AMSI SUMMER RESEARCH SCHOLARSHIPS 2025–26

Get a taste for Research this Summer

Dynamic analysis of lattice basis reduction algorithms

Trill White

Supervised by Kelvin Li, Julien Ugon,

<Deakin University>

Jane Street®

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

1 Introduction

As long as humans have existed, there has been stories, secrets, and codes. Communication is vital, and therefore so is its security. We would like to think that information that is designed for someone goes to that person, and no one else. And, whilst it was once sufficient to write on paper with lemon and lightly bbq the page to read its secrets, todays' methods of communication require methods that guarantee *a bit more* security.

Especially as, during this time, we increasingly live in world where quantum computers are becoming more and more advanced. (for example, with Finland recently launched Europe's first 50-qubit superconducting quantum computer in March 2025 [17]), organisations like CISA, NSA, and NIST are spearheading the transition to Post-Quantum Cryptography (PQC) [15, 2]. This is largely due to a preventative measure against such attacks known as "harvest now, decrypt later" strategy which could be employed by threat actors [12, 14, 15, 16]. This involves collecting encrypted data with a long secrecy lifetime today, with the intent of decrypting it once cryptographically relevant quantum computers (CRQCs) become available.

Standards relying on Public-key systems, like RSA and ECC, currently rely on the difficulty of integer factorization and discrete logarithms [12, 13], and this is not quantum-resistant. While symmetric encryption (e.g., AES) is more resilient, as Grover's algorithm [12] can be mitigated by increasing key sizes, Shor's algorithm [13] could render public-key encryption obsolete. Such a breakthrough would compromise global financial transactions, digital signatures, and secure web traffic (HTTPS) [13, 12, 14].

This is why lattice-based cryptography is relevant. It turns out that **lattice-based cryptography** is a primary solution because the underlying mathematical problems, such as the Shortest Vector Problem (SVP) and Learning With Errors (LWE), are resistant to known quantum shortcuts like Shor's algorithm [12]. Further, there are no known quantum attacks for lattice-based cryptographic systems. FIPS 203, 204 and 205 are primarily based around lattice cryptography.

The motivation for many researchers, (and this report) is to discuss the challenges behind modern cryptanalysis with regards to **lattice cryptography**, which requires a process called **lattice basis reduction**, the methods that researchers are using to analyse the quality of the output, as well as future areas for improvement in the field.

1.1 Background and Motivation

This research report covers both the foundations of lattice-based cryptography and the methods used to assess its security. We first review the necessary lattice theory, then present traditional and state-of-the-art cryptanalysis methods, and describe recent improvements.

A central component in analysing the strength of lattice cryptosystems is done using a process known as **lattice basis reduction**. We begin with the classical example of lattice basis reduction, Gram Schmidt Orthogonalisation, and then move onto Lenstra–Lenstra–Lovász (LLL) algorithm, which provides a polynomial-time method for obtaining a reduced basis. Although LLL typically yields only moderately short vectors, it remains the workhorse for many heuristic and practical attacks due to its

efficiency and robustness.

We then move to stronger reduction algorithms such as Block Korkine–Zolotarev (BKZ) and its variants. BKZ generalises LLL by performing reduction on blocks of basis vectors, allowing it to produce much shorter vectors (and thus stronger cryptanalytic attacks) at the cost of higher running time [3, 21]. Modern BKZ variants incorporate sophisticated enumeration and sieving techniques, and understanding their behaviour is crucial for estimating the security of lattice-based schemes.

We then make use of dynamic, simulator-based analyses of reduction algorithms, following the frameworks developed by Hanrot et al [20], then Li and Nguyen [21], and collaborators [3]. These works provide refined models and empirical data for the behaviour of LLL, BKZ, and related algorithms in practice, including estimates of root-Hermite factors, running times, and success probabilities as functions of dimension and block size. By integrating these dynamic analyses, they obtained more realistic security estimates for lattice-based constructions than those derived from purely theoretical bounds.

1.2 Literature Review

Lattice basis reduction algorithms like LLL and BKZ play a central role in lattice-based cryptanalysis, as they enable us to find shorter and more structured vectors in high-dimensional lattices than other methods can. By progressively reducing a lattice basis, these algorithms provide insight into the true difficulty of problems such as Shortest Vector Problem and Closest Vector Problem, which underpin the security of many lattice-based cryptosystems [3].

The analysis of lattice-based cryptography has been ongoing for several decades. It was popularised in the 1982 with the work of several mathematicians, including Lenstra, Lenstra, and Lovász. They created the LLL algorithm and subsequently contributed to further developments in the area [3]. Stronger algorithms were developed, including the block Korkine–Zolotarev algorithm in the 1980s. This, in turn, led to further advances such as algorithms proposed in SE91, SH95, GHGKN06, GN08a, MW16, ALNS20, ABLR21, LW23, to name a few, and even new lattice frameworks [1] [1].

A significant advancement in the theoretical understanding of the BKZ algorithm was introduced by Hanrot in 2011 [20] . Their work moved beyond static bounds to a dynamic system model, which provides a more granular analysis of how the quality of a lattice basis B. This was followed by Li and Nguyen in 2020, whose work provided a more granular analysis [21]. Unlike previous models that often relied on the Geometric Series Assumption (GSA) as a static rule, Li and Nguyen studied the changing state of the basis at the end of each tour to determine how close the basis truly comes to the theoretical limit [21].

1.3 Report Scope

This research paper will discuss lattices, properties of the lattice structure, the way that they are being used to encrypt information, as well as the algorithms and processes that are used in lattice reduction.

[1] in Lattice cryptography, these are all the normal shorthands for the algorithms that have been developed post LLL

This report will then discuss several important algorithms as well as future work that can be done in the field to apply this dynamic analysis to newer variants.

2 Preliminary

2.1 Lattices

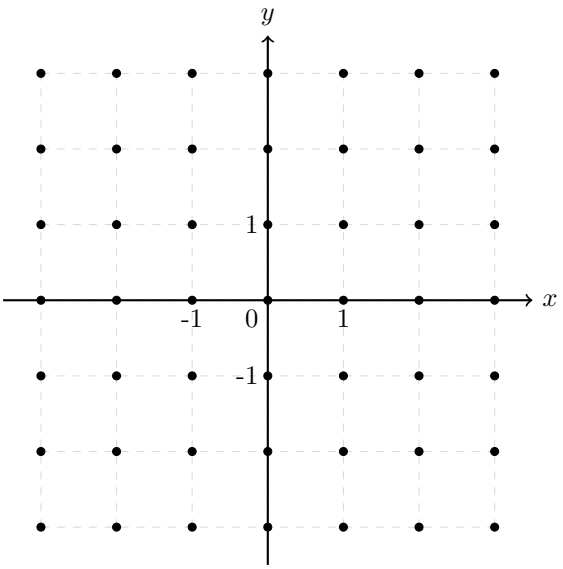
A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$, which simply means that under addition, they are a sub-group of all R^n, specifically integer based [3].

Or, given a set of k linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$, the lattice generated by this basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is defined as the set of all integer linear combinations:

$$\mathcal{L}(B) = \left\{ \sum_{i=1}^{k} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\} \tag{1}$$

When the number of basis vectors equals the dimension of the space ($k = n$), the lattice is referred to as a **full-rank** lattice.

It is important to note that while a lattice is unique, its basis is not; any lattice can be described by an infinite number of equivalent bases, which are related to one another by uni-modular matrices.



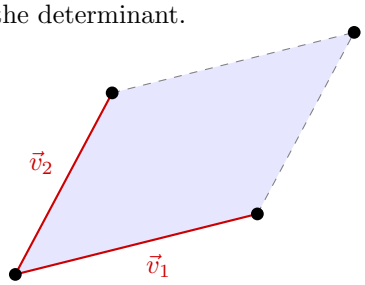
An important property of a lattice is often characterized by its **Fundamental Parallelepiped**, denoted as $\mathcal{P}(B)$, which represents the set of points $\mathcal{P}(B) = \{\sum_{i=1}^{k} \alpha_i \mathbf{b}_i : 0 \leq \alpha_i < 1\}$.

For full-rank lattices, the volume of this parallelepiped is an **invariant of the lattice known as the determinant**, calculated as $\det(\mathcal{L}) = |\det(B)|$. Being invariant, this volume stays the same for the lattice no matter what basis vectors are being used to represent it [3, 21].

This volume, which we can calculate easily for the lattice, is intrinsically linked to the Shortest Vector Problem (SVP) through **Minkowski's Theorem**, which provides an upper bound on the length of the shortest non-zero vector $\lambda_1(L)$ [3]:

$$\lambda_1(L) \leq 2 \left(\frac{\det(L)}{\text{vol}(B_n)} \right)^{1/n} \tag{2}$$

where $\text{vol}(B_n)$ denotes the volume of a unit ball in n dimensions. This is quite handy because as we will see it is not easy to calculate the exact shortest vector problem in lattices, but we will always be able to calculate the determinant.



The fundamental parallelepiped, $\mathcal{P} = \{t_1\vec{v}_1 + t_2\vec{v}_2 \mid 0 \leq t_i < 1\}$

In lattice cryptography theory, it is useful to distinguish between hard (bad) and easy (good) bases.

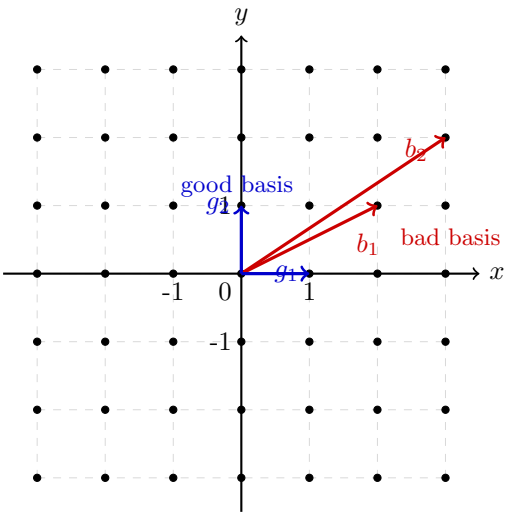
Definition 2.1: A bad basis

bad basis typically consists of long, highly correlated vectors that make solving the SVP or the Closest Vector Problem (CVP) computationally expensive.

Definition 2.2: A good basis

hereas, a good basis consists of vectors that are relatively short and nearly orthogonal.

Below is an example of such a **good** and **bad** full-rank basis vectors in a 2D space:



Lattice reduction algorithms, such as **LLL** and the **BKZ** (Block Korkine-Zolotarev) algorithm which we will discuss further, work by straightening and shortening the basis vectors. The objective is to transform a basis of long, skewed vectors into a reduced basis that more closely approximates an orthogonal set, thereby facilitating more efficient solutions to hard lattice problems.

One of the most relevant methods, and a great place to start, is the Gram Schmidt Orthogonalisation.

2.2 Gram Schmidt

Those familiar with Gram-Schmidt Orthogonalization (GSO) will recognise it as a process that transforms a basis into an orthogonal set. This is achieved by iteratively projecting each vector onto the orthogonal complement of the span of the preceding vectors, effectively straightening the basis one vector at a time [21].

More formally, the GSO process converts an arbitrary set of linearly independent vectors, the basis $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, into an **orthogonal basis** $\mathcal{B}^{\text{ortho}} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$.

The following outlines the process for converting a set of linearly independent vectors $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots\}$ into an **orthogonal basis** $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots\}$.

Let the first vector in the orthogonal basis, $\mathbf{u}_1$, be equal to the first original vector, $\mathbf{v}_1$.

$$\mathbf{u}_1 = \mathbf{v}_1$$

Since no previous orthogonal vectors exist, $\mathbf{u}_1$ is, by definition, orthogonal to an empty set.

Now consider the second vector, $\mathbf{v}_2$. If $\mathbf{v}_2$ is not orthogonal to $\mathbf{u}_1$, we must go through the steps of removing the component of $\mathbf{v}_2$ that lies in the direction of $\mathbf{u}_1$.

Now, we must find the **projection of $\mathbf{v}_2$ onto $\mathbf{u}_1$**. This is done by scaling the vector $\mathbf{u}_1$ by the factor of the dot product $\langle \mathbf{v}_2, \mathbf{u}_1 \rangle$ over the squared magnitude of $\mathbf{u}_1$.

The projection of vector $\mathbf{v}$ onto vector $\mathbf{u}$ is defined as:

$$\text{proj}_{\mathbf{u}}\mathbf{v} = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\|\mathbf{u}\|^2}\mathbf{u}$$

For our specific case:

$$\text{proj}_{\mathbf{u}_1}\mathbf{v}_2 = \frac{\langle \mathbf{v}_2, \mathbf{u}_1 \rangle}{\|\mathbf{u}_1\|^2}\mathbf{u}_1$$

This result, $\text{proj}_{\mathbf{u}_1}\mathbf{v}_2$, represents the component of $\mathbf{v}_2$ that is in the direction of $\mathbf{u}_1$.

2.2.1 Orthogonal Component

The new orthogonal vector, $\mathbf{u}_2$, is then found by taking the original vector $\mathbf{v}_2$ and subtracting the projection component we just calculated. This difference creates a vector that is perpendicular (orthogonal) to $\mathbf{u}_1$.

$$\mathbf{u}_2 = \mathbf{v}_2 - \text{proj}_{\mathbf{u}_1}\mathbf{v}_2$$

This new vector $\mathbf{u}_2$ is now guaranteed to be orthogonal to $\mathbf{u}_1$.

For the third vector, $\mathbf{v}_3$, we must ensure it is orthogonal to *both* $\mathbf{u}_1$ and $\mathbf{u}_2$, which means removing the projection of $\mathbf{v}_3$ onto the subspace (plane) spanned by both $\{\mathbf{u}_1, \mathbf{u}_2\}$.

The **k-th orthogonal vector $\mathbf{u}_k$** is calculated by:

$$\mathbf{u}_k = \mathbf{v}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j}\mathbf{v}_k$$

By continuing this along, we obtain an orthogonal basis $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$.

We also end up with what is called the GSO coefficients, like below $\mu_{k,j}$, :

$$\mu_{k,j} = \frac{\langle \mathbf{v}_k, \mathbf{u}_j \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle}$$

2.3 Early reduction methods

Earlier we discussed that orthogonalised vectors are preferred as basis for the lattice schemes we use for cryptography. Another requirement that is also beneficial, is that they need to be integer based.

In discrete lattice problems, the vectors must have **integer solutions**. This constraint means that an exact GSO is impossible; the resulting basis vectors are only an **approximation** of this orthogonal property that is so important to us. An early algorithm that did this was the size reduction algorithm, that converted a base to a more orthogonalised set of basis vectors using just Gram-Schmidt Orthogonalisation process:

Algorithm 1 A size-reduction algorithm

Compute all Gram-Schmidt coefficients $\mu_{i,j}$.

for $i = 2$ **to** d **do**

  for $j = i - 1$ **downto** 1 **do**

    $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,j} \rceil \, \mathbf{b}_j$ {Reduce the j-th coefficient}

    for $k = 1$ **to** j **do**

      $\mu_{i,k} \leftarrow \mu_{i,k} - \lfloor \mu_{i,j} \rceil \, \mu_{j,k}$ {Update coefficients for the new $\mathbf{b}_i$}

    end for

  end for

end for

We can see that due to rounding, that this is an approximation, and not the exact orthogonality that we were hoping for. Because of the way that the floor function changes the reduction on the $\mu_{i,k}$ vector, each of the vectors in this loop will only be approximately orthogonal.

2.3.1 Size-Reduced Condition

From earlier, we know that the projection coefficients must be rounded to the nearest integer. This shortening requirement, imposed below, ensures that the basis vectors are short relative to their orthogonal complements.

The condition is that the absolute value of the scalar coefficients $\mu_{k,j}$ must be less than or equal to $1/2$ for all $1 \leq j < k \leq n$:

$$|\mu_{k,j}| \leq \frac{1}{2}$$

Another methodology that is useful for lattice basis reduction is projections, and lifting [3].

2.3.2 π Projection Process and Lifting

Lifting is a transformation process on a vector.

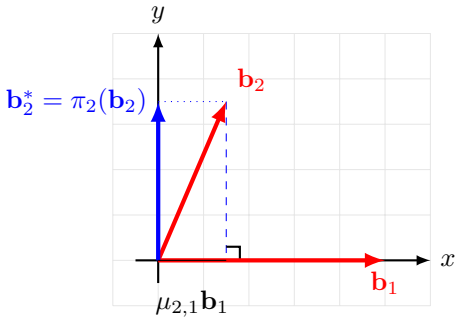
Suppose you have some basis vectors, b_0 ... b_n. You want to reduce to b_{n-1}.

You have to keep the same number of basis vectors, otherwise you have a completely different lattice. But, you can keep the information about a lattice basis vector to recreate it further down the track.

To re-create it, you need to define its overall structure, so that when you go to "lift" the basis vector again, you are adding the dimension back into the lattice, it is with a vector that reasonably approximates the vector that existed there beforehand.

The way we can reasonably approximate it is to define some criteria we would want any approximate to meet - for example, something like meeting the requirements on the length of the vector, or perhaps, replicating the same orthogonality. This is what we do when we design our "lift".

We use the size condition from section 2.3.1, which mandates a certain size for the vectors and therefore a length minimisation. By using this, we can re-create the original vector (or a vector that re-creates the original as much as the original would meet the build criteria).



Many algorithms relies heavily on the projection of basis vectors. In this visualization, we see the vector b_2 being projected onto the orthogonal complement of b_1. This process is fundamental to the underlying LLL algorithm, where the Gram-Schmidt vectors b_i^* determine the quality of the basis reduction.

2.3.3 Lovász Condition

The Lovász condition, named after Hungarian mathematician László Lovász [3], is a key component of the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm, which he co-developed with Arjen and Hendrik Lenstra in 1982. This condition constrains how "close" the Gram–Schmidt vectors of a lattice basis can be in length and alignment, ensuring that the algorithm produces a reduced basis with provably bounded quality in polynomial time. [2].

[2] The LLL algorithm and Lovász's condition have had far-reaching impact in computational number theory, cryptanalysis, and the design and analysis of lattice-based cryptographic schemes, as well as integer linear programming

It ensures that the orthogonal basis vectors $\mathbf{u}_k$ do not become too short too quickly, maintaining a degree of orthogonality throughout the basis. It prevents the basis from being too skewed by requiring:

$$\|\mathbf{u}_k\|^2 \geq \left(\delta - \mu_{k,k-1}^2\right) \|\mathbf{u}_{k-1}\|^2$$

The Lovász condition is designed to regulate the near-orthogonality of the basis, ensuring that the swapping process within the Lenstra–Lenstra–Lovász (LLL) algorithm is efficient and purposeful. By maintaining this condition, we ensure that the norms of the Gram-Schmidt vectors do not decrease too rapidly, which effectively bounds the total number of swaps required. If the difference between two basis vectors was allowed to be very large, we essentially allow for potentially more than one possible sorting order.

Contrary to the intuition that swaps only move from the bottom of the basis vectors to the top, swaps in LLL can trigger a ripple effect. When a swap occurs at index i, it can violate the size-reduction condition for previous indices, potentially requiring the algorithm to shift its focus back toward the start of the basis. However, the Lovász condition guarantees that this process does not continue indefinitely; it ensures that each swap significantly improves the potential of the basis, which is the mathematical mechanism that guarantees **polynomial runtime**.

The Lovász condition is formally expressed as:

$$\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*\|^2 \tag{3}$$

By applying the Pythagorean theorem to the orthogonal components, this is often written as:

$$\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2\|\mathbf{b}_i^*\|^2 \tag{4}$$

Essentially, this condition checks the projection of the $(i+1)$-th vector. If the condition is not met, the vectors $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are swapped. This prevents the basis from becoming too far from orthogonal and keeps the relative lengths of the Gram-Schmidt vectors $\mathbf{b}_i^*$ in check [21].

By controlling these swaps, we achieve a polynomial runtime—specifically $O(n^4 \log B)$, where B is the maximum length of the input vectors. While the LLL algorithm does not solve the Shortest Vector Problem (SVP) exactly (as it is NP-hard), the Lovász condition guarantees that the first vector $\mathbf{b}_1$ in the reduced basis is a strong approximation. Specifically, it guarantees:

$$\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \cdot \lambda_1(L) \tag{5}$$

where $\lambda_1(L)$ represents the length of the actual shortest vector in the lattice. With a standard choice of parameter $\delta = 3/4$, this provides the reliable, efficient performance that makes LLL a cornerstone of modern lattice-based cryptanalysis.

3 Hermites Algorithm

Hermites' algorithm bounds the upper value by what is called the Hermite coefficient, which is where the boundary of the number of points inside a lattice is considered further.

Algorithm 2 A simplified version of Hermite's first reduction algorithm [3].

Input: A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a d-rank lattice L.

Output: A reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

if $d = 1$ **then**

  output $\mathbf{b}_1$

end if

Apply recursively the algorithm to the basis $(\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_d))$ of the projected lattice $\pi_2(L)$. {π_2 projects onto the orthogonal complement of $\mathbf{b}_1$} Lift the vectors $(\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_d))$ into $\mathbf{b}_2, \ldots, \mathbf{b}_d \in L$ in such a way that they are size-reduced with respect to $\mathbf{b}_1$. {This involves the size-reduction from Algorithm 3}

if $\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$ **then**

  output $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$

end if

Swap $\mathbf{b}_1$ and $\mathbf{b}_2$ since $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$, and restart from the beginning. {This is the key reduction/swap step}

3.1 Proof Steps (By Induction)

The proof proceeds by induction on the dimension d. The key intermediate step, derived using the reduction condition (e.g., from the LLL algorithm) and the induction hypothesis on the projected lattice L', yields the following inequality:

The vectors are being swapped if necessary so that they are ordered, and this is important because we now reducing size as well as checking for orthogonality. So size is checked, and then orthogonality is ensured by making the GSO changes to each of the projections. This loops through, but unlike the changes in GSO, which didn't requite a further test, this algorithm doesn't do that.

This algorithm requires that it meets a condition so that it is less than the volume is bounded by the minimum of the volume expected in the space. The first is Minkowskis, and the second component is the approximation we are saying. So, in a way, I think it is saying, "well, we only want to be 4/3 as bad as the minimum bound factor on the minimum volume that can exist from this lattice"

Actually, the process is a bit more tricky than that. One of the steps involves de-projecting the lattice basis vectors down so that there is a "reduction". This comes from the π process, which essentially, take the lattice and take a basis, and project that onto the lattice basis that remain. This means that the essence of that basis vector is absorbed into the remaining basis vectors of the lattice.

The proof behind this in the book on the LLL Algorithm is inductive and based off dimension d. To link the d-dimensional result to the $(d-1)$-dimensional case, the volume lifting step is applied.

Let lattice L be the original d-dimensional lattice with volume $\mathrm{vol}(L) = \det(L)$.

Let projected lattice L' be the $(d-1)$-dimensional lattice spanned by the projections of the basis vectors $\{\mathbf{b}_2, \ldots, \mathbf{b}_d\}$ onto the orthogonal complement of $\mathbf{b}_1$. Its volume is $\mathrm{vol}(L')$.

Since the volume of the area is therefore going to be relational to the volume of the original lattice and the vector b_1 that spanned the space also, the volume relationship between the two is:

$$\text{vol}(L) = \|\mathbf{b}_1\| \cdot \text{vol}(L')$$

This is because it is 1 dimension less, and from earlier, we know that the total volume of the whole space is the product of the basis vectors spanning the space:

$$\text{vol}(L) = \prod_{i=1}^{n} |\mathbf{b}_i|$$

Later, we will substitute this in so that $\text{vol}(L') = \frac{\text{vol}(L)}{\|\mathbf{b}_1\|}$ into the induction hypothesis, transforming the bound from one involving the lower-dimensional $\text{vol}(L')$ to one involving the full-dimensional $\text{vol}(L)$. This matters because when we go to describe the boundary conditions on b_1, we need it to be in terms of the total dimensions of the lattice. **When we use the case on a lattice of a lower dimension, we are doing a proof by induction.**

4 Hermite's next algorithm

Hermite's next algorithm is shown below:

Algorithm 3 The Hermite Algorithm [3]

A basis $\mathbf{b} = (b_1, \ldots, b_d)$ of a lattice $L \subset \mathbb{R}^n$ with $d \geq 1$.

A first-reduced basis $(b_1, \ldots, b_d)$ such that $\|b_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \text{vol}(L)^{1/d}$.

if $d = 1$ **then**

  return b

end if

Compute the **Gram – Schmidt** orthogonalisation $(b_1^*, \ldots, b_d^*)$ of **b**.

Compute the **projected** basis $(\pi_2(b_2), \ldots, \pi_2(b_d))$ of the lattice $\pi_2(L)$.

Recursively apply Algorithm **??** to the projected basis:

$(\tilde{b}_2, \ldots, \tilde{b}_d) \leftarrow \text{Alg4}(\pi_2(b_2), \ldots, \pi_2(b_d))$

Lift $(\tilde{b}_2, \ldots, \tilde{b}_d)$ back to a basis $(b_2, \ldots, b_d)$ of L:

$b_i \leftarrow \tilde{b}_i + \mu_{i,1} b_1$, where $\mu_{i,1} = \frac{\langle b_i, b_1 \rangle}{\langle b_1, b_1 \rangle}$

SizeReduction: For $i = 2, \ldots, d$, replace b_i with $b_i - \lfloor \mu_{i,1} \rceil b_1$.

Swap Check:

if $\|b_2\| < \|b_1\|$ **then**

  Swap b_1 and b_2.

  Restart the algorithm with the new basis.

end if

return $(b_1, \ldots, b_d)$

This is different to the first Hermite algorithm, because this algorithm checks the size Bounds on all the previous vectors in the space, it not only shuffles $b2'$ and $b3'$ etc to be as small as they can be, it does

this in chronological order. So before, the first Hermite algorithm was only checking to see that this was done on the final two vectors, now, this is being done on all the basis vectors.

5 Modern Reduction Methods

5.1 LLL

The **Lenstra-Lenstra-Lovász (LLL)** algorithm 5.1 uses GSO iteratively to find a "good basis," using not only the GSO, but also the size reduced condition and the Lovász Condition:

Algorithm 4 The basic LLL algorithm [3]

Input: a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L.

Output: the basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is LLL-reduced with factor δ.

loop
  Size-reduce $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ (using Algorithm 3).
  if there exists an index j which does not satisfy Lovász' condition **then**
    swap $\mathbf{b}_j$ and $\mathbf{b}_{j+1}$
    continue to Step 1.
  else
    break
  end if
end loop

5.2 HKZ

A basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is Hermite-Korkine-Zolotarev (HKZ) reduced if it satisfies the following three conditions:

1. **Size-Reduction**: The Gram-Schmidt coefficients satisfy $|\mu_{i,j}| \leq 1/2$ for all $1 \leq j < i \leq n$.
2. **Shortest Vector**: The first vector $\mathbf{b}_1$ is a shortest non-zero vector in the lattice L, such that $\|\mathbf{b}_1\| = \lambda_1(L)$.
3. **Recursive Reduction**: The projections of the remaining vectors $\{\mathbf{b}_2, \ldots, \mathbf{b}_n\}$ onto the orthogonal complement of $\mathbf{b}_1$ are themselves HKZ-reduced.

5.3 BKZ

As preparation for the next section, two other techniques will be discussed.

5.3.1 SVP Solver

The core of the BKZ and variant algorithms is the Shortest Vector Problem (SVP) oracle, a tool first formalized for practical lattice reduction in early iterations of the algorithm (often referred to as BKZ

1.0) [21]. An exact solver is designed to identify the absolute shortest non-zero vector within a specific block of rank k. Libraries such as fplll [11] have implemented several tools for this purpose. *Exact solving remains the most computationally expensive component of lattice reduction, as the complexity of full enumeration grows exponentially with the block size [21] [10].*

In traditional lattice reduction, and in original BKZ, the algorithm runs until full convergence (no changes in a full tour), the oracle performs a what is known as a full enumeration, exhaustively searching the basis vectors to guarantee an exact SVP solution [11]. This approach ensures the algorithm runs until full convergence—the state where no further improvements occur within a complete tour.

To mitigate the exponential cost, BKZ 2.0 introduced extreme pruning [**?**]. Rather than searching the entire enumeration tree, the solver restricts its search to a high-probability subset of the lattice. By targeting a specific fraction of the tree where the shortest vector is statistically likely to reside, the oracle achieves significant speed-ups while maintaining a high success rate.

5.3.2 BKZ Algorithm

The BKZ algorithm improves upon the LLL algorithm by replacing the local swap with a full enumeration in the local projected lattice to find a shorter vector [7][21]. This is the computationally intensive part of the process because it utilises an SVP (Shortest Vector Problem) solver, which generally scales exponentially with the block size β. BKZ represents a strategic trade-off between the polynomial-time LLL algorithm and the exponential-time SVP solvers.

Once a shorter vector is found, it is inserted into the basis at the current index. We then apply LLL-reduction to remove the resulting linear dependency (reducing the set from $n+1$ vectors back to n). Similar to LLL, the BKZ algorithm terminates when no further non-trivial insertions can be made across a full pass of the basis. Shown below is the original Schnorr-Euchner algorithm:

Algorithm 5 BKZ Algorithm (Schnorr-Euchner) [7]

LLL-reduce B {Preprocessing}
$z \leftarrow 0, j \leftarrow 0$
while $z < n - 1$ **do**
  $j \leftarrow (j \bmod (n-1)) + 1$
  $k \leftarrow \min(j + \beta - 1, n)$
  Find shortest vector $\mathbf{v}$ in $\pi_j(\mathcal{L}(\mathbf{b}_j, \ldots, \mathbf{b}_k))$ via Enumeration
  if $\delta \|\mathbf{b}_j^*\|^2 > \|\mathbf{v}\|^2$ **then**
    $z \leftarrow 0$
    Insert $\mathbf{v}$ into basis at index j and LLL-reduce locally
  else
    $z \leftarrow z + 1$
    LLL-reduce $(\mathbf{b}_j, \mathbf{b}_{j+1})$
  end if
end while

In this version, z serves as a counter for the number of blocks that already satisfy the reduction

condition [5].

If we step through the logic for a block size $\beta = 3$ at index $j = 1$, the block consists of vectors $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. The algorithm explores the projected lattice $\pi_1(L)$ to find the shortest vector. The enumeration starts at the bottom of the search tree (the highest index, $k = 3$):

1. It picks an integer x_3 (e.g., $0, 1, -1, \dots$). This fixes a certain "height" in the projected lattice.
2. For that fixed x_3, it searches for a valid coefficient x_2.
3. For each pair (x_3, x_2), it searches for x_1.

It calculates the length of the resulting vector $\mathbf{v} = \sum_{i=1}^{3} x_i \pi_1(\mathbf{b}_i)$. If $\|\mathbf{v}\| < \delta\|\mathbf{b}_1^*\|$, the enumeration succeeds and returns this new vector for insertion [7].

A basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ is considered BKZ-reduced with block size β and parameter $\delta \in (1/4, 1]$ if it satisfies:

1. **Size-Reduction**: For all $1 \leq j < i \leq n$, the Gram-Schmidt coefficients satisfy $|\mu_{i,j}| \leq 1/2$.
2. **SVP Condition**: For each $i = 1, \dots, n-1$, the first vector of the local projected block satisfies:

$$\|\mathbf{b}_i^*\| \leq \lambda_1(\pi_i(\mathcal{L}(\mathbf{b}_i, \dots, \mathbf{b}_{\min(i+\beta-1,n)})))$$

   where $\mathbf{b}_i^*$ is the i-th Gram-Schmidt orthogonalized vector.

The process applies LLL-reduction to the initial basis $\mathbf{B}$ as a pre-processing step, followed by the main loop [7]:

- **Main Loop**: For $i = 1$ to $n - 1$:
  1. Let $h = \min(i + \beta - 1, n)$ be the end of the current block.
  2. Use an SVP solver (typically Enumeration or Sieve) to find the shortest non-zero vector $\mathbf{v}$ in the projected lattice $\pi_i(\mathcal{L}(\mathbf{b}_i, \dots, \mathbf{b}_h))$.
  3. If $\delta\|\mathbf{b}_i^*\| > \|\mathbf{v}\|$, update the basis by inserting the new vector and applying LLL to maintain independence and size-reduction. Because adding a new vector creates a set of $n + 1$ vectors in an n-dimensional space, the LLL-reduction identifies and removes the redundancy.
- Termination occurs, when the main loop is repeated until the basis remains unchanged for a full pass ($z = n - 1$). This indicates the basis has converged to a state where the SVP condition is satisfied for all blocks [5].

5.4 BKZ Variants

Different versions of BKZ have emerged as computational techniques improved. Significant advancements include the way SVP solvers are utilized—modern variants like BKZ 2.0 use *Extreme Pruning* to navigate the enumeration tree more efficiently, as searching the entire tree becomes prohibitively expensive for larger block sizes [5].

5.4.1 BKZ 2.0

BKZ 2.0 is an advanced variant of the BKZ algorithm that introduces several optimizations to handle larger block sizes [5]. A key feature is the use of *pre-reduction*: before running the SVP solver on a block of size k, BKZ 2.0 recursively applies BKZ with a smaller block size k' to that local block [5]. This ensures the basis is already well-conditioned, significantly increasing the efficiency of the subsequent SVP solver [5].

5.4.2 Geometric Series Assumption

The efficiency of the BKZ 2.0 algorithm, as introduced by Chen and Hanrot [5], is fundamentally tied to the *Geometric Series Assumption* (GSA) [5]. The GSA posits that after several tours of reduction, the log-lengths of the Gram-Schmidt vectors, defined as $L_i = \ln \|\mathbf{b}_i^*\|$, decay linearly. This linear slope is a primary metric for the quality of the basis; a flatter slope indicates a more reduced basis where the first vector $\mathbf{b}_1$ is significantly shorter.

Linked to BKZ also is the *Gaussian Heuristic*, which serves as a target for the local SVP solver within a block of size k. In practice, a Gaussian Gap often exists between the shortest vector found and the theoretical prediction of the heuristic. BKZ 2.0 optimizes this by using extreme pruning and early-abort strategies, terminating a tour once the observed log-lengths L_i align with the predicted GSA slope.

If the basis follows the Geometric Series Assumption (GSA), the log-lengths L_i decrease linearly. The Gaussian Heuristic acts as the "target" that each local SVP solver attempts to reach to maintain or flatten this slope.

Definition 5.1: Gaussian Heuristic

he Gaussian Heuristic (GH) provides an estimate for the length of the shortest vector in a "random" lattice. For a lattice $\mathcal{L}$ of dimension k, the heuristic predicts:

$$\lambda_1(\mathcal{L}) \approx GH(\mathcal{L}) = \frac{\Gamma(1+k/2)^{1/k}}{\sqrt{\pi}} \cdot (\det \mathcal{L})^{1/k} \approx \sqrt{\frac{k}{2\pi e}} \cdot (\det \mathcal{L})^{1/k}$$

5.5 Self-Dual BKZ

Self-Dual BKZ (SDBKZ) is another reduction algorithm parametrised by a block size k and an SVP oracle in dimension k. Based on the work by Micciancio and Walter [19], it improves upon standard BKZ by alternating the direction of reduction to handle the instability typically found in the final blocks of a basis [19].

Standard BKZ often leaves the last $k-1$ vectors of a basis (the tail) poorly reduced, as it is not reduced further in the final iteration of the algorithm. SDBKZ addresses this by ensuring that the basis is reduced with respect to both the primal and the dual lattice. A full SDBKZ tour consists of two phases:

- **Forward Tour:** This phase is similar to a standard BKZ tour. The algorithm iterates from $i = 1$ to $n-1$, performing SVP reduction on local projected blocks $\pi_i(\mathbf{B}_{[i,\min(i+k-1,n)]})$.

- **Backward Tour:** To address the instability of the basis tail, the algorithm proceeds backwards through the blocks. It applies the SVP oracle to the *dual* of the local projected blocks, which effectively reduces the basis from the bottom up.

In lattice reduction, there is a fundamental symmetry: the lengths of the Gram-Schmidt vectors of the primal basis ($||b_i||$) are directly related to the lengths of the dual basis vectors ($||b_i||^1$) [5].

Algorithm 6 Self-Dual BKZ, [19]

procedure SDBKZ($\mathbf{B}, k, \text{SVP}_k$)
  Input: A lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, block size k, SVP oracle in dimension k.
  Output: A k-reduced basis $\mathbf{B}'$.
  repeat
    for $i = 1 \dots n-1$ **do** {Forward Tour (Primal)}
      SVP reduce $\pi_i(\mathbf{B}_{[i,\min(i+k-1,n)]})$ using SVP_k
    end for
    for $i = n-1 \dots 1$ **do** {Backward Tour (Dual)}
      Dual SVP reduce $\pi_i(\mathbf{B}_{[i,\min(i+k-1,n)]})$ using SVP_k
    end for
  until no more progress is made
  return B
end procedure=0

6 Reduction Analysis

Lattice reduction involves a fundamental trade-off between the quality of the resulting basis specifically the length of the shortest vector - $\mathbf{b}_1$ - and the computational complexity required to find it. Generally, achieving a higher quality reduction (a shorter first basis vector) requires a significantly greater investment in processing time.

Having examined various reduction algorithms, it is useful to compare them directly to highlight their practical limitations. While the LLL algorithm provides a computationally efficient, polynomial-time solution, it offers a relatively loose approximation of the shortest vector. In contrast, the BKZ algorithm improves the basis quality by utilizing a block-wise approach, though it lacks a polynomial-time guarantee for large block sizes. Ultimately, there is no known polynomial-time solution for finding the absolute shortest vector in high dimensions. The most precise method, HKZ reduction, yields the highest quality basis but is computationally exhaustive, as it relies on solving the Shortest Vector Problem (SVP), which is known to be NP-hard.

Table 1 summarizes these trade-offs between approximation quality and complexity.

The quality of the basis for these algorithms was considered difficult to calculate, and in fact, it was only recently that a guarantee of the quality of the BKZ algorithm on any given tour was offered by Hanrot et al in 2011.

Algorithm	LLL	BKZ-β	HKZ
Quality	$2^{\frac{n-1}{4}} \cdot \det(L)^{1/n}$ (Exponentially weak)	$2\gamma_\beta^{\frac{n-1}{2(\beta-1)}+\frac{3}{2}} \cdot \det(L)^{1/n}$ (Scalable approximation)	$\sqrt{\gamma_n} \cdot \det(L)^{1/n}$ (Optimal / Shortest)
Complexity	$O(n^4 \log B)$ (Polynomial Time)	$O(n^3\beta^2 \log n)$ SVP_β calls (Variable, depends on β)	$2^{O(n)}$ or $n^{O(n)}$ (Super-Exponential)

Table 1: Lattice Reduction: Quality vs. Complexity Trade-off.

This is known as dynamic analysis, which gives an insight into the quality of the results after a certain number of iterations of the algorithm (some of their paper is discussed below)

6.1 Dynamic systems by Hanrot

To reason about the *quality* of the reduced basis at any given step, Hanrot et al. used a **profile function** that maps a lattice basis to a vector, combined with a **fixed point argument** to bound convergence [20].

6.1.1 The Profile Function

The proof below is paraphrased and follows from [20] on page 5, and proceeds in four steps:

1. Define a profile function $R(B)$ that measures basis quality.
2. Establish a recurrence $R(B_k) \leq R(B_{k-1}) \cdot M + v$.
3. Find the fixed point w satisfying $w = wM + v$, i.e. $w(I - M) = v$.
4. Show that $(R(B_0) - w) \cdot M^k \to 0$, so the basis quality converges to w.

Hanrot et al. define a profile function P that maps a lattice basis B to a vector:

$$P : B \mapsto \mathbf{p} \in \mathbb{R}^n$$

The termination condition for the algorithm is $P(B) \leq u$ for some small vector u. The smaller u is, the higher the quality of the basis B.

6.1.2 The Fixed Point Argument

Definition 6.1: Fixed Point

fixed point of a map F is a point w such that $F(w) = w$.

For the map $F(x) = x \cdot M + v$, a fixed point $w \in \mathbb{R}^n$ satisfies:

$$F(w) = w \cdot M + v = w$$

6.1.3 Solving for the Fixed Point

Starting from $F(w) = w$:

$$w \cdot M + v = w$$

Rearranging:

$$w - w \cdot M = v$$

$$\boxed{w(1 - M) = v}$$

6.1.4 Contracting the Recurrence

We now subtract the fixed point w from both sides of the recurrence:

$$R(B_k) - w \leq R(B_{k-1}) \cdot M + v - w$$

Substituting $v = w - w \cdot M$:

$$R(B_k) - w \leq R(B_{k-1}) \cdot M + (w - w \cdot M) - w$$

$$R(B_k) - w \leq R(B_{k-1}) \cdot M - w \cdot M$$

$$\boxed{R(B_k) - w \leq \big(R(B_{k-1}) - w\big) \cdot M}$$

The error $(R(B_k) - w)$ contracts by M at each step.

Applying the contraction repeatedly:

$$\begin{aligned} R(B_k) - w &\leq \big(R(B_{k-1}) - w\big) \cdot M \\ &\leq \big(R(B_{k-2}) - w\big) \cdot M^2 \\ &\leq \big(R(B_{k-3}) - w\big) \cdot M^3 \\ &\vdots \\ &\leq \big(R(B_0) - w\big) \cdot M^k \end{aligned}$$

The core strategy for proving the quality of a basis $\mathbf{B}$ involves mapping the basis to a *profile vector* in $\mathbb{R}^n$. The goal is to show that as the algorithm iterates, this profile $\mathbf{P}(\mathbf{B})$ converges toward a fixed point $\mathbf{w}$.

For a function F where $F(\mathbf{w}) = \mathbf{w}$, we define the update rule as:

$$\mathbf{x} \leftarrow \mathbf{x}\mathbf{M} + \mathbf{v} \tag{6}$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is a transition matrix and $\mathbf{v}$ is a constant vector. Hanrot et al. demonstrated that the algorithm terminates when the profile is bounded by this fixed point, expressed through the recurrence:

$$\mathbf{P}(\mathbf{B}_k) - \mathbf{w} \leq (\mathbf{P}(\mathbf{B}_{k-1}) - \mathbf{w})\mathbf{M} \tag{7}$$

By induction, this leads to the convergence term:

$$\mathbf{P}(\mathbf{B}_k) - \mathbf{w} \leq (\mathbf{P}(\mathbf{B}_0) - \mathbf{w})\mathbf{M}^k \tag{8}$$

Since $\mathbf{M}^k \to 0$ as k increases (provided the spectral radius of $\mathbf{M}$ is less than 1), the profile $\mathbf{P}(\mathbf{B}_k)$ eventually satisfies $\mathbf{P}(\mathbf{B}_k) \leq \mathbf{w}$. This fixed point $\mathbf{w}$ can then be related to the Hermite constant to bound the quality of the basis vectors.

6.2 Dynamical Systems by Li and Nguyen

In Li and Nguyen, they used a very similar profile function strategy, constructing a profile function, as well as a bounding condition, using a matrix, $n \times n$ matrix $M \geq 0$ and a vector $v \in \mathbb{R}^n$ such that for $k \geq 1$:

$$R(B_k) \leq R(B_{k-1}) \cdot M + v$$

Their profile function was bounded (at step k) by a linear function of the profile at step $k-1$.

They constructed their vector $c \in \mathbb{R}^n$ such that $R(B_0) - w \leq c$, giving [21]:

$$R(B_k) \leq |c| \cdot |M|^{2k} \cdot \mathbf{1}_n + w$$

Since $M^k \to 0$ as $k \to \infty$ (which holds when the initial basis B_0 is LLL-reduced), we get:

$$R(B_k) \to w \quad \text{as } k \to \infty$$

To quantify this progression, they utilized a specific profile function, denoted as P, used to bound the values within the basis matrix B. By establishing the constraint $P(B) \leq u$, they were able to track the basis's convergence toward a reduced state [21].

Li & Nguyen define their profile function using the **Rankin profile** (page 14 of [21]). Let L be a lattice and $B = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ a basis. The Rankin profile is:

$$R(B) = \big(R_1(B),\, R_2(B),\, \ldots,\, R_n(B)\big) \in \mathbb{R}^n$$

where each component is defined as:

$$R_i(B) = \log\left(\frac{\mathrm{vol}(B[1,i])}{\mathrm{vol}(L)^{i/n}}\right)$$

Here $B[1,i]$ denotes the sublattice spanned by the first i basis vectors, and $\mathrm{vol}(L)$ is the volume of the full lattice. Intuitively, $R_i(B)$ measures how far the volume of the first i basis vectors deviates from what it would be in a perfectly reduced basis.

Note that $R_n(B) = 0$ always, since when $i = n$:

$$R_n(B) = \log\left(\frac{\mathrm{vol}(L)}{\mathrm{vol}(L)^{n/n}}\right) = \log(1) = 0$$

So the last term in the Rankin profile is always zero.

Linking this profile function to the convergence, the Rankin profile converges to the fixed point w. This gives us the bound:

$$R(B_k) \leq w$$

which is the quality bound we are seeking. The fixed point w can then be related to known lattice constants, such as the Hermite constant, to characterise how good the reduced basis is.

6.3 Refinements by Li and Nguyen

Their research demonstrated that as the number of tours increases, the difference between the current basis profile $P(B_k)$ and a target value w approaches zero. This convergence $P(B_k) - w \to 0$ allows for the derivation of a more rigorous upper bound for the initial basis, such that $P(B_0) \leq w$ [21].

The primary impact of this dynamic analysis is the refinement of the boundary values for the length of the shortest vector in the basis. They proposed an improved boundary value formula that incorporates the block size β and the lattice volume more precisely than previous approximations offered than Hanrot [3][21]:

$$\upsilon_{\beta}^{\frac{n-1}{2(\beta-1)}+\frac{\beta(\beta-2)}{2n(\beta-1)}} \operatorname{vol}(\mathcal{L})^{\frac{1}{n}}$$

7 Conclusion

7.1 Summary

This research report provides a comprehensive technical review of both the foundations of lattice theory and the sophisticated methods used to assess the security of modern cryptosystems through basis reduction algorithms. The analysis begins with the classical Lenstra–Lenstra–Lovász (LLL) algorithm, which serves as the essential polynomial-time for obtaining reduced bases and remains useful and valuable in cryptanalysis. The report then transitions to stronger reduction methods, specifically the Block Korkine–Zolotarev (BKZ) algorithm and its advanced variants like BKZ 2.0 and Self-Dual BKZ, which generalize LLL by performing reduction on blocks of vectors to produce significantly shorter vectors at the cost of higher running times.

Rather than relying on static theoretical bounds or the Geometric Series Assumption (GSA), this study analysed the use of dynamic analyses following the frameworks established by Hanrot et al., and subsequently refined by Li and Nguyen in 2020, as the basis for future work on lattice reduction analysis on other variants such as dual-lattice based methods.

Dynamic analysis models the evolving state of the basis at the end of each tour, allowing for more realistic security estimates. This report identifies challenges in modern lattice cryptography and highlights how this whole family of algorithms have provided insight into the true difficulty of solving problems like finding close lattice points, which underpin the security of post-quantum schemes.

7.2 Future works

In future work, we hope to build on the separate and influential line of work by Hanrot, Li, and Nguyen, who introduced and studied profile functions and dynamic analysis techniques for BKZ, and apply their frameworks to dual-lattice based algorithms, which do not appear to have not been covered by the Generalised-BKZ algorithm and the dynamic analysis applied to it.

We aim to adapt and extend these ideas to our setting in **dual-lattice based BKZ algorithms**, to explore how dynamic analysis strategies could if there is a practical performance over dual lattice based BKZ methods.

A further direction of investigation will be to deepen our understanding of lattice and sublattice reductions, in contrast to the more classical focus on basis reduction alone. By examining how sublattice structure interacts with BKZ-style algorithms and their associated profiles, we hope to clarify the relationship between global lattice geometry and local reduction steps, and to potentially identify conditions under which sublattice strategies may lead to stronger or more efficient reductions.

8 Appendix

8.1 Proof of Hermite's inequality (for the Upper Bound on the Shortest Vector Length)

This follows the *LLL Algorithm Book* Proof by stepping through the proof on page 44 much more verbosely [3]:

8.1.1 Considering the base case ($d = 2$)

Let $L \subset \mathbb{R}^d$ be a d-dimensional lattice with volume $\text{vol}(L) = \det(L)$.

The length of the shortest non-zero vector is $\|\mathbf{b}_1\|$.

Let $\mathbf{b}'_2$ be the shortest vector in the projected lattice L'.

For $d = 2$, the bound holds on the shortest vector is known as:

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{1/4} \cdot \text{vol}(L)^{1/2}$$

And we also know from the lifting process that a lift requires that the boundary be such as:

$$\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \|\mathbf{b'}_2\|$$

The next step is to realise that we can lift $\mathbf{b}'_2$ into a non-zero vector $\mathbf{b}_2 \in L$ such that it satisfies $\mathbf{b}_2 = \mathbf{b}'_2 + \mu_{2,1}\mathbf{b}_1$, where $\mu_{2,1} \in \mathbb{R}$.

The size-reduction requirement on $\mathbf{b}_2$ ensures that $\mathbf{b}_2$ is chosen such that the coefficient $|\mu_{2,1}| \leq 1/2$. If we rely on Pythagoras giving us that the hypotenuse squared is the length of the square of the sides:

$$\|\mathbf{b}_2\|^2 = \|\mathbf{b}'_2\|^2 + \mu_{2,1}^2\|\mathbf{b}_1\|^2$$

Since the basis is chosen such that $\|\mathbf{b}_1\|$ is the shortest vector of the lattice L, it cannot be longer than $\mathbf{b}_2$, thus $\|\mathbf{b}_1\|^2 \leq \|\mathbf{b}_2\|^2$. Then we can re-write as:

$$\|\mathbf{b}_1\|^2 \leq \|\mathbf{b}'_2\|^2 + \mu_{2,1}^2\|\mathbf{b}_1\|^2$$

Rearranging the terms to relate $\|\mathbf{b}_1\|$ and $\|\mathbf{b}'_2\|$:

$$\|\mathbf{b}_1\|^2(1 - \mu_{2,1}^2) \leq \|\mathbf{b}'_2\|^2$$

Since $\mathbf{b}_2$ is size-reduced, we have $\mu_{2,1}^2 \leq \left(\frac{1}{2}\right)^2 = \frac{1}{4}$.

Then:

$$\|\mathbf{b}_1\|^2 \left(\frac{3}{4}\right) \leq \|\mathbf{b}_2'\|^2$$

$$\|\mathbf{b}_1\|^2 \leq \frac{4}{3} \cdot \|\mathbf{b}_2'\|^2$$

Taking the square root:

$$\|\mathbf{b}_1\| \leq \sqrt{\frac{4}{3}} \cdot \|\mathbf{b}_2'\|$$

8.1.2 Inductive Step ($d \to d-1$)

Let $\mathbf{b}_1$ be the shortest non-zero vector of L.

Let $L' = \pi_2(L)$ be the $(d-1)$-rank lattice obtained by projecting L over $\mathbf{b}_1^{\perp}$.

Let $\mathbf{b}_2'$ be a shortest non-zero vector of L', such that **the induction assumption holds**:

$$\|\mathbf{b}_2'\| \leq \left(\frac{4}{3}\right)^{(d-2)/4} \text{vol}(L')^{1/(d-1)}$$

The volume of the original lattice is related to the projected volume by $\text{vol}(L) = \|\mathbf{b}_1\| \cdot \text{vol}(L')$.

Now, let us assume the upper bound holds for any $(d-1)$-dimensional lattice L'. The shortest vector $\|\mathbf{b}_2'\|$ in L' is therefore assumed to be bounded by:

$$\|\mathbf{b}_2'\| \leq \left(\frac{4}{3}\right)^{((d-1)-1)/4} \cdot \text{vol}(L')^{1/(d-1)} = \left(\frac{4}{3}\right)^{(d-2)/4} \cdot \text{vol}(L')^{1/(d-1)}$$

Similarly we are also assuming that the shortest non-zero vector, $\lambda_1(L) = \|\mathbf{b}_1\|$, is bounded by:

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \cdot vol(L)^{1/d}$$

This is exactly the same form of the equation that occurs before it, only that it is now written for the dth number of basis vectors. Now we have a way of relating the two, based on our inductive proof, we can move to the next step.

The key intermediate step, which we get by relating the reduction condition and the induction hypothesis on the projected lattice L', yields:

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{d/4} \cdot (\text{vol}(L'))^{1/(d-1)}$$

Substitute the Volume Relation. We already know that the volume of the original lattice L is related to the projected lattice L' by $\text{vol}(L) = \|\mathbf{b}_1\| \cdot \text{vol}(L')$. substitute $\text{vol}(L') = \text{vol}(L)/\|\mathbf{b}_1\|$:

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{d/4} \cdot \left(\frac{\frac{\text{vol}(L)}{\|\mathbf{b}_1\|}}{\|\mathbf{b}_1\|}\right)^{1/(d-1)} = \left(\frac{4}{3}\right)^{d/4} \cdot \left(\frac{\text{vol}(L)}{\|\mathbf{b}_1\|^2}\right)^{1/(d-1)}$$

Isolate $\|\mathbf{b}_1\|$. move the $\|\mathbf{b}_1\|$ term from the right-hand side (RHS) to the left-hand side (LHS):

$$\|\mathbf{b}_1\| \cdot \left(\|\mathbf{b}_1\|^2\right)^{1/(d-1)} \leq \left(\frac{4}{3}\right)^{d/4} \cdot (\mathrm{vol}(L))^{1/(d-1)}$$

$$\|\mathbf{b}_1\|^{1+1/(d-1)} \leq \left(\frac{4}{3}\right)^{d/4} \cdot \mathrm{vol}(L)^{1/(d-1)}$$

$$\|\mathbf{b}_1\|^{d/(d-1)} \leq \left(\frac{4}{3}\right)^{d/4} \cdot \mathrm{vol}(L)^{1/(d-1)}$$

Raise both sides to the power of $(d-1)/d$:

$$\|\mathbf{b}_1\| \leq \left(\left(\frac{4}{3}\right)^{d/4}\right)^{(d-1)/d} \cdot \left(\mathrm{vol}(L)^{1/(d-1)}\right)^{(d-1)/d}$$

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \cdot \mathrm{vol}(L)^{1/d}$$

This leads to what is the Hermite Constant bound, which we can show below:

8.1.3 Theory behind Hermite Constant Bound

Before, we had

$$\|\mathbf{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \cdot \mathrm{vol}(L)^{1/d}$$

Squaring the LHS and the RHS gives us:

$$\|\mathbf{b}_1\|^2 \leq \left(\frac{4}{3}\right)^{(d-1)/2} \cdot \mathrm{vol}(L)^{2/d}$$

$$\frac{\|\mathbf{b}_1\|^2}{\mathrm{vol}(L)^{2/d}} \leq \left(\frac{4}{3}\right)^{(d-1)/2}$$

From [3], we had that the factor γ_d is the supremum of the left-hand side, and knowing that Hermite's Constant, $\gamma_2, = \sqrt{4/3}$ (thus $\gamma_2^2 = 4/3$), the bound holds for the Hermite constant and this can be re-written as:

$$\begin{aligned}\gamma_d &\leq \left(\frac{4}{3}\right)^{(d-1)/2} \\ &= (\gamma_2^2)^{(d-1)/2} \\ &= \gamma_2^{d-1}\end{aligned}$$

The $\gamma_2 = 4/3$ term is the constant from the two-dimensional reduction step. And, since the induction progresses dimension by dimension, applying a similar reduction step $d-1$ times, the bound accumulates at each stage:

$$\gamma_d \leq \gamma_2 \cdot \gamma_{d-1} \leq \gamma_2 \cdot (\gamma_2 \cdot \gamma_{d-2}) \leq \cdots \leq \underbrace{\gamma_2 \times \gamma_2 \times \cdots \times \gamma_2}_{d-1 \text{ times}}$$

Therefore, this boundary is something that we can meet at every dimension in the lattice. And it also demonstrates how the bound in d dimensions is built upon $d-1$ successive applications of the 2-dimensional reduction constant.

The reason the Hermite reduction algorithm (and the proof of the Hermite bound) must compare the shortest vector $\|\mathbf{b}_1\|$ against the shortest projected vector $\|\mathbf{b}_2'\|$ is to ensure the tightest possible shortness guarantee on $\mathbf{b}_1$ [3].

Because, when we look at b_2, we are not necessarily guaranteed the smallest value of b_2 to be in the second spot. Actually, it could be anything, because in the first case of this algorithm, b_2 has not been size reduced maximally.

The desired upper bound on $\|\mathbf{b}_1\|$ is derived from the properties of the entire lattice L. To prove the bound holds, $\mathbf{b}_1$ must be compared against the shortest possible element that can be constructed from the remaining basis vectors.

The algorithm's recursive step on the projected lattice $L' = \pi_2(L)$ is performed to guarantee that the vector $\mathbf{b}_2'$ is the **absolute shortest non-zero vector in the $(d-1)$-dimensional projected lattice** L'.

Because its a recursive algorithm, any vector $\mathbf{b}_i'$ in L' that is shorter than $\mathbf{b}_2'$ would have been ALREADY swapped into the $\mathbf{b}_2'$ position by the recursion.

If we were to skip the recursion, the resulting $\mathbf{b}_2'$ might simply be the projection of the original $\mathbf{b}_2$ (which could be actually quite large long), leading to a worse bond that the tightest possible.

Because the recursion ensures $\mathbf{b}_2'$ is minimised, the simple $\mathbf{b}_1/\mathbf{b}_2'$ check, $\|\mathbf{b}_1\|^2 \leq \frac{4}{3}\|\mathbf{b}_2'\|^2$, is enough to test that $\mathbf{b}_1$ is reduced according to the Hermite bound derived via induction.

8.2 Generalised BKZ

References

[1] Divesh Aggarwal, Thomas Espitau, et al, "Recursive lattice reduction—A framework for finding short lattice vectors" https://arxiv.org/abs/2311.15064v3

[2] NIST, *"Why prepare Now? Quantum Readiness"*, 2023, [Online], Available: https://www.cisa.gov/sites/default/files/2023-08/Quantum%20Readiness_Final_CLEAR_508c%20%283%29.pdf

[3] Nguyen, P. Q. (2010). The LLL Algorithm: Survey and Applications. Springer Publishing Company, Incorporated. ISBN: 978-3-642-02294-4.

[4] Jean-Philippe Aumasson, Serious Cryptography, 2nd Edition, "A Practical Introduction to Modern Encryption" August 2024, ISBN-13: 9781718503847

[5] Y. Chen and N. Hanrot, "BKZ 2.0: Better Lattice Reduction," in *Advances in Cryptology – ASIACRYPT 2011*, vol 7073, Springer, Berlin, Heidelberg, 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-25385-0_1

Algorithm 7 GBKZ: a generic BKZ algorithm [21]

Input: A blocksize $\beta \geq 2$, two relaxation factors $\delta \geq \eta \geq 1$, a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L in $\mathbb{R}^m$, and two GBKZ-compatible subroutines $\mathcal{A}_{\text{reduce}}$ and $\mathcal{A}_{\text{extract}}$.

Output: A new basis of L.

repeat

for $j = 1$ to $n - 1$ **do**

$\beta_j \leftarrow \min\{\beta, n - j + 1\}$; $n_j \leftarrow \min\{j + \beta - 1, n\}$

Find a primitive vector $\mathbf{b}$ for $L(\mathbf{b}_j, \ldots, \mathbf{b}_{n_j})$ such that $\|\pi_j(\mathbf{b})\| \leq \sqrt{\eta\gamma_{\beta_j}} \cdot \text{vol}(B_{[j,n_j]})^{1/\beta_j}$ {The factor η parameterises approximate SVP oracles used in the current BKZ implementation.}

if $\eta \times \|\mathbf{b}_j^*\|^2 > \delta \times \|\pi_j(\mathbf{b})\|^2$ **then**

Extract a new basis B by calling $\mathcal{A}_{\text{extract}}$ on the generator matrix $G = (\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}, \mathbf{b}, \mathbf{b}_j, \ldots, \mathbf{b}_n)$

else

Reduce B by calling $\mathcal{A}_{\text{reduce}}$ {Right before Step 8, we have $\|\mathbf{b}_j^*\| \leq \sqrt{\delta\gamma_{\beta_j}} \cdot \text{vol}(B_{[j,n_j]})^{1/\beta_j}$.}

end if

end for

until no change occurs or termination is requested, and **return** B.

[6] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[7] [SE91] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Fundamentals of Computation Theory*, Springer, 1991, pp. 68–91.

[8] [GN08a] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In STOC, pages 207–216, 2008.

[9] [LW23] J. Li and M. Walter. Improving convergence and practicality of slide-type reductions. Inf. Comput., 291:105012, 2023

[10] [ABLR21] M. R. Albrecht, L. Ducas, G. Herold, and P. Kirchner, "The General Sieve Kernel and New Records in Lattice Reduction," *Journal of Cryptology*, 2021.

[11] FPLLL Python Package, Lattice algorithms using floating-point arithmetic, Online, Availabe: https://github.com/fplll/fplll

[12] Planning for post-quantum cryptography, [Online], Available: https://www.cyber.gov.au/resources-business-and-government/governance-and-user-education/governance/planning-post-quantum-cryptography

[13] Chris Vale, Shor and Grover's algorothms, [Online], Available: https://kuscholarworks.ku.edu/server/api/core/bitstreams/0b163bff-f673-454e-b0e4-8a2e4abc7b9a/content

[14] Post-Quantum Cryptography Initiative , [Online], Available: https://www.cisa.gov/resources-tools/resources/quantum-readiness-migration-post-quantum-cryptography

[15] NIST, Announcing Approval of Three Federal Information Processing Standards (FIPS) for Post-Quantum Cryptography , [Online], Available: https://csrc.nist.gov/news/2024/postquantum-cryptography-fips-approved

[16] NIST, Finalising An Important Step Towards Quantum Safe Future, [Online], Available: https://cloudsecurityalliance.org/blog/2024/08/15/nist-fips-203-204-and-205-finalized-an-important-step-towards-a-quantum-safe-future

[17] IQM and VTT Launch Europe's 1st 50-Qubit Superconducting Quantum Computer, March 4, 2025 https://www.hpcwire.com/off-the-wire/iqm-and-vtt-launch-europes-1st-50-qubit-superconducting-quantum-computer/

[18] THE P VERSUS NP PROBLEM, Stephen Cook, https://www.claymath.org/wp-content/uploads/2022/06/pvsnp.pdf

[19] D. Micciancio and M. Walter, "Practical, Shortest Vector Computation in Lattice Cryptography," [Online]. Available: https://eprint.iacr.org/2015/1123.pdf

[20] Hanrot, G., Pujol, X., Stehlé, D. (2011). Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In: Rogaway, P. (eds) Advances in Cryptology – CRYPTO 2011. CRYPTO 2011. Lecture Notes in Computer Science, vol 6841. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22792-9_25

[21] Jianwei Li, Phong Q Nguyen, 'A Complete Analysis of the BKZ Lattice Reduction Algorithm', [Online], Available: https://hal.science/hal-04728596v1/document