T}$. In this case $\|\mathbf{w}^\star\|_{\mathbf{Q}} = \|(f, g)\| \approx \sqrt{2n}\sigma_{\mathsf{pk}} \leq 2\sqrt{2n}\sigma_{\mathsf{ver}} = L$. This is a consequence of us choosing σ_{pk} as small as our practical cryptanalysis will allow us, see Section 5.3.2, and not being able to choose σ_{ver} small enough to mitigate this without unacceptably increasing the restart probability. This is an instance of us choosing to follow practical cryptanalysis for setting parameters, and using our theoretical reduction to convince us of the soundness of our design; we could easily fix this by increasing σ_{pk}, but this would also increase the size of our public keys. We note that, while this decision makes our reduction vacuous, we do not know practically how to use these public key vectors to create forgeries against Hawk. More generally, one can consider an adversary that performs lattice reduction on the form $\mathbf{Q}$ and possibly combines the result with samples from the samp oracle.

In a more idealised setting, we are effectively asking a omSVP adversary to perform a relaxed notion of lattice sieving for just one vector. The standard heuristic from that literature is that the vectors $\mathbf{w}$ from samp are i.i.d. uniform points on a sphere of radius $2\sqrt{2n}\sigma_{\mathsf{sign}}$ [NV08]. Rather than asking for enough distinct shorter combinations of such $\mathbf{w}$ to recurse a sieving operation, the SAMPLE game asks for just one combination of $\mathbf{w}$, provided it does not fall into $\mathcal{L}_{\mathsf{samples}}$, that can actually be slightly longer, by a factor of $\sigma_{\mathsf{ver}}/\sigma_{\mathsf{sign}} \geq 1$. Optimising k-sieving techniques [BLS16, HK17] for this setting where a single relaxed combination is sought may be an interesting approach. Also, whether having

access to (lattice reduction on) $\mathbf{Q}$ can help an adversary in this setting, or whether one can reduce to a variant of omSVP where the adversary is not given $\mathbf{Q}$ by showing something equivalent can be obtained efficiently by using the samp oracle, are other interesting avenues for further work.

Finally, the applicability of learning style attacks [NR09, DN12b] to the particular ac-omSVP instances relevant to Hawk should be considered. Lemma 5.19 tells us that $\widetilde{\mathcal{D}}_{\mathbf{Q},2\sigma_{\mathsf{sign}}}$ and $\mathcal{D}_{\mathbf{Q},2\sigma_{\mathsf{sign}}}$ are close, which relate to $\mathbf{w}$ sampled internally during Hawk signing and the output of the samp oracle in the SAMPLE game. Recall that $\mathbf{w}$ is public in Hawk signing since it can be computed from $\mathbf{s}$ and $\mathbf{h}$. Therefore, if one can mount such a learning style attack against Hawk signatures, then one can mount such a learning attack against the SAMPLE game using $\mathbf{w}$ from samp with a very similar probability.

Chapter 6

Solving Module-LIP using Automorphisms

This chapter is based on joint work with D.M.H van Gent published in IACR's Communications in Cryptology [vGP25].

The search rank-2 module Lattice Isomorphism Problem, or smLIP, over a cyclotomic ring of degree a power of two, can be reduced to a LIP instance of at most half the rank if an adversary knows a nontrivial automorphism of the underlying integer lattice. Knowledge of such a nontrivial automorphism speeds up the key recovery attack on Hawk at least quadratically, which would halve the number of security bits.

The initial version of omSVP was trivially broken by the symplectic automorphism, and it was easily made hard again by including the symplectic automorphism as a trivial win. However, the question remains if more easily computable automorphisms exist. This work provides confidence in the soundness of this updated definition, assuming smLIP is hard, since there are plausibly no more trivial automorphisms that allow winning the omSVP game easily.

Although this work does not affect the security of Hawk, it opens up a new attack avenue involving the automorphism group that may be theoretically interesting on its own.

6.1 Introduction

The *Lattice Isomorphism Problem* (LIP) has recently been introduced as a building block for cryptography [DvW22, BGPS23], and was used in Chapter 5 to develop the fast and compact signature scheme named Hawk. LIP for the integer lattice ($\mathbb{Z}$LIP) has been studied well [Szy03, BM21, Duc24, BN24]. Moreover, LIP for *ideal lattices*, i.e., rank-1 module lattices, is solvable in polynomial time using the so-called Gentry–Szydlo algorithm [GS02, LS17, LS19]. However, the SUF-CMA security of Hawk, which uses a rank-2 module lattice for efficiency reasons, is based on a new problem called *one more shortest vector problem* (omSVP). Although omSVP has received some attention so far [DPPvW22a, LJPW24], it needs to be studied more extensively in order to gain confidence in Hawk.

Private key recovery for Hawk has received significantly more attention than omSVP. The former requires solving the so-called *search module LIP* (smLIP) between free modules of rank 2 over a (totally complex) cyclotomic number field. A recent line of work [MPPW24, LJPW24, APvW25] has developed a polynomial-time attack on smLIP over a number field with at least one real embedding, which remarkably is not the case with Hawk. Instead, smLIP for Hawk reduces uniformly to a principal ideal problem in a quaternion algebra [CME+25], and it remains unknown whether Gentry–Szydlo, the algorithm for the case of number fields, generalises to such algebras. Thus far, it seems smLIP is not much easier than $\mathbb{Z}$LIP despite the extra module structure.

The signature distribution of Hawk cannot be simulated as easily as that of NTRU-based signatures see Section 5.6.2, so it is not known how to instantiate the framework of [GPV08], which prevents leakage of the trapdoor basis [NR09, DN12b]. Instead, security of Hawk is based on omSVP, in which an adversary, given an oracle that produces short vectors, needs to output a short vector that is not a "trivial automorphism", such as negation, of any oracle output. The parameters of Hawk are chosen based on practical cryptanalysis. For these parameters, the security reduction to omSVP produces a trivial instance of omSVP. Nevertheless, the reduction shows soundness of the design, and larger parameters reduce to a presumably hard instance of omSVP but are unfavourable for the compactness of Hawk.

The formulation of omSVP is a delicate matter, because it needs

to include all trivial automorphisms that an adversary can compute easily. In the original formulation [DPPvW22a], the trivial automorphisms were believed to be multiplications by roots of unity, provided by the module structure in Hawk. Later, however, another trivial automorphism, stemming from the self-duality of the module lattice, was discovered [LJPW24]. This so-called *symplectic automorphism* solved the original omSVP trivially. Hence, when Hawk advanced to the second round of NIST's standardisation process of additional digital signature schemes [BBD+24], its specification document was amended to include the symplectic automorphism in the definition of omSVP. In other words, two vectors are deemed equivalent if they are in the same orbit under the group generated by the roots of unity and the symplectic automorphism. Although this amendment presumably makes omSVP harder, it does not provide a satisfactory answer to the following question.

Question 6.1. Which automorphisms can be computed easily by an adversary against the omSVP game?

6.1.1 Contributions

Our main result is the following theorem, which regards $\mathbb{Z}^{2n}$ as a module lattice R^2 over the ring $R = \mathbb{Z}[X]/(X^n + 1)$. The group $\mathcal{G}_{\mathbf{Q}}$ is defined in Definition 5.14.

Theorem 6.2. *Given a Hermitian form* $\mathbf{Q}$ *of the module lattice* R^2*, and an automorphism* $\mathbf{U} \in \mathrm{GL}_{2n}(\mathbb{Z}) \setminus \mathcal{G}_{\mathbf{Q}}$ *of* $\mathrm{Rot}(\mathbf{Q})$*, then, using standard lattice reduction heuristics, one can solve smLIP for* $\mathbf{Q}$ *by running BKZ-*β *with* $\beta = n/2 + 1$ *on some sublattice* Λ *constructed in polynomial time from* $\mathbf{U}$ *and that has rank at most* n*.*

Informally, one may think of Theorem 6.2 as follows: if one has a rotation of the module lattice R^2 and a nontrivial automorphism of the underlying integer lattice, one can recover that rotation by finding a shortest vector in a sublattice of half the rank and containing an unusually short vector.

Theorem 6.2 shows that with a nontrivial automorphism one can construct a specific lattice Λ of rank at most n. In Section 6.3, we show that BKZ-β heuristically recovers a shortest vector in such Λ for $\beta = n/2+1$, basically because this lattice is as unusual as (and similar to) $\mathbb{Z}^n$. The heuristics for $\mathbb{Z}^n$ have been experimentally verified [DPPvW22a],

and it is proved [Duc24] that BKZ-β recovers a unit vector in $\mathbb{Z}^n$ when $\beta = n/2 + o(n)$. Thus, Theorem 6.2 provides a *quadratic speed up* in solving smLIP, since BKZ-β runs in time exponential in β. Roughly speaking, if it is easy to find a nontrivial automorphism, then so is smLIP. We view this as evidence that computing a nontrivial automorphism is hard, although it may as well be that smLIP is just easier than what is currently assumed.

If we have an oracle for a *random* automorphism, then [JWL+23] solves SVP significantly faster. Even without the oracle, if the automorphism is random we can with high probability obtain a much better result. It is unreasonable to assume, however, that if an attacker obtains a nontrivial automorphism it will be a truly random one. More likely, it will be highly structured, like the symplectic automorphism. Hence, Theorem 6.2 justifies the choice for $\mathcal{G}$, *the group of trivial automorphisms*, in the definition of omSVP [BBD+26]. We know that $\mathcal{G}$ must include the roots of unity *and* the symplectic automorphism [LJPW24], because one can trivially win the omSVP game otherwise. Conversely, the theorem shows that smLIP and hence omSVP becomes much easier if one knows an additional automorphism $\sigma \notin \mathcal{G}$. If there is any choice of $\mathcal{G}$ that would make omSVP hard, then the current $\mathcal{G}$ is the smallest such choice.

The situation is summarised in Figure 6.1.

6

6.2 Main result

In this section we will prove the main result. Recall the definitions of $\mathcal{G}$ and $\mathcal{G}_{\mathbf{Q}}$ from Definition 5.14, in particular $\mathcal{G} = \langle X, \omega \rangle$.

Lemma 6.3. *The group $G = \mathcal{G}$ satisfies all the following properties.*

1. *G acts regularly, i.e., transitively and freely, on the shortest vectors of the lattice R^2;*
2. *G has exactly one element of order 2, namely $-1 = X^n = \omega^2$;*
3. *G is isomorphic to $(\mu \rtimes \langle\omega\rangle)/\langle -1\rangle \cong (C_{2n} \rtimes C_4)/C_2$, where ω acts on μ by $X \mapsto X^\star = X^{-1}$,*
4. *the multiplication map $\mu \times \{1, \omega\} \to G$ is a bijection.*

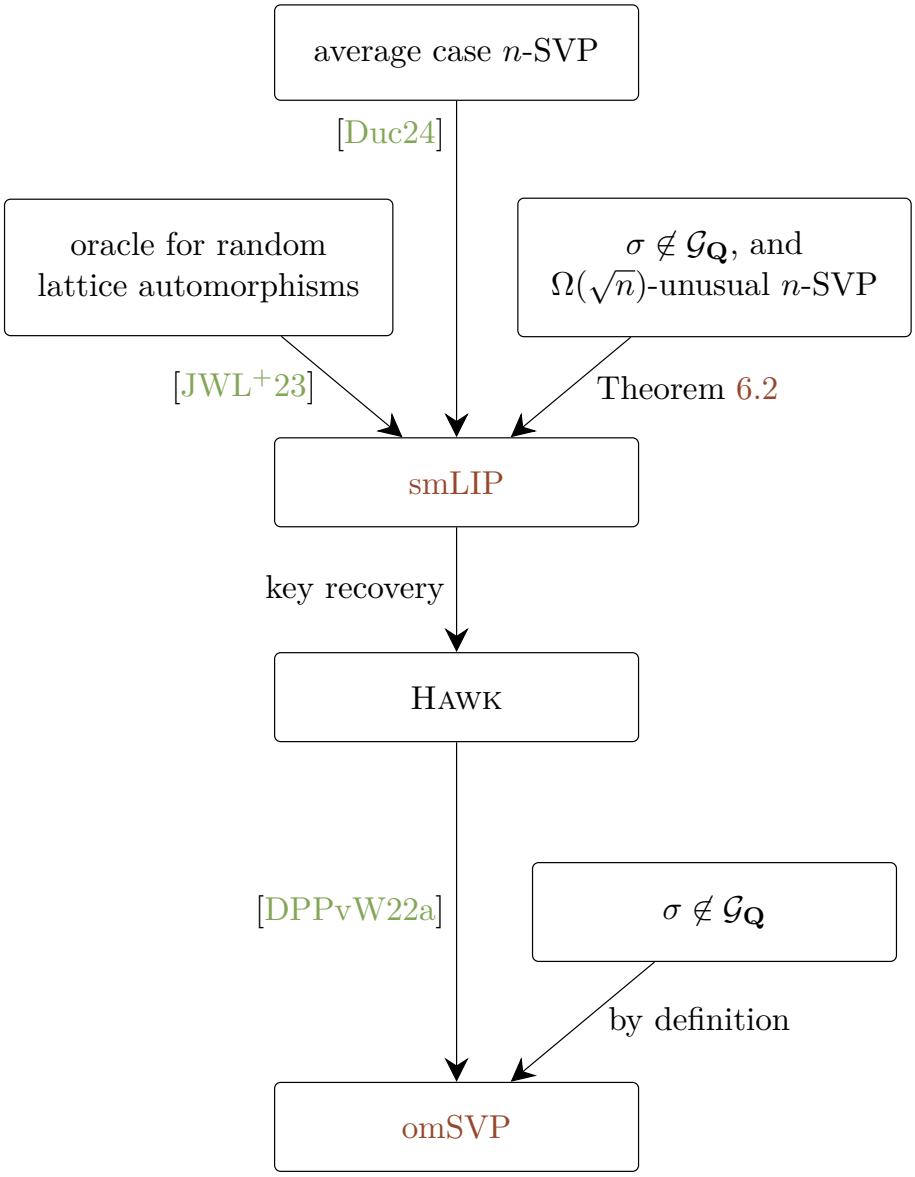


Figure 6.1: Reductions between problems related to the SUF-CMA security of Hawk. An arrow $A \to B$ indicates B reduces to A, i.e., B can be solved in polynomial time by making multiple queries to an algorithm that solves A.

Proof. For all $x, y \in R$ we have

$$\omega X \begin{pmatrix} x \\ y \end{pmatrix} = \omega \begin{pmatrix} Xx \\ Xy \end{pmatrix} = \begin{pmatrix} +X^\star \cdot y^\star \\ -X^\star \cdot x^\star \end{pmatrix} = X^\star \begin{pmatrix} +y^\star \\ -x^\star \end{pmatrix} = X^\star \omega \begin{pmatrix} x \\ y \end{pmatrix},$$

so $\omega X \omega^{-1} = X^\star$. It follows that $\mu \rtimes \langle \omega \rangle \to G$ is a (surjective) group homomorphism. Its kernel consists of all pairs (X^i, ω^j) such that $X^i = \omega^{-j}$. Since $\omega^2 = -1 \in \mu$ and $\omega \notin \mu$, we conclude that the kernel is generated by $(-1, -1)$, establishing the group structure. It follows that the multiplication map $\mu \times \{1, \omega\} \to G$ is a bijection.

In μ, the only element of order 2 is -1. As $(X^i\omega)^2 = X^i \omega X^i \omega = X^i X^{-i} \omega\omega = \omega^2 = -1$, we conclude that -1 is the only element of G of order 2.

The set S of shortest vectors of R^2 equals $(\mu \times \mathbf{0}) \sqcup (\mathbf{0} \times \mu)$. It is easy to see that the action of G on S is transitive: under $\mu \subseteq G$ there are the two orbits $\mu \times \mathbf{0}$ and $\mathbf{0} \times \mu$, while ω interchanges the two. Suppose $\mathbf{x} \in S$ is fixed by $X^i\omega^j$ with $j \in \{0, 1\}$. Then $j = 0$, otherwise $\mathbf{x}$ and $\omega^j \mathbf{x}$ are in different orbits under μ. It follows that $X^i \mathbf{x} = \mathbf{x}$, which is only possible if $X^i = 1$. Hence, the action is free. □

Lemma 6.4. *There exists a polynomial time algorithm that, given a Hermitian form* $\mathbf{Q}$ *of* R^2 *and* $\mathbf{x} \in R^2$ *of length* 1 *or* $\sqrt{2}$*, computes all shortest vectors of* $\mathbf{Q}$*.*

Proof. By [LJPW24] we may compute $G = \mathcal{G}_\mathbf{Q}$.

Case $\|\mathbf{x}\| = 1$: We may compute by Lemma 6.3 all shortest vectors as the orbit of $\mathbf{x}$ under G.

Case $\|\mathbf{x}\| = \sqrt{2}$: Compute for each $\sigma \in G \setminus \{1\}$ the element

$$\mathbf{y}_\sigma = \Big(\sum_{i=0}^{k_\sigma - 1} \sigma^i \Big)(\mathbf{x}) \ ,$$

where $k_\sigma = \mathrm{ord}(\sigma)/2$. We will show that $\mathbf{y}_\sigma$ is twice a shortest vector for some σ, in which case we may reduce to the previous case with $\mathbf{x} \leftarrow \mathbf{y}_\sigma/2$. Note that $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ for some shortest vectors $\mathbf{x}_1 \neq \mathbf{x}_2$. By Lemma 6.3 there is some $\sigma \in G$ such that $\sigma(\mathbf{x}_1) = \mathbf{x}_2$, and $\sigma^{k_\sigma} = -1$. Then

$$\begin{aligned} \mathbf{y}_\sigma &= \sum_{i=0}^{k_\sigma - 1} \sigma^i(\mathbf{x}_1) - \sum_{i=0}^{k_\sigma - 1} \sigma^i(\mathbf{x}_2) = \sum_{i=0}^{k_\sigma - 1} \sigma^i(\mathbf{x}_1) - \sum_{i=0}^{k_\sigma - 1} \sigma^{i+1}(\mathbf{x}_1) \\ &= \mathbf{x}_1 - \sigma^{k_\sigma}(\mathbf{x}_1) = 2\mathbf{x}_1, \end{aligned}$$

as was to be shown. □

Proposition 6.5. *There is a polynomial-time reduction from smLIP on a Hermitian form* $\mathbf{Q}$ *of* R^2 *to finding a nonzero vector of length at most* $\sqrt{2}$ *of* $\mathbf{Q}$.

Proof. Let $\mathbf{Q}$ be a Hermitian form of R^2 and suppose we have such a vector. By Lemma 6.4 we may compute the shortest vectors. The action of μ partitions them into two orbits, and let $\mathbf{b}_1$ and $\mathbf{b}_2$ be elements of the two orbits. With $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ we have $\mathbf{B}^\star\mathbf{Q}\mathbf{B} = \mathbf{I}_2$, so we may return $\mathbf{B}^{-1}$. □

In the lemma below, recall A_k is the root lattice of type A from Example 2.14.

Lemma 6.6. *Let* $\sigma \in \mathrm{O}(\mathbb{Z}^n)$ *with order dividing a prime* p *and let* $\Phi_p(Z) = Z^{p-1} + Z^{p-2} + \cdots + 1$. *Then the sublattices* $\Lambda_1 = \ker(\sigma - 1)$ *and* $\Lambda_2 = \ker(\Phi_p(\sigma))$ *of* $\mathbb{Z}^n$ *satisfy* $\mathrm{rk}(\Lambda_1) + \mathrm{rk}(\Lambda_2) = n$. *Moreover, we have*

$$\Lambda_1 \cong \mathbb{Z}^a \times \sqrt{p} \cdot \mathbb{Z}^c \quad \text{and} \quad \Lambda_2 \cong \mathbb{Z}^b \times A_{p-1}^c, \tag{6.1}$$

for some $a, b, c \in \mathbb{N}$, *with* $b = 0$ *if* $p > 2$. *If* $\sigma \neq \pm 1$, *we have* $\Lambda_1, \Lambda_2 \neq \mathbf{0}$.

Proof. Since σ is a signed permutation, it partitions the coordinates of $\mathbb{Z}^n$. This gives a decomposition $\mathbb{Z}^n = L_1 \oplus \cdots \oplus L_k$ into pairwise orthogonal sublattices, with each L_i isomorphic to either $\mathbb{Z}$ or $\mathbb{Z}^p$. With $\pi_i : \mathbb{Z}^n \to L_i$ the corresponding projections, we have $\sigma \circ \pi_i = \pi_i \circ \sigma$. It follows from linear algebra that

$$\Lambda_j = (L_1 \cap \Lambda_j) \oplus \cdots \oplus (L_k \cap \Lambda_j) \ ,$$

holds for $j = 1$ and $j = 2$. Hence, it is sufficient to prove the lemma for the case $\mathbb{Z}^n = L_1$.

If $n = 1$, then $\sigma = \pm 1$. Hence, (Λ_1, Λ_2) equals $(\mathbb{Z}, \mathbf{0})$, or if $p = 2$ possibly $(\mathbf{0}, \mathbb{Z})$. In particular, they are of the prescribed forms.

Suppose now that $n = p$. Additionally assume that σ is just a permutation without sign flips, which is automatic when p is odd. Let

$$B_1 = \mathbb{Z}\mathbf{z} \quad \text{and} \quad B_2 = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{z}^\mathsf{T}\mathbf{x} = 0\},$$

where $\mathbf{z}$ is the all-1 vector in $\mathbb{Z}^n$. It is easy to verify that $B_1 \subseteq \Lambda_1$ and $B_2 \subseteq \Lambda_2$. Since $\mathbb{Q}B_1 + \mathbb{Q}B_2 = \mathbb{Q}^n$, we have by linear algebra that

$\Lambda_j = (\mathbb{Q}B_j) \cap \mathbb{Z}^n = B_j$ $(j = 1, 2)$. Hence, $\Lambda_1 = \mathbb{Z}\mathbf{z} \cong \sqrt{p} \cdot \mathbb{Z}$ and $\Lambda_2 = A_{p-1}$.

Suppose now that $n = 2$ and σ is not a permutation. Then $-\sigma$ is a permutation, otherwise $\sigma^2 \neq 1$. Compared to σ, this results in interchanging the corresponding lattices Λ_1 and Λ_2. Since $A_1 \cong \sqrt{2} \cdot \mathbb{Z}$, we are done. □

Proposition 6.7. *There exists a polynomial-time algorithm that, given a Hermitian form* $\mathbf{Q}$ *of* R^2 *and* $\sigma \in \mathrm{O}(\mathrm{Rot}(\mathbf{Q})) \setminus \mathcal{G}_{\mathbf{Q}}$*, computes a nonzero sublattice* Λ *of* $\mathrm{Rot}(\mathbf{Q})$ *of rank at most* n *such that* $\lambda_1(\Lambda) \leq \sqrt{2}$*.*

Proof. We may compute in polynomial time, e.g. using the characteristic polynomial of σ, the multiplicative order of σ. In particular, we may test whether $-1 \in \langle\sigma\rangle$.

Assume first that this is not the case. Then by replacing σ by some power, we may assume σ has prime order p and $\sigma \neq \pm 1$. We may now use Lemma 6.6 with $\sigma \in \mathrm{O}\,\mathbb{Z}^{2n}$, which allows us to obtain the lattices Λ_1 and Λ_2 from this lemma using linear algebra. Let Λ to be the one of minimal rank. Since $\mathrm{rk}(\Lambda_1) + \mathrm{rk}(\Lambda_2) = 2n$, we have $\mathrm{rk}\,\Lambda \leq n$. It remains to show that $\lambda_1(\Lambda_1), \lambda_1(\Lambda_2) \leq \sqrt{2}$. For Λ_2, this immediately follows from Lemma 6.6 and $\lambda_1(A_{p-1}) = \sqrt{2}$. If $p = 2$, then the same follows for Λ_1. Suppose p is odd. Then $\Lambda_1 \cong \mathbb{Z}^a \times \sqrt{p} \cdot \mathbb{Z}^c$ and $\Lambda_2 \cong A_{p-1}^c$ for some a, c. As $a + pc = \mathrm{rk}(\Lambda_1) + \mathrm{rk}(\Lambda_2) = 2n$ is a power of 2 and p is odd, we conclude that $a > 0$. Hence, $\lambda_1(\Lambda_1) = 1$ and we are done.

Now consider the case for general σ. By [LJPW24], we may compute $G = \mathcal{G}_{\mathbf{Q}}$. It suffices to show that $-1 \notin \langle\tau\rangle$ for some $\tau \in G \cdot \sigma$, since we may then apply the above argument on τ in the rôle of σ. Let $\mathbf{x} \in R^2$ be a shortest vector. Then by Lemma 6.3 there exists some $\rho \in G$ such that $\rho(\mathbf{x}) = \sigma(\mathbf{x})$. Hence, $\tau = \rho^{-1} \cdot \sigma$ fixes $\mathbf{x}$, and $\tau \neq 1$ because otherwise $\sigma \notin G$. Since -1 fixes no shortest vectors, we conclude that $-1 \notin \langle\tau\rangle$. □

6.3 Heuristic hardness of SVP

In this section, we analyze the hardness of recovering a shortest vector from a lattice Λ appearing in Proposition 6.7. Specifically, we determine the minimal block size β such that BKZ-β finds a shortest vector in Λ, using standard heuristics in lattice reduction.

We use the methodology based on the "2016 estimates" which were phrased for lattices related to *Learning with Errors* [ADPS16], and verified experimentally [AGVW17, DDGR20, PV21]. Here, we apply this methodology to a lattice of the form as in Equation (6.1). These 2016 estimates have already been used to motivate the experimentally verified estimates that BKZ-β recovers the shortest vector in a rotation of $\mathbb{Z}^n$ when $\beta = n/2 + o(n)$ [DPPvW22a].

Definition 6.8 ([DvW22])**.** A rank-n lattice Λ is an *f-unusual-SVP* instance if we have

$$f \cdot \lambda_1(\Lambda) \leq \mathrm{GH}(n) \cdot \det(\Lambda)^{1/n}.$$

For example, $\mathbb{Z}^n$ and A_n are $\Omega(\sqrt{n})$-unusual-SVP instances, as n tends to infinity.

The hardness of recovering a shortest vector in an unusual-SVP lattice is mainly determined by the ratio $\mathrm{GH}(n)\det(\Lambda)^{1/n}/\lambda_1(\Lambda)$, rather than the UniqueSVP factor $\lambda_2(\Lambda)/\lambda_1(\Lambda)$ [AD21]. For such f-unusual-SVP instance with an unusually short vector $\mathbf{v}$, a *threshold phenomenon* occurs in the terminal block of a BKZ-β tour once the projection of $\mathbf{v}$ onto this block is much shorter than the expected first minimum if that terminal block had been a random lattice. Now when this $\mathbf{v}$ is part of the basis at position $\mathrm{rk}(\Lambda) - \beta + 1$, subsequent tours of BKZ-β will then move $\mathbf{v}$ to the front of the basis with steps of $\beta - 1$ per tour [PV21].

The lattice computed in Proposition 6.7 is of the form $\mathbb{Z}^a \times \sqrt{p}\mathbb{Z}^c$ (with $a > 0$) or $\mathbb{Z}^b \times A_{p-1}^c$, and thus is an instance of the $\Omega(\sqrt{k})$-unusual-SVP, where $k = \mathrm{rk}(\Lambda) \leq n$. In the following heuristic, let $\delta_\beta = \mathrm{GH}(\beta)^{1/(\beta-1)}$ be the *root Hermite factor*, and note $\delta_\beta \approx (\beta/(2\pi e))^{\frac{1}{2(\beta-1)}}$ for $\beta > 50$.

Heuristic 6.9. *Suppose $\Lambda \subseteq R^2$ is a nonzero rank-k lattice with $\lambda_1(\Lambda) \leq \sqrt{2}$. If $\beta \in \mathbb{Z}_{\geq 2}$ satisfies*

$$\sqrt{2\beta/k} \leq \delta_\beta^{2\beta-k-1}, \tag{6.2}$$

then BKZ-β will recover a shortest vector of Λ. In particular, this condition holds asymptotically for $\beta = k/2 + 1$.

Heuristic Justification. Because Λ can be identified with a sublattice of $\mathbb{Z}^{2n}$, by [Duc24, Lem. 2], its volume is at least 1. Note that the projection of a shortest vector of Λ onto the terminal block of a BKZ-β tour has

an expected norm of $\lambda_1(\Lambda) \cdot \sqrt{\beta/k} \leq \sqrt{2\beta/k}$. Moreover, if the terminal block during a BKZ-β tour would be a random block, its first minimum would be $\delta_\beta^{2\beta-k+1} \cdot \det(\Lambda)^{1/k} \geq \delta_\beta^{2\beta-k+1}$. Thus, if Equation (6.2) holds, then the [ADPS16] success condition

$$\lambda_1(\Lambda) \cdot \sqrt{\beta/k} \leq \delta_\beta^{2\beta-k+1} \cdot \det(\Lambda)^{1/k}$$

is satisfied, and we expect a threshold phenomenon to occur. For the last statement, note the left hand side of Equation (6.2) is equal to $\sqrt{1+2/k} \leq e^{1/k}$, and the right hand side of Equation (6.2) is at least $(k/(4\pi e))^{1/k}$, which is larger than $e^{1/k}$ once $k > 100$. △

Note that we use the *Geometric Series Assumption* (GSA) for this estimate, i.e., we assume the lengths of the Gram–Schmidt basis vectors follow a geometric series, and there are better simulators for these lengths [CN11, BSW18]. Still, the GSA gives a decent approximation. In addition, there is a more refined simulator that can take multiple shortest vectors of a lattice (e.g. n for $\mathbb{Z}^n$) into account [DDGR20], which expects a slightly lower required block size. However, because we are interested in the worst-case instance of Λ occurring in Proposition 6.7, such a refined simulator does not help in cases when there is a single shortest vector.

6

6.4 Group-theoretic heuristics

Our results are strong in that they impose no conditions on the automorphism, besides being nontrivial, but they 'only' halve the security parameter of Hawk. As [JWL+23] shows, under the stronger assumption that an attacker has access to an oracle giving automorphisms of a lattice, SVP can be (probabilistically) solved in polynomial time. If instead of having an oracle, the attacker generates a single uniformly random automorphism, then we can, as we will argue, heuristically break Hawk with high probability.

Suppose σ is uniformly sampled from $\mathrm{O}(\mathrm{Rot}(\mathbf{Q}))$, and let $g \in S_{2n}$ be the corresponding permutation on the shortest vectors of $\mathrm{Rot}(\mathbf{Q}) \cong \mathbb{Z}^{2n}$ modulo sign, i.e., on the coordinates of the lattice. Then, g is a uniformly random permutation. Each orbit $\Omega/\{\pm 1\}$ of g comes with a sign which is negative if the corresponding action of σ on Ω is transitive. Let c_k^s be the number of orbits of g of length $k \in [2n]$ and sign $s \in \{-, +\}$. One

can show, analogously to Lemma 6.6, that the characteristic polynomial of σ equals

$$\prod_{s\in\{-,+\}}\prod_{k=1}^{2n}(X^{2n}-s)^{c_k^s},$$

and that we have

$$\Lambda_1 = \ker(\sigma-1) \cong \prod_{k=1}^{2n}(\sqrt{k}\cdot\mathbb{Z})^{c_k^+}.$$

We will now argue that with high probability $\Lambda_1 \neq 0$ and $\operatorname{rk}\Lambda_1 = O(\log n)$ holds. In particular, smLIP reduces, given σ, to a comparatively trivial instance of SVP.

The probability that g has no fixed points is approximately $1/e$, so we may assume that g *has* a fixed point. Alternatively, one notes that if we multiply σ by the unique element of $\mathcal{G}_{\mathbf{Q}}$ such that the result g' fixes some preselected shortest vector $\mathbf{x}$, then g' is a uniformly random permutation among all permutations fixing $\mathbf{x}$. Multiplying σ by -1 if necessary, we may assume $c_1^+ > 0$. In particular, $\Lambda_1 \neq \{\mathbf{0}\}$ and $\lambda_1(\Lambda_1) = 1$. The expected number of orbits of g is

$$\sum_{k=1}^{n} 1/k \approx \log(n)\ .$$

Hence, the expected rank of Λ_1 is at most $\log(n)$.

On a less rigorous note, the group generated by $\mathcal{G}_{\mathbf{Q}}$ and σ will often be significantly larger than $|\mathcal{G}_{\mathbf{Q}}|\cdot|\langle\sigma\rangle|$. It is also not unlikely that it maps surjectively to S_{2n}. In this case it becomes possible to sample random-enough automorphisms, bringing us to the regime of [JWL+23].

6.5 Conclusion

Theorem 6.2 now easily follows from combining Proposition 6.7 and Heuristic 6.9. Namely, given a Hermitian form $\mathbf{Q}$ of R^2 and a nontrivial automorphism σ of $\textsc{Rot}(\mathbf{Q})$, Proposition 6.7 computes in polynomial time a basis for a lattice Λ of rank at most n, such that recovering a shortest vector of Λ allows solving smLIP for $\mathbf{Q}$. Then, Heuristic 6.9 shows, heuristically, that BKZ-β recovers a shortest vector of Λ when $\beta = \operatorname{rk}(\Lambda)/2+1 \leq n/2+1$.

Based on the results of Section 6.4, it is reasonable to suspect that Λ is of much lower rank in the average case. Indeed, sampling σ uniformly at random likely gives a lattice of rank at most $\log(n)$, for which directly solving SVP only takes polynomial time in n.

In practice, if one happens to find a nontrivial automorphism σ, it may be worthwhile to take a random composition of σ, X and $\omega_{\mathbf{Q}}$'s until finding a lattice of rank at most $\log(n)$, since $\langle \sigma, X, \omega_{\mathbf{Q}} \rangle = \mathrm{O}(\mathrm{Rot}(\mathbf{Q}))$ may happen.

Although we prove a worst-case result that, given a nontrivial automorphism, block size $n/2 + 1$ is heuristically sufficient, in practice we expect one could find a lattice of extremely low rank and subsequently solve smLIP and break Hawk. In this light, we believe that smLIP is *practically* as hard as finding a nontrivial automorphism, but theoretically not harder than finding a nontrivial automorphism and solving SVP on an easier lattice.

Bibliography

[AS64] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. U.S. Government Printing Office, 1964. URL: https://archive.org/details/AandS-mono600. Cited on pages 26, 123, and 124.

[AKSY22] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53. ACM Press, November 2022. DOI: 10.1145/3548606.3560650. Cited on page 148.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022. DOI: 10.4230/LIPIcs.ICALP.2022.8. Cited on page 150.

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in $\mathsf{NP} \cap \mathsf{coNP}$. In *45th FOCS*, pages 362–371. IEEE Computer Society Press, October 2004. DOI: 10.1109/FOCS.2004.35. Cited on pages 86, 87, 90, 92, 96, and 117.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108.

ACM Press, May 1996. DOI: 10.1145/237814.237838. Cited on page 39.

[Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th ACM STOC*, pages 10–19. ACM Press, May 1998. DOI: 10.1145/276698.276705. Cited on pages 14 and 41.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, Berlin, Heidelberg, July 1999. DOI: 10.1007/3-540-48523-6_1. Cited on pages 39 and 190.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd ACM STOC*, pages 601–610. ACM Press, July 2001. DOI: 10.1145/380752.380857. Cited on page 76.

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the third round of the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, July 2022. Update 1 (2022-09-26). DOI: 10.6028/NIST.IR.8413-upd1. Cited on pages 8, 10, and 146.

[ABC+26] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. Status report on the second round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, May 2026. DOI: 10.6028/NIST.IR.8610. Cited on page 11.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL.

In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Cham, April / May 2017. DOI: 10.1007/978-3-319-56614-6_4. Cited on pages 86 and 87.

[AD21] Martin R. Albrecht and Léo Ducas. *Lattice Attacks on NTRU and LWE: A History of Refinements*, pages 15–40. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. DOI: 10.1017/9781108854207.004. Cited on pages 79, 158, and 213.

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746. Springer, Cham, May 2019. DOI: 10.1007/978-3-030-17656-3_25. Cited on pages 81, 89, 111, 122, 134, and 251.

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 297–322. Springer, Cham, December 2017. DOI: 10.1007/978-3-319-70694-8_11. Cited on page 213.

[AS23] Martin R. Albrecht and Yixin Shen. Quantum augmented dual attack, 2023. arXiv:2205.13983. Cited on pages 87, 88, 116, and 117.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange — a new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium*, pages 327–343, Austin, TX, USA, August 2016. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim. Cited on pages 15, 86, 87, 95, 116, 213, and 214.

[APvW25] Bill Allombert, Alice Pellet-Mary, and Wessel P.J. van Woerden. Cryptanalysis of rank-2 module-LIP: A single real embedding is all it takes. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 184–212. Springer, Cham, May 2025. DOI: 10.1007/978-3-031-91124-8_7. Cited on page 206.

[AEN19] Yoshinori Aono, Thomas Espitau, and Phong Q. Nguyen. Random lattices: Theory and practice. Preprint, 2019. URL: https://espitau.github.io/bin/random_lattice.pdf. Cited on page 39.

[AGLL94] Derek Atkins, Michael Graff, Arjen K. Lenstra, and Paul C. Leyland. The magic words are squeamish ossifrage. In Josef Pieprzyk and Reihaneh Safavi-Naini, editors, *ASIACRYPT'94*, volume 917 of *LNCS*, pages 263–277. Springer, Berlin, Heidelberg, November / December 1994. DOI: 10.1007/BFb0000440. Cited on pages 5 and 6.

[ADM+24] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing - BUFF security of the additional NIST PQC signatures. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *PQCrypto 2024, Part II*, pages 301–335. Springer, Cham, June 2024. DOI: 10.1007/978-3-031-62746-0_13. Cited on page 175.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. DOI: 10.1007/BF02579403. Cited on pages 59, 60, 148, 154, and 175.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Cham, July 2014. DOI: 10.1007/978-3-319-08344-5_21. Cited on page 38.

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *LMS Journal of Computation and Mathematics*,

19(A):146–162, 2016. DOI: 10.1112/S1461157016000292. Cited on pages 80 and 202.

[BLR+18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018. DOI: 10.1007/s00145-017-9265-9. Cited on page 26.

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 369–404. Springer, Cham, December 2018. DOI: 10.1007/978-3-030-03326-2_13. Cited on pages 79, 80, 158, and 214.

[BN24] Henry Bambury and Phong Q. Nguyen. Improved provable reduction of NTRU and hypercubic lattices. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *PQCrypto 2024, Part I*, pages 343–370. Springer, Cham, June 2024. DOI: 10.1007/978-3-031-62743-9_12. Cited on page 206.

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Math. Ann.*, 296(1):625–635, 1993. DOI: 10.1007/BF01445125. Cited on page 57.

[BGMN05] Franck Barthe, Olivier Guédon, Shahar Mendelson, and Assaf Naor. A probabilistic approach to the geometry of the ℓ_p^n-ball. *Annals of Probability*, 33(2):480–513, 2005. DOI: 10.1214/009117904000000874. Cited on pages 127 and 134.

[BW25] Kaveh Bashiri and Andreas Wiemers. On the independence heuristic in the dual attack. *Journal of Mathematical Cryptology*, 19(1), 2025. DOI: 10.1515/jmc-2024-0028. Cited on page 122.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016. DOI: 10.1137/1.9781611974331.ch2. Cited on pages 80, 81, 86, 91, 105, and 121.

[BGPS23] Huck Bennett, Atul Ganju, Pura Peetathawatchai, and Noah Stephens-Davidowitz. Just how hard are rotations of $\mathbb{Z}^n$? algorithms and cryptography with the simplest lattice. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 252–281. Springer, Cham, April 2023. DOI: 10.1007/978-3-031-30589-4_9. Cited on pages 146, 162, and 206.

[Ber08] Daniel J. Bernstein. ChaCha, a variant of Salsa20. In Christophe De Cannière and Orr Dunkelman, editors, *State of the Art of Stream Ciphers*, SASC, pages 273–278, Lausanne, Switzerland, February 13–14, 2008. URL: https://cr.yp.to/chacha/chacha-20080128.pdf. Cited on page 170.

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *27th SODA*, pages 893–902. ACM-SIAM, January 2016. DOI: 10.1137/1.9781611974331.ch64. Cited on page 19.

[BM21] Tamar Lichter Blanks and Stephen D. Miller. Generating cryptographically-strong random lattice bases and recognizing rotations of $\mathbb{Z}^n$. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, pages 319–338. Springer, Cham, 2021. DOI: 10.1007/978-3-030-81293-5_17. Cited on page 206.

[Bli29] Hans Frederick Blichfeldt. The minimum value of quadratic forms, and the closest packing of spheres. *Math. Ann.*, 101:605–608, 1929. DOI: 10.1007/BF01454863. Cited on page 31.

[Bli35] Hans Frederick Blichfeldt. The minimum values of positive quadratic forms in six, seven and eight variables. *Mathematische Zeitschrift*, 39:1–15, 1935. DOI: 10.1007/BF01201341. Cited on page 31.

[BBD+23] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. Technical report, National Institute of Standards and Technology, 2023. URL: https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. Cited on pages 21 and 175.

[BBD+24] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. Technical report, National Institute of Standards and Technology, 2024. URL: https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures. Cited on pages 175 and 207.

[BBD+26] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. Technical report, National Institute of Standards and Technology, 2026. URL: https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures. Cited on pages xvi, 21, 145, and 208.

[BM18] Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *PQCrypto 2018*, pages 25–46. Springer, Cham, 2018. DOI: 10.1007/978-3-319-79063-3_2. Cited on page 89.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. DOI: 10.1145/2488608.2488680. Cited on page 57.

[CGS14] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: a cautionary tale. ETSI 2nd Quantum-Safe Crypto Workshop, October 2014. URL: https://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf. Cited on page 19.

[CDMT22] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 477–507. Springer, Cham, December 2022. DOI: 10.1007/978-3-031-22972-5_17. Cited on page 89.

[CDMT24] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Reduction from sparse LPN to LPN, dual attack 3.0. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 286–315. Springer, Cham, May 2024. DOI: 10.1007/978-3-031-58754-2_11. Cited on page 122.

[CST22] Kevin Carrier, Yixin Shen, and Jean-Pierre Tillich. Faster dual lattice attacks by using coding theory. Cryptology ePrint Archive, Report 2022/1750, 2022. Version dated 2022-12-20. URL: https://eprint.iacr.org/archive/2022/1750/20221220:184709. Cited on pages 87, 88, 116, and 117.

[Cas96] John William Scott Cassels. *An introduction to the geometry of numbers*. Classics in Mathematics. Springer, Berlin, Heidelberg, 1996. DOI: 10.1007/978-3-642-62035-5. Cited on pages 28 and 30.

[CSV12] Xiao-Wen Chang, Damien Stehlé, and Gilles Villard. Perturbation analysis of the QR factor R in the context of LLL lattice basis reduction. *Mathematics of Computation*, 81(279):1487–1511, 2012. DOI: 10.1090/S0025-5718-2012-02545-2. Cited on page 75.

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. PhD thesis, Université Paris Diderot, November 13, 2013. URL: https://archive.org/details/PhDChen13. Cited on pages 40 and 78.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, December 2011. DOI: 10.1007/978-3-642-25385-0_1. Cited on pages 79, 80, 100, 158, 159, and 214.

[CFS25] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. Reducing the number of qubits in quantum factoring. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part II*, volume 16001 of *LNCS*, pages 384–415. Springer, Cham, August 2025. DOI: 10.1007/978-3-032-01878-6_13. Cited on page 6.

[CME+25] Clémence Chevignard, Guilhem Mureau, Thomas Espitau, Alice Pellet-Mary, Heorhii Pliatsok, and Alexandre Wallet. A reduction from hawk to the principal ideal problem in a quaternion algebra. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 154–183. Springer, Cham, May 2025. DOI: 10.1007/978-3-031-91124-8_6. Cited on page 206.

[CK09] Henry Cohn and Abhinav Kumar. Optimality and uniqueness of the Leech lattice among lattices. *Annals of Mathematics*, 170(3):1003–1050, 2009. DOI: 10.4007/annals.2009.170.1003. Cited on page 31.

[CKM+17] Henry Cohn, Abhinav Kumar, Stephen D. Miller, Danylo Radchenko, and Maryna Viazovska. The sphere packing problem in dimension 24. *Annals of Mathematics*, 185(3):1017–1033, 2017. DOI: 10.4007/annals.2017.185.3.8. Cited on page 31.

[CS98] John H. Conway and Neil J.A. Sloane. *Sphere Packings, Lattices and Groups*, volume 3 of *Grundlehren der mathe-*

matischen Wissenschaften. Springer, New York, 1998. DOI: 10.1007/978-1-4757-6568-7. Cited on page 30.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Berlin, Heidelberg, August 2000. DOI: 10.1007/3-540-44598-6_14. Cited on page 10.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Berlin, Heidelberg, May 2016. DOI: 10.1007/978-3-662-49896-5_20. Cited on page 19.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 324–348. Springer, Cham, April / May 2017. DOI: 10.1007/978-3-319-56620-7_12. Cited on page 19.

[CDF+21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE Computer Society Press, May 2021. DOI: 10.1109/SP40001.2021.00093. Cited on page 175.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Cham, August 2020. DOI: 10.1007/978-3-030-56880-1_12. Cited on pages 80, 158, 159, 160, 162, 163, 164, 213, and 214.

[DADRT23] Thomas Debris-Alazard, Léo Ducas, Nicolas Resch, and Jean-Pierre Tillich. Smoothing codes and lattices: Sys-

tematic study and new bounds. *IEEE Transactions on Information Theory*, 69(9):6006–6027, 2023. DOI: 10.1109/TIT.2023.3276921. Cited on pages 88, 101, and 104.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145. Springer, Cham, April / May 2018. DOI: 10.1007/978-3-319-78381-9_5. Cited on page 110.

[Duc24] Léo Ducas. Provable lattice reduction of $\mathbb{Z}^n$ with blocksize n/2. *DCC*, 92(4):909–916, 2024. DOI: 10.1007/s10623-023-01320-7. Cited on pages 206, 208, 209, and 213.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40041-4_3. Cited on page 146.

[DN12a] Léo Ducas and Phong Q. Nguyen. Faster Gaussian lattice sampling using lazy floating-point arithmetic. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, December 2012. DOI: 10.1007/978-3-642-34961-4_26. Cited on page 146.

[DN12b] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 433–450. Springer, Berlin, Heidelberg, December 2012. DOI: 10.1007/978-3-642-34961-4_27. Cited on pages 156, 191, 203, and 206.

[DPPvW22a] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 65–94. Springer,

Cham, December 2022. DOI: 10.1007/978-3-031-22972-5_3. Cited on pages xv, 21, 145, 175, 206, 207, 209, and 213.

[DPPvW22b] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. Cryptology ePrint Archive, Report 2022/1155, 2022. URL: https://eprint.iacr.org/2022/1155. Cited on pages 164, 188, 189, and 200.

[DP16] Léo Ducas and Thomas Prest. Fast Fourier orthogonalization. In *2016 International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 191–198, Waterloo, ON, Canada, July 20–22, 2016. ACM Press, New York. DOI: 10.1145/2930889.2930923. Cited on pages 146, 152, 153, 174, and 251.

[DP23a] Léo Ducas and Ludo Pulles. Does the dual-sieve attack on learning with errors even work? Cryptology ePrint Archive, Report 2023/302, 2023. URL: https://eprint.iacr.org/2023/302. Cited on pages 95, 101, 110, 112, and 141.

[DP23b] Léo Ducas and Ludo N. Pulles. Accurate score prediction for dual-sieve attacks. Cryptology ePrint Archive, Report 2023/1850, 2023. URL: https://eprint.iacr.org/2023/1850. Cited on pages 20, 119, and 122.

[DP23c] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 37–69. Springer, Cham, August 2023. DOI: 10.1007/978-3-031-38548-3_2. Cited on pages xv, 20, 85, and 119.

[DP26] Léo Ducas and Ludo N. Pulles. Accurate score prediction for Dual-Sieve attacks. *Journal of Cryptology*, 39(8), 2026. DOI: 10.1007/s00145-025-09560-7. Cited on pages xv, 20, and 119.

[DPS25] Léo Ducas, Ludo N. Pulles, and Marc Stevens. Towards a modern LLL implementation. In Goichiro Hanaoka

and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part III*, volume 16247 of *LNCS*, pages 65–99. Springer, Singapore, December 2025. DOI: 10.1007/978-981-95-5099-9_3. Cited on pages xvi, 65, and 75.

[DSvW21] Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. Advanced lattice sieving on GPUs, with tensor cores. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 249–279. Springer, Cham, October 2021. DOI: 10.1007/978-3-030-77886-6_9. Cited on page 81.

[DvW22] Léo Ducas and Wessel P. J. van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 643–673. Springer, Cham, May / June 2022. DOI: 10.1007/978-3-031-07082-2_23. Cited on pages 146, 147, 150, 158, 159, 188, 206, 213, and 252.

[Dud14] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding, 2014. arXiv:1311.2540. Cited on page 168.

[DTGD15] Jarek Duda, Khalid Tahboub, Neeraj J. Gadgil, and Edward J. Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69, 2015. DOI: 10.1109/PCS.2015.7170048. Cited on page 168.

[DV90] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990. URL: https://infoscience.epfl.ch/record/59946. Cited on page 56.

[DHVvW20] Mathieu Dutour Sikirić, Anna Haensch, John Voight, and Wessel P.J. van Woerden. A canonical form for positive definite matrices. In Steven D. Galbraith, editor, *ANTS*

XIV, volume 4 of *Open Book Series*, pages 179–195, Auckland, New Zealand, June 29 – July 4, 2020. Mathematical Sciences Publishers. DOI: 10.2140/obs.2020.4.179. Cited on page 17.

[DvW25] Mathieu Dutour Sikirić and Wessel van Woerden. The lattice packing problem in dimension 9 by Voronoi's algorithm, 2025. arXiv:2508.20719. Cited on page 31.

[vEB81] Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, April 1981. URL: https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/mi8104c.html. Cited on page 41.

[vEH14] Tim van Erven and Peter Harremoës. Rényi divergence and Kullback–Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014. DOI: 10.1109/TIT.2014.2320500. Cited on page 26.

[EFG+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Cham, May / June 2022. DOI: 10.1007/978-3-031-07082-2_9. Cited on pages 149 and 150.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. On a dual/hybrid approach to small secret LWE - A dual/enumeration technique for learning with errors and application to security estimates of FHE schemes. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 440–462. Springer, Cham, December 2020. DOI: 10.1007/978-3-030-65277-7_20. Cited on pages 86, 87, 88, 91, 95, 109, and 116.

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter hash-and-sign lattice-based signa-

tures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 245–275. Springer, Cham, August 2022. DOI: 10.1007/978-3-031-15979-4_9. Cited on page 168.

[FH23] Serge Fehr and Yu-Hsuan Huang. On the quantum security of HAWK. In Thomas Johansson and Daniel Smith-Tone, editors, *PQCrypto 2023*, pages 405–416. Springer, Cham, August 2023. DOI: 10.1007/978-3-031-40003-2_15. Cited on pages 200 and 201.

[Fis13] Daniel Fischer. Fourier transform of the indicator of the unit ball. Mathematics Stack Exchange, 2013. Version: 2013-09-12. URL: https://math.stackexchange.com/a/492055. Cited on page 124.

[FPL24] The FPLLL development team. fplll, a lattice reduction library, Version: 5.5.0, November 2024. URL: https://github.com/fplll/fplll. Cited on page 159.

[FPL25] The FPLLL development team. fpylll, a Python wrapper for the fplll lattice reduction library, Version: 0.6.3, January 2025. URL: https://github.com/fplll/fpylll. Cited on pages 89 and 122.

[Gal62] Robert Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. DOI: 10.1109/TIT.1962.1057683. Cited on pages 86, 89, and 90.

[GHN06] Nicolas Gama, Nick Howgrave-Graham, and Phong Q. Nguyen. Symplectic lattice reduction and NTRU. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 233–253. Springer, Berlin, Heidelberg, May / June 2006. DOI: 10.1007/11761679_15. Cited on page 195.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Berlin, Heidelberg, April 2008. DOI: 10.1007/978-3-540-78967-3_3. Cited on pages 78 and 79.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, Berlin, Heidelberg, May / June 2010. DOI: 10.1007/978-3-642-13190-5_13. Cited on pages 76 and 79.

[Gar77] Martin Gardner. Mathematical games: A new kind of cipher that would take millions of years to break. *Scientific American*, 237(2):120–124, August 1977. DOI: 10.1038/scientificamerican0877-120. Cited on pages 3, 4, and 5.

[GS64] Israel Moiseevich Gel'fand [Ізраїль Мойсейович Гельфанд] and Georgi Evgen'evich Shilov [Георгий Евгеньевич Шилов]. *Generalized functions. Volume I: Properties and operations*. Academic Press, New York and London, 1964. Translated by Eugene Saletan. DOI: 10.1016/C2013-0-12223-4. Cited on page 126.

[GM18] Nicholas Genise and Daniele Micciancio. Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 174–203. Springer, Cham, April / May 2018. DOI: 10.1007/978-3-319-78381-9_7. Cited on page 146.

[vGP25] Daniël M.H. van Gent and Ludo N. Pulles. HAWK: Having automorphisms weakens key. *IACR Communications in Cryptology*, 2(2), 2025. DOI: 10.62056/a3qjp2w9p. Cited on pages xv, 21, and 205.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. DOI: 10.1145/1374376.1374407. Cited on pages 14, 58, 146, 176, 183, 190, 191, 206, and 252.

[GS02] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Lars R. Knudsen,

editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 299–320. Springer, Berlin, Heidelberg, April / May 2002. DOI: 10.1007/3-540-46035-7_20. Cited on pages 147, 184, and 206.

[Gid25] Craig Gidney. How to factor 2048 bit RSA integers with less than a million noisy qubits, 2025. arXiv:2505.15917. Cited on pages 6 and 7.

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021. DOI: 10.22331/q-2021-04-15-433. Cited on page 6.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of hecke points. *Forum Mathematicum*, 15(2):165–189, 2003. DOI: 10.1515/form.2003.009. Cited on pages 36 and 37.

[Gol66] Solomon W. Golomb. Run-length encodings (corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966. DOI: 10.1109/TIT.1966.1053907. Cited on page 167.

[GJ21] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 33–62. Springer, Cham, December 2021. DOI: 10.1007/978-3-030-92068-5_2. Cited on pages 20, 81, 85, 87, 88, 91, 92, 95, 96, 97, 99, 101, 102, 105, 108, 109, 110, 112, 116, 117, 120, and 141.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Berlin, Heidelberg, August 2011. DOI: 10.1007/978-3-642-22792-9_25. Cited on pages 76 and 79.

[Hås88] Johan Håstad. Dual vectors and lower bounds for the nearest lattice point problem. *Combinatorica*, 8(1):75–81,

1988. DOI: 10.1007/BF02122554. Cited on pages 86, 87, 90, and 96.

[HR14] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In Chandra Chekuri, editor, *25th SODA*, pages 391–404. ACM-SIAM, January 2014. DOI: 10.1137/1.9781611973402.29. Cited on page 17.

[Hea17] D.R. Heath-Brown. Iteration of quadratic polynomials over finite fields. *Mathematika*, 63(3):1041–1059, 2017. DOI: 10.1112/S0025579317000328. Cited on page 5.

[Her1850] Charles Hermite. Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objets de la théorie des nombres. (continuation). *J. Reine Angew. Math.*, 1850(40):279–315, 1850. DOI: 10.1515/crll.1850.40.279. Cited on page 80.

[HK17] Gottfried Herold and Elena Kirshanova. Improved algorithms for the approximate k-list problem in euclidean norm. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 16–40. Springer, Berlin, Heidelberg, March 2017. DOI: 10.1007/978-3-662-54365-8_2. Cited on page 202.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, second edition, August 1, 2002. DOI: 10.1137/1.9780898718027. Cited on pages 46 and 47.

[vH02] Mark van Hoeij. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, 95(2):167–189, 2002. DOI: 10.1006/jnth.2001.2763. Cited on page 71.

[HHP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Berlin, Heidelberg, April 2003. DOI: 10.1007/3-540-36563-X_9. Cited on page 147.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P.

Buhler, editor, *Algorithmic Number Theory*, ANTS '98, pages 267–288, Portland, OR, USA, June 21–25, 1998. Springer, Berlin, Heidelberg. DOI: 10.1007/BFb0054868. Cited on pages 17 and 19.

[How07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Berlin, Heidelberg, August 2007. DOI: 10.1007/978-3-540-74143-5_9. Cited on page 109.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Cited on page 146.

[AlJ01] A.Kh. Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 1–8. Springer, Berlin, Heidelberg, December 2001. DOI: 10.1007/3-540-45325-3_1. Cited on pages 89 and 90.

[Jaq25] Sam Jaques. Landscape of quantum computing – 2025 update, 2025. Accessed: 2026-01-22. URL: https://sam-jaques.appspot.com/quantum_landscape. Cited on page 7.

[JWL+23] Kaijie Jiang, Anyu Wang, Hengyi Luo, Guoxiao Liu, Yang Yu, and Xiaoyun Wang. Exploiting the symmetry of $\mathbb{Z}^n$: Randomization and the automorphism problem. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 167–200. Springer,

Singapore, December 2023. DOI: 10.1007/978-981-99-8730-6_6. Cited on pages 208, 209, 214, and 215.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206. ACM Press, April 1983. DOI: 10.1145/800061.808749. Cited on page 32.

[KKN+26] Alexander Karenin, Elena Kirshanova, Julian Nowakowski, Eamonn W. Postlethwaite, Ludo N. Pulles, Fernando Virdia, and Paul Vié. Cool + cruel = dual, and new benchmarks for sparse LWE. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part IV*, volume 16544 of *LNCS*, pages 304–333. Springer, Cham, May 2026. DOI: 10.1007/978-3-032-25327-9_11. Cited on page xvi.

[Kir16] Paul Kirchner. Algorithms on ideal over complex multiplication order. Cryptology ePrint Archive, Report 2016/220, 2016. URL: https://eprint.iacr.org/2016/220. Cited on page 184.

[KEF21] Paul Kirchner, Thomas Espitau, and Pierre-Alain Fouque. Towards faster polynomial-time lattice reduction. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 760–790, Virtual Event, August 2021. Springer, Cham. DOI: 10.1007/978-3-030-84245-1_26. Cited on pages 65 and 75.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *11th SODA*, pages 937–941. ACM-SIAM, January 2000. Cited on page 146.

[Klü10] Jürgen Klüners. The van Hoeij algorithm for factoring polynomials. In Nguyen and Vallée [NV10], pages 283–291. DOI: 10.1007/978-3-642-02295-1. Cited on page 71.

[Knu97] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison–Wesley, third edition, 1997. Cited on page 5.

[KZ1873] Aleksandr Nikolaevich Korkine [Александр Николаевич Коркин] and Egor Ivanovich Zolotarev [Егор Иванович Золотарёв]. Sur les formes quadratiques. *Math. Ann.*, 6(4):366–389, 1873. DOI: 10.1007/BF01442795. Cited on page 80.

[KS01a] Henrik Koy and Claus-Peter Schnorr. Segment LLL-reduction of lattice bases. In Joseph H. Silverman, editor, *Cryptography and Lattices*, pages 67–80. Springer, 2001. DOI: 10.1007/3-540-44670-2_7. Cited on page 75.

[KS01b] Henrik Koy and Claus-Peter Schnorr. Segment LLL-reduction with floating point orthogonalization. In Joseph H. Silverman, editor, *Cryptography and Lattices*, pages 81–96. Springer, Berlin, Heidelberg, 2001. DOI: 10.1007/3-540-44670-2_8. Cited on page 75.

[Laa21] Thijs Laarhoven. Approximate Voronoi cells for lattices, revisited. *Journal of Mathematical Cryptology*, 15(1):60–71, 2021. DOI: 10.1515/jmc-2020-0074. Cited on page 121.

[LW21] Thijs Laarhoven and Michael Walter. Dual lattice attacks for closest vector problems (with preprocessing). In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 478–502. Springer, Cham, May 2021. DOI: 10.1007/978-3-030-75539-3_20. Cited on pages 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 101, 102, 104, 115, 117, 120, and 126.

[LO83] J. C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. In *24th FOCS*, pages 1–10. IEEE Computer Society Press, November 1983. DOI: 10.1109/SFCS.1983.70. Cited on page 71.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015. DOI: 10.1007/s10623-014-9938-4. Cited on page 19.

[LM00] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statis-*

tics, 28(5):1302–1338, 2000. DOI: 10.1214/aos/1015957395. Cited on page 182.

[LL93] Arjen K. Lenstra and Hendrik W. Lenstra, Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, Berlin, Heidelberg, 1993. DOI: 10.1007/BFb0091534. Cited on pages 6 and 252.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982. DOI: 10.1007/BF01457454. Cited on pages 14, 59, 60, 70, 71, 72, 73, 159, 252, and 255.

[Len83] Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. DOI: 10.1287/moor.8.4.538. Cited on page 60.

[LS17] Hendrik W. Lenstra, Jr. and Alice Silverberg. Lattices with symmetry. *Journal of Cryptology*, 30(3):760–804, July 2017. DOI: 10.1007/s00145-016-9235-7. Cited on pages 147, 184, and 206.

[LS19] Hendrik W. Lenstra, Jr. and Alice Silverberg. Testing isomorphism of lattices over CM-Orders. *SIAM Journal on Computing*, 48(4):1300–1334, 2019. DOI: 10.1137/17M115390X. Cited on page 206.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 348–359. Springer, Berlin, Heidelberg, September 2006. DOI: 10.1007/11832072_24. Cited on page 86.

[LN25] Jianwei Li and Phong Q. Nguyen. A complete analysis of the BKZ lattice reduction algorithm. *Journal of Cryptology*, 38(1):12, January 2025. DOI: 10.1007/s00145-024-09527-0. Cited on pages 39 and 76.

[LJPW24] Hengyi Luo, Kaijie Jiang, Yanbin Pan, and Anyu Wang. Cryptanalysis of rank-2 module-LIP with symplectic au-

tomorphisms. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 359–385. Springer, Singapore, December 2024. DOI: 10.1007/978-981-96-0894-2_12. Cited on pages 195, 206, 207, 208, 210, and 212.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Cited on page 146.

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Berlin, Heidelberg, August 2009. DOI: 10.1007/978-3-642-03356-8_34. Cited on pages 44 and 252.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010. DOI: 10.1007/978-3-642-13190-5_1. Cited on page 49.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6), November 2013. Preliminary version in EUROCRYPT '10. DOI: 10.1145/2535925. Cited on page 19.

[MAT22] MATZOV. Report on the security of LWE: Improved dual lattice attack, April 2022. DOI: 10.5281/zenodo.6493704. Cited on pages 20, 85, 87, 88, 91, 92, 95, 96, 101, 102, 105, 108, 109, 110, 112, 115, 116, 117, 120, and 121.

[McH15] Nat McHugh. The magic words are squeamish ossifrage - factoring RSA-129 using CADO-NFS. Blogspot, March 2015. URL: https://natmchugh.blogspot.com/2015/03/

the-magic-words-are-squeamish-ossifrage.html. Cited on page 6.

[MT23] Charles Meyer-Hilfiger and Jean-Pierre Tillich. Rigorous foundations for dual attacks in coding theory. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 3–32. Springer, Cham, November / December 2023. DOI: 10.1007/978-3-031-48624-1_1. Cited on pages 89 and 141.

[Mic14] Daniele Micciancio. Lattice algorithms and applications, lecture 3: The dual lattice, 2014. URL: https://cseweb.ucsd.edu/classes/sp14/cse206A-a/lec3.pdf. Cited on page 55.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671 of *The Springer International Series in Engineering and Computer Science*. Springer, New York, first edition, March 31, 2002. Cited on page 33.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. DOI: 10.1137/S0097539705447360. Cited on pages 57, 58, and 156.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In Moses Charika, editor, *21st SODA*, pages 1468–1480. ACM-SIAM, January 2010. DOI: 10.1137/1.9781611973075.119. Cited on pages 42, 80, 86, 105, and 132.

[MW17] Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 455–485. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63715-0_16. Cited on page 27.

[Min1896] Hermann Minkowski. *Geometry of Numbers [Geometrie der Zahlen]*. B.G. Teubner, Leipzig, Germany, 1896. Cited on page 30.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985. DOI: 10.2307/2007970. Cited on page 177.

[MSV09] Ivan Morel, Damien Stehlé, and Gilles Villard. H-LLL: using Householder inside LLL. In *2009 International Symposium on Symbolic and Algebraic Computation*, ISSAC '09, pages 271–278, Seoul, Republic of Korea, July 28–31, 2009. ACM Press, New York. DOI: 10.1145/1576702.1576740. Cited on page 75.

[MdJvH+20] Moritz Müller, Jins de Jong, Maran van Heesch, Benno Overeinder, and Roland van Rijswijk-Deij. Retrofitting post-quantum cryptography in internet protocols: a case study of dnssec. *SIGCOMM Comput. Commun. Rev.*, 50(4):49–57, October 2020. DOI: 10.1145/3431832.3431838. Cited on page 146.

[MPPW24] Guilhem Mureau, Alice Pellet-Mary, Georgii Pliatsok, and Alexandre Wallet. Cryptanalysis of rank-2 module-LIP in totally real number fields. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 226–255. Springer, Cham, May 2024. DOI: 10.1007/978-3-031-58754-2_9. Cited on page 206.

[Nat16] National Institute of Standards and Technology. Announcing request for nominations for public-key post-quantum cryptographic algorithms, December 20, 2016. Accessed: 2026-02-20. URL: https://csrc.nist.gov/News/2016/Public-Key-Post-Quantum-Cryptographic-Algorithms. Cited on page 8.

[Nat22] National Institute of Standards and Technology. Call for additional digital signature schemes for the post-quantum cryptography standardization process, October 2022. Accessed: 2026-02-20. URL: https://csrc.nist.gov/Projects/

pqc-dig-sig/standardization/call-for-proposals. Cited on page 11.

[Nat24a] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication (FIPS) 203, August 13, 2024. DOI: 10.6028/NIST.FIPS.203. Cited on page 10.

[Nat24b] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication (FIPS) 204, August 13, 2024. DOI: 10.6028/NIST.FIPS.204. Cited on page 10.

[Nat24c] National Institute of Standards and Technology. Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standards Publication (FIPS) 205, August 13, 2024. DOI: 10.6028/NIST.FIPS.205. Cited on page 10.

[NS16] Arnold Neumaier and Damien Stehlé. Faster LLL-type reduction of lattice bases. In *2016 International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 373–380, Waterloo, ON, Canada, July 20–22, 2016. ACM Press, New York. DOI: 10.1145/2930889.2930917. Cited on page 75.

[NP33] Jerzy Neyman and Egon Sharpe Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933. DOI: 10.1098/rsta.1933.0009. Cited on page 95.

[NR09] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, 22(2):139–160, April 2009. DOI: 10.1007/s00145-008-9031-0. Cited on pages 14, 146, 147, 156, 191, 203, and 206.

[NS06] Phong Q. Nguyen and Damien Stehlé. LLL on the average. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, ANTS '06, pages 238–256, Berlin, Germany, July 23–28, 2006. Springer, Berlin, Heidelberg. DOI: 10.1007/11792086_18. Cited on page 78.

[NS09] Phong Q. Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009. Preliminary version in EUROCRYPT '05. DOI: 10.1137/070705702. Cited on page 75.

[NV10] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. ISC. Springer, 2010. DOI: 10.1007/978-3-642-02295-1. Cited on pages 14, 71, 236, and 249.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008. DOI: 10.1515/JMC.2008.009. Cited on pages 80, 81, 86, 105, and 202.

[OtR85] Andrew M. Odlyzko and Herman J.J. te Riele. Disproof of the Mertens conjecture. *J. Reine Angew. Math.*, 1985(357):138–160, 1985. DOI: 10.1515/crll.1985.357.138. Cited on page 71.

[Ove06] Raphael Overbeck. Statistical decoding revisited. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 06*, volume 4058 of *LNCS*, pages 283–294. Springer, Berlin, Heidelberg, July 2006. DOI: 10.1007/11780656_24. Cited on pages 89 and 90.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Berlin, Heidelberg, August 2010. DOI: 10.1007/978-3-642-14623-7_5. Cited on page 146.

[PS97] W. Plesken and B. Souvignier. Computing isometries of lattices. *J. Symbolic Computation*, 24(3–4):327–334, 1997. DOI: 10.1006/jsco.1996.0130. Cited on page 17.

[Poh87] Michael Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, 1987. DOI: 10.1016/S0747-7171(87)80061-5. Cited on pages 74 and 76.

[Pol75] John M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15:331–334, September 1975. DOI: 10.1007/BF01933667. Cited on page 5.

[Pom85] Carl Pomerance. The quadratic sieve factoring algorithm. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *EUROCRYPT'84*, volume 209 of *LNCS*, pages 169–182. Springer, Berlin, Heidelberg, April 1985. DOI: 10.1007/3-540-39757-4_17. Cited on page 6.

[Por23] Thomas Pornin. Improved key pair generation for falcon, BAT and hawk. Cryptology ePrint Archive, Report 2023/290, 2023. URL: https://eprint.iacr.org/2023/290. Cited on page 175.

[Por25] Thomas Pornin. Improved (again) key pair generation for falcon, BAT and hawk. Cryptology ePrint Archive, Report 2025/1239, 2025. URL: https://eprint.iacr.org/2025/1239. Cited on page 175.

[PP19] Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 504–533. Springer, Cham, April 2019. DOI: 10.1007/978-3-030-17259-6_17. Cited on pages 147, 151, 152, 153, and 172.

[PS05] Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 2005*, volume 3531 of *LNCS*, pages 138–150. Springer, Berlin, Heidelberg, June 2005. DOI: 10.1007/11496137_10. Cited on page 175.

[PV21] Eamonn W. Postlethwaite and Fernando Virdia. On the success probability of solving unique SVP via BKZ. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 68–98. Springer, Cham, May 2021. DOI: 10.1007/978-3-030-75245-3_4. Cited on pages 80, 158, and 213.

[PS24] Amaury Pouly and Yixin Shen. Provable dual attacks on learning with errors. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 256–285. Springer, Cham, May 2024. DOI: 10.1007/978-3-031-58754-2_10. Cited on pages 92 and 122.

[Pre15] Thomas Prest. *Gaussian sampling in lattice-based cryptography*. PhD thesis, Ecole normale supérieure-ENS PARIS, 2015. URL: https://tprest.github.io/pdf/pub/thesis-thomas-prest.pdf. Cited on page 174.

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 347–374. Springer, Cham, December 2017. DOI: 10.1007/978-3-319-70694-8_13. Cited on page 26.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Cited on pages 57, 145, 146, 149, 157, 165, 167, 168, 169, 170, 171, 173, 179, and 190.

[PT24] Ludo N. Pulles and Mehdi Tibouchi. Cryptanalysis of EagleSign. In Clemente Galdi and Duong Hieu Phan, editors, *SCN 24, Part II*, volume 14974 of *LNCS*, pages 165–186. Springer, Cham, September 2024. DOI: 10.1007/978-3-031-71073-5_8. Cited on page xvi.

[PV25] Ludo N. Pulles and Paul Vié. Accelerating the primal hybrid attack against sparse LWE using GPUs. Cryptology ePrint Archive, Paper 2025/1990, 2025. URL: https://eprint.iacr.org/2025/1990. Cited on page xvi.

[Rad68] Charles M. Rader. Discrete Fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968. DOI: 10.1109/PROC.1968.6477. Cited on page 56.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. DOI: 10.1145/1060590.1060603. Cited on page 86.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, September 2009. Preliminary version in STOC '05. DOI: 10.1145/1568318.1568324. Cited on pages 19, 44, 86, and 252.

[Reg25] Oded Regev. An efficient quantum factoring algorithm. *J. ACM*, 72(1):1–13, January 2025. DOI: 10.1145/3708471. Cited on page 6.

[Rén61] Alfréd Rényi. On measures of entropy and information. In Jerzy Neyman, editor, *4th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 547–561. University of California Press, 1961. URL: http://digicoll.lib.berkeley.edu/record/112906. Cited on page 26.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. DOI: 10.1145/359340.359342. Cited on pages 4, 10, and 252.

[Rog56] Claude A. Rogers. The number of lattice points in a set. *Proceedings of the London Mathematical Society*, s3-6(2):305–320, 1956. DOI: 10.1112/plms/s3-6.2.305. Cited on page 39.

[RH23] Keegan Ryan and Nadia Heninger. Fast practical lattice reduction through iterated compression. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 3–36. Springer, Cham, August 2023. DOI: 10.1007/978-3-031-38548-3_1. Cited on pages 63, 65, and 75.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987. DOI: 10.1016/0304-3975(87)90064-8. Cited on pages 14, 75, and 251.

[Sch03] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *STACS '03*, pages 145–156. Springer, Berlin, Heidelberg, February 27 – March 1, 2003. DOI: 10.1007/3-540-36494-3_14. Cited on pages 78 and 251.

[Sch06] Claus-Peter Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, 2006. DOI: 10.1016/j.ic.2005.04.004. Cited on page 75.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994. Preliminary version in FCT '91. DOI: 10.1007/BF01581144. Cited on pages 59 and 75.

[Sch91] Arnold Schönhage. Fast reduction and composition of binary quadratic forms. In *1991 International Symposium on Symbolic and Algebraic Computation*, ISSAC '91, pages 128–133, Bonn, West Germany, July 15–17, 1991. ACM Press, New York. DOI: 10.1145/120694.120711. Cited on page 70.

[SS71] Arnold Schönhage and Volker Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7(3):281–292, September 1971. DOI: 10.1007/BF02242355. Cited on page 70.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Cited on page 109.

[Sey93] Martin Seysen. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3):363–376, 1993. DOI: 10.1007/BF01202355. Cited on pages 64, 65, and 67.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. DOI: 10.1109/SFCS.1994.365700. Cited on page 6.

[Sie45] Carl Ludwig Siegel. A mean value theorem in geometry of numbers. *Annals of Mathematics*, 46(2):340–347, 1945. DOI: 10.2307/1969027. Cited on pages 36 and 39.

[Sie89] Carl Ludwig Siegel. *Lectures on the Geometry of Numbers*. Springer, Berlin, Heidelberg, 1989. DOI: 10.1007/978-3-662-08287-4. Cited on page 28.

[Sil21] Joseph Silverman. Origin of name "NTRU"?, June 9, 2021. URL: https://en.wikipedia.org/wiki/Talk:NTRU#Origin_of_name_%22NTRU%22? Cited on page 252.

[SLM+25] M.C. Smith, A.D. Leu, K. Miyanishi, M.F. Gely, and D.M. Lucas. Single-qubit gates with errors at the 10-7 level. *Physical Review Letters*, 134(23):230601, 2025. DOI: 10.1103/42w2-6ccy. Cited on page 6.

[Söd11] Anders Södergren. On the Poisson distribution of lengths of lattice vectors in a random lattice. *Mathematische Zeitschrift*, 269(3):945–954, 2011. DOI: 10.1007/s00209-010-0772-8. Cited on page 39.

[Ste10] Damien Stehlé. Floating-point LLL: Theoretical and practical aspects. In Nguyen and Vallée [NV10], pages 179–213. DOI: 10.1007/978-3-642-02295-1. Cited on page 75.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Berlin, Heidelberg, December 2009. DOI: 10.1007/978-3-642-10366-7_36. Cited on page 19.

[SW71] Elias M. Stein and Guido Weiss. *Introduction to Fourier Analysis on Euclidean Spaces*. Number 32 in Princeton Mathematical Series. Princeton University Press, 1971. URL: https://www.jstor.org/stable/j.ctt1bpm9w6. Cited on pages 28, 54, and 126.

[Sto96] Arne Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In *1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC '96, pages 267–274, Zürich, Switzerland, July 24–26, 1996. ACM Press, New York. DOI: 10.1145/236869.237084. Cited on page 98.

[SL96] Arne Storjohann and George Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In *1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC '96, pages 259–266, Zürich, Switzerland, July 24–26, 1996. ACM Press, New York. DOI: 10.1145/236869.237083. Cited on page 35.

[Szy03] Michael Szydlo. Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 433–448. Springer, Berlin, Heidelberg, May 2003. DOI: 10.1007/3-540-39200-9_27. Cited on pages 147 and 206.

[Via17] Maryna S. Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, 185(3):991–1015, 2017. DOI: 10.4007/annals.2017.185.3.7. Cited on page 31.

[Vor1908a] Georges Voronoï [Георгій Феодосійович Вороний]. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier Mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *J. Reine Angew. Math.*, 1908(133):97–102, 1908. DOI: 10.1515/crll.1908.133.97. Cited on page 31.

[Vor1908b] Georges Voronoï [Георгій Феодосійович Вороний]. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième Mémoire. Recherches sur les paralléllоèdres primitifs. *J. Reine Angew. Math.*, 1908(134):198–287, 1908. DOI: 10.1515/crll.1908.134.198. Cited on page 31.

[Wat1922] George Neville Watson. *A treatise on the theory of Bessel functions*. Cambridge University Press, 1922. URL: https://archive.org/details/treatiseontheory00watsuoft. Cited on page 123.

[Wes25] Bas Westerbaan. State of the post-quantum Internet in 2025. Cloudflare Blog, October 2025. Accessed: 2026-01-22. URL: https://blog.cloudflare.com/pq-2025/. Cited on page 8.

[WV24] Bas Westerbaan and Luke Valenta. A look at the latest post-quantum signature standardization candidates. Cloudflare Blog, November 7, 2024. Accessed: 2026-02-20. URL: https://blog.cloudflare.com/another-look-at-pq-signatures/. Cited on page 8.

[Yap92] Chee-Keng Yap. Fast unimodular reduction: Planar integer lattices (extended abstract). In *33rd FOCS*, pages 437–446. IEEE Computer Society Press, October 1992. DOI: 10.1109/SFCS.1992.267808. Cited on page 70.

List of Acronyms

AVX2 advanced vector extensions 2; 148, 169–171, 173, 176, 178, 180, 181

BDD bounded distance decoding; 14–16, 41–44, 86–92, 94–97, 101–108, 110–112, 114, 115, 117, 120–122, 125, 129, 133–138, 141, *see* CVP

BKZ block Korkine–Zolotarev basis reduction [Sch87], see Section 2.5.5; ix, 14, 15, 75, 76, 78–80, 92, 96, 100, 101, 116, 117, 158–163, 166, 207, 208, 212–215

CDF cumulative distribution function; 25, 114, 128, 130–133, 136, 137

CPU central processing unit; 6, 10, 111, 148, 171

CRQC cryptographically relevant quantum computer; 6–8, 10

CVP closest vector problem; 15, 16, 40, 41, 59, *see also* SVP

DFT discrete Fourier transform, see Section 2.3.2; 56, 97

DGS discrete Gaussian sampling, see Section 2.4; 146, 171, 178

EUF-CMA existential unforgeability under chosen message attack, see Section 2.7; 9, 10, 82

ffNP fast Fourier variant of NP [DP16]; 152, 174, 186, *see* NP

FFT fast Fourier transform; ix, 19, 56, 85–87, 89, 99, 116, 117, 122, 141, 148, 170–172, 176–178, 180–182, *see* DFT

G6K general sieve kernel [ADH+19], /ʒe.si.ka/; 81, 110, 111, 134

GH Gaussian heuristic, see Section 2.2.2; 88

GSA geometric series assumption [Sch03]; 79, 214

HKZ Hermite–Korkine–Zolotarev basis reduction [Sch87]; 80, *see* BKZ

HNF Hermite normal form, see Definition 2.31; 35, 36, 185

i.i.d. independent and identically distributed; 26, 81, 89, 127–129,

191, 202

KEM key encapsulation mechanism; 8, 12, 14

LIP Lattice Isomorphism Problem [DvW22]; 17–19, 21, 34, 35, 48, 52, 145–147, 158, 184, 188, 205, 206, 258, 260

LLL Lenstra–Lenstra–Lovász basis reduction [LLL82], see Section 2.5.4; ix, 14, 15, 71–76, 78, 79, 159, 188

LWE learning with errors [Reg09]; 19, 44, 86, 87, 95, 97, 99, 109, 159, *see* BDD

MATZOV Israel's Center of Encryption and Information Security, MATZOV; 85, 87, 91, 95, 101, 109

NFS number field sieve [LL93]; 6

NIST National Institute of Standards and Technology; xvi, 8, 10, 11, 20, 21, 85, 145, 146, 175, 207, 263

NP Babai's nearest-plane algorithm; 60, 91, 251, *see also* RO

NTRU number theorists 'r' us, according to [Sil21]; 14, 17, 19, 146, 149, 164, 206

NTT number theoretic transform; 148, 170–172, 176–178, 181–183, *see* DFT

omSVP one more (approximate) SVP; 21, 147, 156, 183, 192–194, 196, 197, 202, 203, 205–209, 258, 260, *see* SVP

PDF probability density function; 25, 129, 130

PSF Preimage Sampleable Function [GPV08], see Section 5.6.2; 183, 190, 192

RAM random access memory; 169–171, 175

RHF root Hermite factor, see Definition 2.108; 77, 78

RO rounding-off algorithm; 60, *see also* NP

ROM random oracle model; xiv, 192, 193, 201

RSA Rivest–Shamir–Adleman [RSA78]; 4–8, 10

SF survival function; 25, 113, 139, 140, *see* CDF

smLIP search module LIP; 21, 52, 183–185, 188, 189, 205–209, 211, 215, 216, 258, 260, *see* LIP

SNF Smith normal form, see Definition 2.35; 36, 98

SUF-CMA strong existential unforgeability under chosen message attack; xiv, 82, 190, 192, 193, 200, 201, 206, 209, *see* EUF-CMA

SVP shortest vector problem, see Section 2.2.3; 14, 17, 40–42, 44, 75, 81, 147, 214, 215

UniqueSVP unique shortest vector problem [LM09]; 44, 213, *see* SVP

WHT Walsh–Hadamard transform; 56, 111, *see* DFT

List of Symbols

$\lvert\cdot\rvert$	absolute value, *or* cardinality of a set
$\lVert\cdot\rVert$	Euclidean norm, or ℓ^2 norm
$\lVert\cdot\rVert_{\mathrm{F}}$	Frobenius norm of a matrix, i.e., ℓ^2-norm of all its entries
$[n]$	the set $\{1, 2, \ldots, n\}$ given $n \in \mathbb{N}$
$[\![\cdot]\!]$	Iverson bracket, i.e., 1 if its content is true, and 0 otherwise
$\lceil\cdot\rfloor$	rounding to nearest integer, $\mathbb{R} \to \mathbb{Z}$, such that $x - \lceil x \rfloor \in (-1/2, 1/2]$ holds for all $x \in \mathbb{R}$
$\mathbf{1}_{(\cdot)}$	indicator function on a set
$\widehat{(\cdot)}$	Fourier transform of a function, *or* the dual of a group (Definition 2.70)
$(\cdot)^\vee$	dual of a lattice (Definition 2.71), *or* dual of a basis
${}^\vee(\cdot)$	reversed dual basis related to some primal basis
$(\cdot)^\intercal$	transpose of a matrix
$(\cdot)^\star$	complex conjugation in $\mathbb{K}$, *or* Hermitian adjoint of a matrix
$(\cdot)^*$	Gram–Schmidt orthogonalisation of a basis
$\cdot \,\|\, \cdot$	concatenation of two strings
$\langle\cdot, \cdot\rangle$	dot product of two vectors

Greek

Γ	gamma function, $\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t) \mathrm{d}t$
$\eta_\varepsilon(\Lambda)$	smoothing parameter
$\xi(\alpha; x)$	Scaled multiple of Bessel function , *see* J_α
σ	canonical embeddings of $\mathbb{K}$ into $\mathbb{C}$, *or* parameter of (discrete) Gaussian distribution
μ_{ij}	Gram–Schmidt coefficient , *see* $(\cdot)^*$
$\mu_{\mathbb{K}}$	roots of unity of a number field $\mathbb{K}$, *see* $\mathbb{K}$

ρ_σ	Gaussian mass with parameter σ
$\rho_{\mathbf{Q},\sigma}$	Gaussian mass with respect to quadratic form $\mathbf{Q}$, *see* ρ_σ

Alphabetic

$\mathbb{C}, \mathbb{N}, \mathbb{Q}, \mathbb{R}, \mathbb{Z}$	sets of complex, natural, rational and real numbers, and integers, respectively. Natural numbers start from 0
$\lg, \ln, \log$	logarithm base $2, e$ and 10, respectively
$\arg\max_{x \in S} f(x)$	argument $x \in S$ maximising f, i.e., $f(x) = \max f(S)$
$\mathcal{B}^n$	n-dimensional ball
DFT_G	discrete Fourier transform using group G , *see* DFT
$\mathrm{diag}(\mathbf{d})$	diagonal matrix given by $\begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$
$\mathcal{D}_{\Lambda+\mathbf{c},\sigma}$	discrete Gaussian on $\Lambda + \mathbf{c}$ with parameter σ
$\mathcal{D}_{\mathbf{Q},\sigma}$	discrete Gaussian with respect to a quadratic form $\mathbf{Q}$, *see* $\mathcal{D}_{\Lambda+\mathbf{c},\sigma}$
$\widetilde{\mathcal{D}}_{\mathbf{Q},\sigma}[\mathbf{h}]$	discrete Gaussian on the coset $R^2 + \mathbf{h}$ with respect to a quadratic form $\mathbf{Q}$, *see* $\mathcal{D}_{\mathbf{Q},\sigma}$
erf	error function
erfc	complementary error function, i.e., $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$
ev_x	evaluation at x, i.e., $\mathrm{ev}_x(f) = f(x)$
$\mathbb{E}$	expectation value
$\mathbb{F}_q$	finite field of q elements
$\mathrm{GH}(n)$	Gaussian heuristic, i.e., $r \in \mathbb{R}_{>0}$ satisfying $\mathrm{Vol}_n(r\mathcal{B}^n) = 1$
$\mathrm{GL}_n(R)$	general linear group, $\{\mathbf{M} \in R^{n\times n} : \det \mathbf{M} \in R^*\}$
$\mathcal{G}$	Group of trivial automorphisms of $\mathbf{I}_2(R)$
$\mathcal{G}_\mathbf{Q}$	Group of trivial automorphisms of $\mathbf{Q}$
$\mathbf{I}_n$	identity matrix of order n
J_α	Bessel function (of the first kind) of order α
$j_{\alpha,n}$	nth root of the Bessel function of order α , *see* J_α
$\mathbb{K}$	algebraic number field, but in this thesis mostly the cyclotomic field $\mathbb{K} = \mathbb{Q}(\zeta_{2n})$
$\mathcal{L}(\mathbf{B})$	lattice generated by a basis $\mathbf{B}$
$\mathcal{L}_q(\mathbf{A})$	q-ary lattice generated by matrix $\mathbf{A}$
$\mathcal{L}_q^\perp(\mathbf{A})$	q-ary lattice generated by all vectors orthogonal to $\mathbf{A}$
N	algebraic (absolute) field norm

$\mathcal{N}(c, \sigma^2)$	Gaussian with centre c and standard deviation σ
$\mathcal{N}_n(\mathbf{c}, \sigma^2)$	n-dimensional normal distribution with centre $\mathbf{c}$ and standard deviation σ
$\mathrm{O}_n(R)$	orthogonal group, $\{\mathbf{M} \in R^{n\times n} : \mathbf{M}^\mathsf{T}\mathbf{M} = 1\}$
$\mathrm{O}(\Lambda)$	automorphism group of a lattice
$\mathrm{O}(\mathbf{Q})$	automorphism group of a quadratic form $\mathbf{Q}$
ord	order of a group element
$\mathcal{O}_\mathbb{K}$	ring of integers of $\mathbb{K}$, *see* $\mathbb{K}$
$\mathcal{P}(\mathbf{B})$	fundamental domain generated by basis $\mathbf{B}$
$\mathrm{Pot}(\mathbf{B})$	lattice potential of a basis, see [LLL82, Eq. (1.25)]
$\Pr[\,\cdot\,]$	probability of a certain event happening
R	ring of integers of $\mathbb{K}$, *see* $\mathcal{O}_\mathbb{K}$ & $\mathbb{K}$
rk	rank of lattice
$\mathrm{R}_a(D_1 \parallel D_2)$	Rényi divergence at order a of distribution D_1 from D_2
$\textsc{Rot}(\mathbf{x})$	coefficient embedding of all rotations, $R \to \mathbb{Z}^{n\times n}$, *see* $\textsc{Vec}(\mathbf{x})$
$\mathrm{SL}_n(R)$	special linear group, $\{\mathbf{M} \in R^{n\times n} : \det \mathbf{M} = 1\}$
$\mathrm{sl}(\mathbf{B})$	slope of a lattice basis
$\mathrm{span}(\,\cdot\,)$	$\mathbb{R}$-linear span
$\mathcal{S}^1$	unit circle (subset of $\mathbb{C}$)
$\mathcal{S}^{n-1}$	sphere of dimension $n-1$, which is a subset of $\mathbb{R}^n$
Supp	support (of a distribution)
Tr	algebraic (absolute) field trace
$\mathrm{U}(\,\cdot\,)$	uniform distribution on a given set
$\mathbb{V}$	variance, i.e., $\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$
$\textsc{Vec}(\mathbf{x})$	coefficient embedding $R \to \mathbb{Z}^n$
Vol_k	volume of k-dimensional object
$\mathcal{V}(\Lambda)$	Voronoi cell of lattice Λ

Samenvatting

Dit proefschrift getiteld “Rooster Cryptografie: Isomorfismes & Duale Aanvallen” gaat over roosters. Veel van de cryptografie die nu gebruikt wordt, kan in de toekomst met (grote) quantumcomputers worden gekraakt! Voor roosters bestaan er diverse berekeningsproblemen, die makkelijk zijn in 2 of 3 dimensies, maar moeilijk zijn in hoge dimensies. Deze problemen zijn, voor zover bekend, moeilijk voor computers *én* quantumcomputers. Daarom kunnen we met roosters *post-quantumcryptografie* bouwen, wat veilig blijft zelfs als er grote quantumcomputers zijn.

Cryptografie heeft als doel om communicatie tussen twee of meer partijen veilig te laten verlopen. Twee belangrijke cryptografische protocollen zijn *versleuteling* en *handtekeningen*. Versleuteling maakt de inhoud van berichten onbegrijpbaar voor afluisteraars. Door een handtekening valt het te controleren of iemand echt een bepaald bericht heeft gestuurd. In een handtekening protocol bezitten beide partijen verschillende sleutels, vergelijkbaar met de echte wereld: één partij schrijft handtekeningen met een bijzondere pen, en de ander weet de unieke kleursamenstelling van de inkt uit de pen. Versleuteling gaat vergelijkbaar: iedereen kan een bericht versleutelen met een *openbare sleutel*, maar alleen degene met de *geheime sleutel* kan het ontcijferen.

Roosters kunnen worden gebruikt om zo’n openbaar/geheim sleutelpaar te maken. De geheime sleutel is een beschrijving van het rooster, die bestaat uit de kortste roosterpunten. Deze sleutel is geheim omdat het in hoge dimensie moeilijk is om überhaupt één redelijk kort roosterpunt te berekenen. De openbare sleutel is een willekeurige beschrijving van het rooster met lange roosterpunten. Het snelste algoritme om de geheime sleutel te geven die hoort bij een openbare sleutel, heeft tijd en geheugen nodig dat exponentieel groeit in de dimensie.

Deel II gaat over de *duale aanval*. Dit is een algoritme wat als invoer een roosterbeschrijving en een punt krijgt, en dan met behulp van het *duaal rooster* het roosterpunt vindt wat er het dichtste bij ligt. Elk duaal roosterpunt correspondeert met een golffunctie (karakter) die in elk roosterpunt maximaal is. Met deze golffunctie kun je bepalen of een punt dicht bij het rooster ligt. Door het rooster op te delen als een schaakbord in een zwart deelrooster en een witte nevenklasse, kun je bepalen of het zwarte deelrooster dichter bij het punt ligt dan het witte. Als je deze methode vaak genoeg herhaalt, weet je uiteindelijk welk roosterpunt het dichtste bij het punt ligt.

Hoofdstuk 3 toont aan dat een in de literatuur gebruikte aanname voor het analyseren van de zogenaamde *duale-zeefaanval* fout is, want de aanname is tegenstrijdig in verschillende theoretische situaties, en uit experimenten blijkt deze verkeerde voorspelling te geven. Hoofdstuk 4 bevat een alternatieve aanname voor de duale zeefaanval. Deze aanname leidt tot nauwkeurige voorspellingen voor verschillende belangrijke kansverdelingen van punten.

Deel III gaat over het *rooster isomorfisme probleem* (LIP). Speciale roosters met goede eigenschappen zijn handig om in *handtekeningenschema's* te gebruiken, want deze maken het schema zeer efficiënt. Omdat elke cryptoloog goede beschrijvingen van deze roosters kent, moet het rooster op een andere manier verborgen worden. Met behulp van LIP kunnen we het rooster verbergen, door het in een willekeurige richting te draaien. De draaiing is nu de geheime sleutel.

In Hoofdstuk 5 bouwen we het handtekeningenschema Hawk op basis van LIP. Om de geheime sleutel van Hawk te krijgen, moet je smLIP, een moduulvariant van LIP oplossen voor het simpele rooster $\mathbb{Z}^n$, het rooster van geheeltallige punten. Veiligheid van Hawk is gebaseerd op het nieuwe *nog een korte vector probleem* (omSVP). In Hoofdstuk 6 bestuderen we *automorfismes* van het moduulrooster: een draaiing van de ruimte die het rooster op zichzelf afbeeldt. Stelling 6.2 zegt dat met behulp van een niet-triviaal automorfisme, het veel makkelijker is om korte vectoren te vinden, en om smLIP op te lossen. Een automorfisme stuurt een kort roosterpunt naar een (mogelijk ander) kort roosterpunt, en kan hierdoor omSVP oplossen. Als smLIP een lastig probleem is, dan zal het moeilijk zijn om een niet-triviaal automorfisme te vinden, en zal het moeilijk zijn om omSVP op te lossen door het vinden van een niet-triviaal automorfisme.

Summary

This thesis is about lattices. Much of the cryptography that is being used today, can be broken by (large) quantum computers in the future! There are multiple computational problems involving lattices that are easy in dimension 2 or 3 but are hard in high dimensions. These problems are, as far as we know, difficult for *both* classical computers *and* quantum computers. Therefore, we can use lattices to build so-called *post-quantum cryptography*, which will remain secure even when large quantum computers are built.

A goal of cryptography is to make secure communication between two or more parties possible. There are two important cryptographic protocols: *encryption* and *signature schemes*. Encryption makes the content of messages incomprehensible for eavesdroppers. A signature scheme enables one to verify whether a message was really sent by some party. The two parties hold different keys in a signature scheme, just like in the real world: one party writes signatures with a special pen, while the other knows the precise colour composition of the ink in the pen. Encryption is similar: everyone can encrypt a message using a *public key*, while deciphering is only possible for one owning the *private key*.

Lattices can be used to generate such a public/private key pair. The private key is a description of the lattice, that consists of the shortest lattice points. This key is private because it is difficult in high dimensions, to even compute a somewhat short lattice point. The public key is an arbitrary description of the lattice, which usually consists of long lattice points. The fastest algorithm that upon input a public key, computes the private key associated to it, requires time and memory that grows exponentially as a function of the dimension.

Part II concerns the *dual attack*. This is an algorithm that, upon

input a description of a lattice and a point, outputs the lattice point that is the closest to the point while making use of the *dual lattice*. Every dual lattice point corresponds with a wave function (in fact, character) that is maximal at every lattice point. Using this wave function, one can decide whether a point is close to the lattice or not. By subdividing the lattice like a chess board into a black sublattice and a white coset, you can determine whether the black sublattice is closer to the point than the white coset. If this method is repeatedly sufficiently many times, one can determine which lattice point is closest to the point.

Chapter 3 shows that an assumption that was used in literature to analyse the so-called *dual-sieve attack* is incorrect, because it produces contradictions in multiple theoretic regimes, and experiments show that it leads to wrong predictions. Chapter 4 contains an alternative assumption for the dual-sieve attack. This assumption leads to accurate predictions for multiple important probability distributions of points.

Part III concerns the *lattice isomorphism problem* (LIP). Special lattices with good properties are convenient when building a *signature scheme*, because they make a signature scheme very efficient. Because every cryptology researcher knows a good description of such a lattice, the lattice has to be hidden differently. Using LIP we can hide the lattice by randomly rotating it. The specific rotation now becomes the private key.

Chapter 5 is dedicated to the construction of the signature scheme Hawk, which is based on LIP. To find the private key of Hawk, one needs to solve smLIP, a module variant of LIP for the simple lattice $\mathbb{Z}^n$, the lattice consisting of all integer vectors. Security of Hawk is based on the new *one more shortest vector problem* (omSVP). In Chapter 6 we study *automorphisms* of the module lattice: a rotation that maps the lattice to itself. Theorem 6.2 states that, using a nontrivial automorphism, it becomes much easier to find a short vector, and to solve smLIP. An automorphism sends a short lattice point to a (possibly different) short lattice point, and therefore may easily solve omSVP. If smLIP is a difficult computational problem, then it will be difficult to find a nontrivial automorphism, and thus, it will be difficult to solve omSVP by 'just' finding a nontrivial automorphism.

Acknowledgments

I would first like to thank my promotors, ***Prof.dr. Léo Ducas*** and ***Prof.dr. Ronald Cramer*** for their supervision during my PhD.

Léo, my first promotor and main advisor, introduced me to lattices with great enthusiasm. His focus on geometric intuition and the right formalism were valuable and inspirational to me. His push in the right direction, or pull from the wrong 'rabbit holes' were helpful in becoming a better researcher.

Ronald, my second promotor and group leader, is gratefully acknowledged for the opportunities that he provided directly and indirectly by bringing together such a wonderful group of people.

I am grateful to Prof.dr. ***Cong Ling***, Prof.dr. ***Phong Nguyen*** and dr. ***Thomas Debris***, members of my Doctorate Committee, for reading my thesis and providing feedback.

Moreover, I would like to thank all my co-authors: ***Wessel***, mijn academische grote broer, voor de fijne carpoolritjes vanuit Leiden en voor het uitstippelen van mijn academische pad; ***Eamonn*** for your humour and passionate story-telling, for enlightening me about LaTeX pedantry and the English language, and for your genuine interest in my well-being; ***Joppe***, ***Olivier***, ***Serge***, ***Yu-Hsuan***, ***Thomas Pornin*** and ***Thomas Prest*** for helping Hawk fly; ***Mehdi*** for the great remote collaboration; ***Damien***, for hosting me twice at CryptoLab in Lyon; ***Daan van G.*** voor het laten zien dat artikelen kort kunnen zijn; ***Marc*** voor alle programmeer tips en tricks; ***Paul*** for doing an internship with me; ***Alexander K.***, ***Elena***, ***Fernando*** and ***Julian*** for joining forces and helping with computations.

Paranymph, academic twin and fellow duckling ***Shane*** was the ideal office mate. It was a joy to jointly chair the activity committee. Your compassion and patience are admirable, and your advice in various

personal matters was valuable. I hope your ears did not go numb from hearing all my random fun facts.

Paranimf en grote speler ***Jelle***, ik dank je voor de vele lange avonden in Kafé België, waarin je zeer gepassioneerd praatte over de meest onnozele onderwerpen. De roadtrips door Tunesië en met ***Yaïr*** door Roemenië, langs holbewoners en beren, respectievelijk, waren origineel en onvergetelijk.

Thanks to all the crypto group members and colleagues at CWI, for making it such a warm and vibrant research environment. I got off to a flying start at CWI by joining the amazing trip to Pampus for CWI's 75 year anniversary in my second week, and joining the activity committee that same day. I am grateful to my fellow hardworking committee members ***Isa***, ***Ruben***, ***Hema***, ***Arjan***, ***Vlad***, ***Sanne***, ***Sebas***, ***Max***, ***Nikhil***, ***Shane***, ***Lynn***, ***Ila***, ***Henrik***, ***Randy***, ***Jelena***, ***Emil***, ***Savvina***, ***Irene***, ***Estéban***, ***Alexander B.***, for organising together so many events, such as workshops on making soap or solving Rubik's cubes, a trip to Texel, movie nights and tournaments. It is great to see the Monday tea tasting sessions being continued up to this day. Thanks to ***Simona*** and ***Maxime*** for smoothly running our group seminar. ***Aron*** and ***Eamonn***, we had some great evenings at het concertgebouw, and many *encores*. ***Sophie***, wat een goed idee om met collega's de Amstel Gold Race (AGR) te fietsen. Daarnaast gaat mijn dank uit naar alle vaste krachten: ***Emil*** (voor het tot leven brengen van de tot dood verklaarde server), ***Dick*** (voor het soepeltjes regelen van contracten, en deelnames aan AGR of BAPC), ***Irene***, ***Minnie***, ***Susanne***, ***Joan-Anne*** en ***Pascal***. I enjoyed playing foosball with ***Arjan***, ***Daan P.***, ***Davi***, ***Dyon***, ***Fran***, ***Isa***, ***Léo C.***, ***Lisa***, ***Llorenç***, ***Lorenzo***, ***Lynn***, ***Marten***, ***Max***, ***Nikhil***, ***Randy***, ***Ruben***, ***Sebas***, ***Sebastiaan***, ***Shane***, ***Simona***, ***Yanlin***, ***Yaroslav***, and more, to lower strain on the brain.

To all my ICPC friends, ***Pim***, ***Reinier***, ***Jorke***, ***Wouter*** and ***Yu-Hsuan***, it was a joy coding with you. Mijn waardering gaat uit naar ***Marcel*** voor de productieve CodeCup weekenden met culinaire meesterwerken, en ik dank ***Kim*** dat ik als begeleiding mee mocht naar de onvergetelijke IOI in Szeged. Bedankt oud-huisgenoten ***Edzard*** en ***Jelte*** voor de fietsrondjes, en ***Bas*** voor de potjes *Age of Empires II*.

Tot slot wil ik mijn vrienden, familie en vriendin bedanken voor al hun steun en toeverlaat tijdens mijn promotietraject. ***Puck***, dank je wel voor alle gekkigheid en vrolijkheid, zelfs als het druk werd, de afgelopen 3 jaar. Hiermee is het proefschrift compleet. Op naar Bordeaux!

Curriculum Vitæ

Ludo Niels Pulles was born in Wageningen (Gelderland) on 13 March 1997, and raised in Gennep (Limburg). He obtained his high school diploma from Stedelijk Gymnasium Nijmegen in 2015.

Subsequently, he went studying at Utrecht University, and obtained bachelor's degrees in computer science, mathematics and physics in 2018, after having written an honours bachelor thesis titled 'A theoretic background on Pollard's Rho algorithm.' After an interlude of studying in Stockholm for half a year, he continued to study mathematics at Universiteit Leiden, and obtained his master's degree in 2021. His master thesis 'Finite separable rings' was supervised by Prof.dr. H.W. Lenstra Jr.

In 2021, Ludo became a PhD candidate in the Cryptology Group at Centrum Wiskunde & Informatica (CWI) in Amsterdam, and was supervised by Prof.dr. Léo Ducas. Here, he was able to apply his knowledge in mathematics and algorithms to topics in cryptology. His main focus was the cryptanalysis of lattice-based schemes, including determining the concrete hardness of attacks on them. He is involved in the design of the signature scheme Hawk, which is part of NIST's standardisation process of additional digital signatures.

After his PhD, Ludo has moved to Bordeaux, where he is working as a postdoctoral researcher at Université de Bordeaux, with dr. Alice Pellet–Mary.