

Master's Programme in Computer, Communication and Information Sciences

Parameter Selection in Cryptography based on Lattice Isomorphism

Cryptanalysis of the Lattice Isomorphism Problem

Christian E. Häggblom

© 2024

This work is licensed under a [Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/) “Attribution-NonCommercial-ShareAlike 4.0 International” license.



Author Christian E. Häggblom

Title Parameter Selection in Cryptography based on Lattice Isomorphism —
Cryptanalysis of the Lattice Isomorphism Problem

Degree programme Computer, Communication and Information Sciences

Major Computer Science

Supervisor Prof. Russell W. F. Lai

Advisor Prof. Russell W. F. Lai

Collaborative partner Aalto University

Date 21 December 2024

Number of pages 64

Language English

Abstract

The Lattice Isomorphism Problem (LIP) is a promising foundation for post-quantum cryptography due to its structural parallels with classical lattice problems and its computational hardness. This thesis explores the security of LIP, focusing on cryptanalytic reductions and parameter choices that influence its difficulty. Key contributions include an analysis and extension of the Hull Attack, which leverages the trivial hull property of certain lattices to reduce LIP to the Permutation Code Equivalence (PCE) problem, through a novel reduction from q -ary to p -ary LIP, under certain conditions.

Additionally, the thesis extends the Lattice Estimator to evaluate the complexity of solving LIP using various algorithms, such as Haviv and Regev’s algorithm and the Hull Attack. It also demonstrates the existence of orthogonal transformations beyond signed permutations between q -ary lattices, broadening the scope of isomorphic transformations. These results provide theoretical and practical insights into the resilience of LIP-based cryptographic schemes against emerging attacks.

Keywords Post-Quantum Cryptography, Lattice Isomorphism, Cryptanalysis, Linear Codes, q -ary Lattices, Code Equivalence, Lattice-based Cryptography

Författare Christian E. Häggblom

Titel Parameterval i Kryptografi baserad på Gitterisomorfism — Kryptanalys av Gitterisomorfismproblemet

Utbildningsprogram Computer, Communication and Information Sciences

Huvudämne Computer Science

Övervakare Prof. Russell W. F. Lai

Handledare Prof. Russell W. F. Lai

Samarbetspartner Aalto Universitet

Datum 21 December 2024

Sidantal 64

Språk engelska

Sammandrag

Lattice Isomorphism Problem (LIP) är en lovande grund för post-kvantkryptografi på grund av dess strukturella likheter med klassiska gitterproblem och dess beräkningsmässiga svårighet. Denna avhandling undersöker säkerheten hos LIP, med fokus på kryptanalytiska reduktioner och val av parametrar som påverkar dess komplexitet. Viktiga bidrag inkluderar en analys och utvidgning av Hull Attack, som utnyttjar egenskapen av trivial hull hos vissa gitter för att reducera LIP till Permutation Code Equivalence (PCE)-problemet, genom en ny reduktion från q -ärt till p -ärt LIP, under vissa villkor.

Avhandlingen utökar dessutom Lattice Estimator programmet för att utvärdera komplexiteten i att lösa LIP med olika algoritmer, såsom Haviv och Regev's algoritm samt Hull Attack. Den visar också existensen av ortogonala transformationer bortom signerade permutationer mellan q -ära gitter, vilket breddar omfånget av isomorfa transformationer. Dessa resultat ger både teoretiska och praktiska insikter i motståndskraften hos LIP-baserade kryptografiska system mot framväxande attacker.

Nyckelord Post-kvantkryptografi, Gitterisomorfism, Kryptanalys, Linjära koder, q -ära gitter, Kodekvivalens, Gitterbaserad kryptografi

Preface

I would like to thank Professor Russell W. F. Lai for the instrumental support and help in writing this thesis, and for the flexibility and adaptability, in practice and pedagogy, making my time at Aalto much more enjoyable. I would also like to thank the Cryptography Group at Aalto as a whole, for the help and assistance when needed, and for the fun and friendly atmosphere at seminars and conferences.

And finally, I of course want to thank my family and friends for supporting me during my studies and making the whole endeavor an experience to remember, and for being patient and understanding, even when it would be understandable to not be.

Otaniemi, 21 December 2024

Christian E. Häggblom

Contents

Abstract	3
Abstract (in Swedish)	4
Preface	5
Contents	6
Symbols and Abbreviations	8
1 Introduction	9
1.1 Contributions	10
1.2 Open Problems	10
1.3 Organization	11
2 Preliminaries	12
2.1 Codes	12
2.1.1 Linear Codes	12
2.1.2 Code Equivalence Problem (CEP)	12
2.1.3 Permutation Equivalence Problem (PEP)	12
2.1.4 Signed Permutation Equivalence Problem (SPEP)	12
2.1.5 The Hull of a Code and Its Dimension	13
2.2 Lattices	13
2.2.1 Q-ary lattices	13
2.2.2 Construction A	14
2.2.3 Gram Matrices	14
2.2.4 Rank	15
2.2.5 Determinant	15
2.2.6 Dual Lattice	16
2.2.7 Basis Change	16
2.2.8 Quadratic Form	16
2.2.9 Relation Between Bases	17
2.2.10 Lattice Reduction	17
2.2.11 Lattice Isomorphism Problem (LIP)	18
3 Previous LIP Cryptanalysis	20
3.1 LIP Haviv Regev	20
3.2 A Formal and Convenient Framework for LIP-Based Cryptography	22
3.3 Hull Attacks on Lattice Isomorphism	24
3.4 Module LIP	25
3.4.1 Cryptanalytic Insights and Structured Assumptions	25
3.4.2 Generalizing Cryptanalysis Through Parallel Cases	25

4	Cryptanalysis of LIP	27
4.1	Hull Attacks and their Significance	27
4.1.1	Field Characteristics	28
4.1.2	Properties of Lattices and Codes	28
4.1.3	Trivial hulls in lattices	28
4.1.4	Reduction of q LIP to p LIP	28
4.2	Avoiding Attacks through Non-trivial Hulls	33
4.3	Constraints on Codes with Non-trivial Hulls	33
4.3.1	Combinatorial Constraints from Enforcing a Non-Trivial Hull	34
4.3.2	Impact on Construction A Lattices and Cryptographic Security	34
4.3.3	Open problem	35
4.4	Extension of q -ary Lattice Transformations	35
4.5	Explicit Two-Dimensional Example	37
4.6	General Rational Orthogonal Transformations	41
5	Estimating Concrete Hardness of LIP	42
5.1	The Lattice Estimator	42
5.2	LIP parameters	42
5.3	PEP estimator module	43
5.3.1	Information Set Decoding (ISD)	44
5.3.2	Leons' Algorithm	45
5.3.3	Beullens'	46
5.3.4	Support Splitting Algorithm	48
5.3.5	BOS Algorithm	49
5.4	LIP estimator module	51
5.4.1	Haviv and Regev's attack	51
5.4.2	Hull Attack	53
5.4.3	Arithmetic Invariants	55
5.4.4	Geometric Invariants	57
6	Conclusions	59
	References	63

Symbols and Abbreviations

Symbols

\mathcal{L}	A lattice defined in Euclidean space \mathbb{R}^n
B	Basis of a lattice
Λ	Construction A lattice, q -ary lattice
q	Modulus of q -ary lattices or codes
C	Linear code
$\phi^{-1}(C)$	Preimage of a code under natural reduction modulo q
G	Gram matrix of a lattice basis
$\lambda_1(\mathcal{L})$	First successive minimum of a lattice
$\det(\mathcal{L})$	Determinant of a lattice \mathcal{L}
$H(C)$	Hull of a code C
$H_s(\mathcal{L})$	S-hull of a lattice \mathcal{L}
δ	Root-Hermite factor achieved by lattice reduction
β	Block size parameter for BKZ reduction
O	Orthogonal transformation
U	Unimodular transformation

Abbreviations

LIP	Lattice Isomorphism Problem
PCE	Permutation Code Equivalence
PEP	Permutation Equivalence Problem (same as PCE)
ISD	Information Set Decoding
SPCE	Signed Permutation Code Equivalence
SPEP	Signed Permutation Equivalence Problem (same as SPCE)
SVP	Shortest Vector Problem
LLL	Lenstra-Lenstra-Lovász lattice reduction algorithm
BKZ	Block Korkine-Zolotarev lattice reduction algorithm
SZK	Statistical Zero-Knowledge
CRT	Chinese Remainder Theorem
FHE	Fully Homomorphic Encryption
MPC	Multi-Party Computation

1 Introduction

The advent of quantum computing poses a critical threat to the security of classical cryptographic systems, many of which rely on computational problems that quantum algorithms can efficiently solve, such as factoring integers and computing discrete logarithms [Sho97]. This challenge has spurred a global effort to develop cryptographic systems resistant to quantum attacks. Among the most promising candidates for post-quantum cryptography are lattice-based schemes, which are built on the computational hardness of lattice problems such as the Learning with Errors (LWE) [Reg05] and the Shortest Vector Problem (SVP) [Ajt96]. These problems not only exhibit worst-case hardness but also offer practical advantages in terms of efficiency and scalability. However, their concrete security often depends heavily on the choice of parameters, which remains an area of active research [AGPS19].

The Lattice Isomorphism Problem (LIP) has recently emerged as a compelling candidate for cryptographic constructions due to its structural parallels with other lattice problems and its potential for efficient decoding capabilities [Mic11]. Unlike LWE or SVP, LIP involves determining whether two given lattices are isomorphic under an orthogonal transformation [HR13]. This structural problem is intriguing not only for its theoretical implications but also for its practical cryptographic applications, as it introduces a novel approach to designing secure systems [Beu20].

Despite its promise, the cryptanalysis of LIP is still in its infancy. Previous work has explored its complexity and connections to other computational problems, such as the Graph Isomorphism Problem [Bab16], yet significant gaps remain in understanding its concrete hardness and the impact of parameter selection. This thesis addresses these gaps by systematically analyzing LIP's security and proposing new tools for its evaluation.

The main contributions of this thesis are threefold. First, we present a detailed survey of existing cryptanalytic techniques for LIP, highlighting their strengths and limitations. Second, we extend these techniques to explore new attack vectors, including the Hull Attack, which exploits structural properties of lattices to reduce LIP to the Permutation Code Equivalence (PCE) problem [Sen06]. We demonstrate a novel reduction from q -ary to p -ary LIP, and extend the transformation space between q -ary lattices beyond that of the currently known signed permutations, providing valuable new insights into the problem's complexity. Third, we develop and extend the Lattice Estimator [APS15] to include modules for evaluating the concrete security of LIP-based schemes. This tool incorporates the most recent algorithms, enabling a comprehensive analysis of parameter selection and its impact on security.

By bridging theoretical and practical perspectives, this thesis advances the understanding of LIP and its role in post-quantum cryptography. It offers both rigorous theoretical insights for further evaluating the problem's complexity and practical guidelines for designing secure systems based on LIP. In doing so, it lays the groundwork for future research on the hardness of LIP.

In this work, we intend to investigate whether the increased structure in lattices can be leveraged to attack LIP on other similarly structured lattices. We also want to answer the question of how well we can estimate the concrete security of LIP, with the

current knowledge and algorithms.

1.1 Contributions

In this work, the contributions are of both practical and theoretical nature, and below a brief summary of the contributions are given, in the order they appear in the work:

- We give an overview of previous research made on the security of LIP. This overview is focused on the security of the Lattice Isomorphism Problem (LIP), and the attacks produced by their cryptanalysis. These attacks are the ones that our extension of the Lattice Estimator will be based on, and hence will be referred to later on. Furthermore, we note that the results suggest looking at specific cases of LIP, due to inherent connections to other problems.
- We then analyze the security of LIP-based cryptography and show that the properties of the underlying structures, such as the hull, and their connection to other problems significantly affect the security of schemes. We go on to give a reduction from q LIP to p LIP for, and discuss what the possible implications of this would be.
- Based on the discoveries made in the previous section, we show the existence of orthogonal transformations between q -ary lattices that are not signed permutation matrices. This decreases the impact of the previous q LIP- p LIP reduction on the use of q LIP in cryptographic schemes.
- We extend the Lattice Estimator to include security estimation modules for LIP allowing investigation of how the selection of parameters affect the concrete security of LIP as a foundation for cryptographic schemes. We do this by taking into account previous research into the security of LIP, reviewing successful attacks on LIP and the effects of these attacks on the security of LIP as a whole, as well as future implementations using LIP as a foundation.

1.2 Open Problems

The following are questions that this work leaves open for further research:

- **Effect of trivial hulls on structure in construction A lattices:** In section 4.2 we describe the resulting reduction in the number of valid codes, if we ensure a code has a non-trivial hull. This has the direct effect of reducing the size of the code space, but whether this restricted subset of codes has some symmetries or other vulnerabilities that can be used to attack lattice problems on the associated Construction A lattices, and what these are, needs to be investigated further.
- **General rational orthogonal transformations between q -ary lattices:** We gave a proof, in section 4.4, for the existence of orthogonal transformations

between q -ary lattices that are not signed permutations, followed by a concrete example. However, a more general understanding of the limitations and which transformations satisfy the conditions for such transformations, and a general method for constructing such transformations, remains open.

- **Characterizing quadratic forms for general LIP estimation parameters:** We note the difficulty of estimating the security of LIP without specific quadratic forms for an instance. Finding and characterizing classes of quadratic forms associated with specific classes of lattices would alleviate this issues and allow for more general parameters to estimate security on.

1.3 Organization

The remainder of this thesis is organized as follows: Chapter 2 covers the preliminaries for codes and lattices in cryptography, as well as the notions required for talking formally about LIP.

Chapter 3 reviews related work on LIP, highlighting the security of LIP for specific cases and the attacks that have been discovered so far, giving an overview of the current state and possible directions for cryptanalysis of LIP.

Chapter 4 explores the security of LIP on q -ary lattices and the connection to other problems, and gives a reduction from q -ary LIP to p -ary LIP for a $q \in \mathbb{N}$ and a prime p , under specific circumstances. Then, a proof of the existence of orthogonal transformations between q -ary lattices that are not signed permutations, expanding the search space and mitigating the effect of the previous reduction to some extent.

Chapter 5 delves into the implementation of the estimator modules for lattices on LIP, and codes on PCE. In this section, we also briefly note the addition of new invariants to the Estimator, beyond the ones already investigated in the LIP context, that would aid in giving a more precise estimation of security for quadratic forms.

Finally, Chapter 6 concludes the thesis with conclusions and a summary of results and contributions, along with suggestions for future research directions.

2 Preliminaries

In this section, we introduce the foundational concepts and terminologies essential for understanding the subsequent analysis on the estimation of security of lattices for post-quantum cryptography, with a focus on the Lattice Isomorphism Problem (LIP).

2.1 Codes

Here we give the essential definitions and concepts from coding theory that we need in this work. The concepts we introduce here are relevant to our work on lattices via the construction A method introduced later, and so the definitions given here are brief and are explained only to the extent they are relevant here.

2.1.1 Linear Codes

A linear code C of length n and dimension k over a finite field \mathbb{F}_q , denoted as $[n, k]_q$, is a k -dimensional subspace of the vector space \mathbb{F}_q^n . It is characterized by a generator matrix $G \in \mathbb{F}_q^{k \times n}$, where each row of G is a codeword in C . The dual code C^\perp of C is the set of vectors orthogonal to every codeword in C under the standard inner product, with dimension $n - k$.

2.1.2 Code Equivalence Problem (CEP)

The Code Equivalence Problem (CEP) considers two linear codes C_1 and C_2 over \mathbb{F}_q . The codes are said to be equivalent if there exists a permutation π of the coordinate positions and a non-singular matrix $M \in \mathbb{F}_q^{n \times n}$ such that $C_2 = \pi(C_1)M$.

2.1.3 Permutation Equivalence Problem (PEP)

The Permutation Equivalence Problem (PEP) is a special case of the CEP where the equivalence between two linear codes C_1 and C_2 is restricted to permutations of coordinate positions. Formally, C_1 and C_2 are permutation equivalent if there exists a permutation matrix $P \in \mathbb{F}_q^{n \times n}$ such that $C_2 = C_1P$. This problem is also known as the Permutation Code Equivalence (PCE) problem, and in this work the terms can be considered interchangeable.

2.1.4 Signed Permutation Equivalence Problem (SPEP)

The Signed Permutation Equivalence Problem (SPEP) extends the concept of code equivalence to include multiplication of coordinates by non-zero scalars from \mathbb{F}_q , in addition to permutations. Codes C_1 and C_2 are said to be signed permutation equivalent if there exists a diagonal matrix D with non-zero entries from \mathbb{F}_q and a permutation matrix P such that $C_2 = C_1DP$. Similar to the PEP/PCE problem, this problem also has an alternative naming, Signed Permutation Code Equivalence (SPCE), which can also be considered interchangeable with SPEP.

2.1.5 The Hull of a Code and Its Dimension

The hull of a linear code C , denoted $\text{Hull}(C)$, plays a critical role in the structural analysis of the code. It is defined as the intersection of C with its dual C^\perp . Mathematically, the hull is given by

$$\text{Hull}(C) = C \cap C^\perp.$$

The dimension of the hull, $\dim(\text{Hull}(C))$, is a fundamental parameter that reflects the intrinsic symmetry properties of the code and is defined as the dimension of this intersection as a subspace over \mathbb{F}_q . It provides insights into the complexity of certain code equivalence problems and can significantly impact the algorithmic approaches to solving them.

The dimension of the hull can range from 0 to k , where k is the dimension of C . A hull of dimension 0, where $\text{Hull}(C) = \{\mathbf{0}\}$, implies that C and C^\perp only intersect at the zero vector, suggesting C is as far from being self-dual as possible. On the other end of the spectrum, a hull of maximum dimension k indicates that C is self-dual, containing vectors orthogonal to every vector in itself, including the non-zero vectors.

The computational complexity associated with the Code Equivalence Problem (CEP), Permutation Equivalence Problem (PEP), and Signed Permutation Equivalence Problem (SPEP) can be heavily influenced by the dimension of the hull. For instance, codes with higher-dimensional hulls have a larger group of automorphisms, which may simplify the equivalence problems under certain conditions. In contrast, codes with low-dimensional hulls may present more complex equivalence challenges.

In the context of post-quantum cryptography, the dimension of the hull can be exploited to design cryptosystems with particular security properties or to assess the vulnerability of code-based cryptosystems to various attack strategies. It is, therefore, a crucial aspect of code-based cryptography that merits in-depth exploration.

2.2 Lattices

A lattice \mathcal{L} in an n -dimensional Euclidean space \mathbb{R}^n is defined as the set of all integral linear combinations of a set of k linearly independent basis vectors $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k \in \mathbb{R}^n$, where $k \leq n$. Formally, the lattice generated by the basis $B = [\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_k]$ is given by

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

2.2.1 Q-ary lattices

A q -ary lattice is a special type of lattice in \mathbb{R}^n associated with a modulus $q \in \mathbb{Z}_{>0}$. Given a matrix $A \in \mathbb{Z}_q^{k \times n}$, the q -ary lattice $\mathcal{L}_q(A)$ is defined as the set of all integer vectors in \mathbb{Z}^n that satisfy the homogeneous linear congruence modulo q :

$$\mathcal{L}_q(A) = \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{A}\mathbf{x} \equiv \mathbf{0} \pmod{q}\}.$$

In other words, $\mathcal{L}_q(A)$ consists of all integer vectors whose images under the linear transformation defined by A are congruent to the zero vector modulo q . This lattice can be viewed as the kernel (null space) of A over the ring \mathbb{Z}_q , lifted to the integer lattice \mathbb{Z}^n .

2.2.2 Construction A

Construction A is a procedure to construct a lattice from a linear code, and can be defined as follows: Given a linear code $C \subseteq \mathbb{Z}_q^n$ over the finite field \mathbb{Z}_q , we construct a q -ary lattice Λ in \mathbb{R}^n as follows:

$$\Lambda = q\mathbb{Z}^n + \phi^{-1}(C),$$

where $q\mathbb{Z}^n$ denotes the set of all vectors in \mathbb{Z}^n scaled by q , and $\phi^{-1}(C)$ is the preimage of C under the natural reduction modulo q map $\phi : \mathbb{Z}^n \rightarrow \mathbb{Z}_q^n$. Explicitly, the lattice Λ consists of all integer vectors whose components reduce modulo q to a codeword in C :

$$\Lambda = \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{x} \bmod q \in C\}.$$

In other words, Λ is formed by lifting the code C from \mathbb{Z}_q^n to \mathbb{Z}^n and adding all multiples of q . This construction embeds the algebraic structure of the code into the geometric structure of the lattice, allowing techniques from coding theory to be applied to lattice problems.

2.2.3 Gram Matrices

A Gram matrix is a fundamental concept in lattice theory that captures the inner product relationships between the basis vectors of a lattice. Given a basis $B = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_k]$ of a lattice \mathcal{L} in \mathbb{R}^n , where each $\mathbf{b}_i \in \mathbb{R}^n$, the Gram matrix G associated with B is defined as:

$$G = B^\top B = (\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{i,j=1}^k,$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product in \mathbb{R}^n , and B^\top is the transpose of B . The entry G_{ij} of the Gram matrix represents the inner product between the basis vectors \mathbf{b}_i and \mathbf{b}_j :

$$G_{ij} = \langle \mathbf{b}_i, \mathbf{b}_j \rangle = \mathbf{b}_i^\top \mathbf{b}_j.$$

The Gram matrix G is a symmetric positive-definite matrix that encapsulates the geometric properties of the lattice generated by B , such as the lengths of the basis vectors and the angles between them.

2.2.4 Rank

Definition 2.1 (Rank of a Lattice). Let \mathcal{L} be a lattice in the Euclidean space \mathbb{R}^n . The rank of the lattice \mathcal{L} is defined as the maximum number of linearly independent vectors in \mathcal{L} .

More formally, there exists a set of linearly independent vectors $\{v_1, v_2, \dots, v_r\}$ in \mathbb{R}^n such that every element of \mathcal{L} can be written as an integer linear combination of these vectors:

$$\mathcal{L} = \left\{ \sum_{i=1}^r a_i v_i \mid a_i \in \mathbb{Z} \right\}.$$

The integer r is called the rank of the lattice \mathcal{L} . If the rank r is equal to n , the lattice \mathcal{L} is called full-rank lattice. If $r < n$, the lattice is referred to as a lower rank lattice and spans a subspace of \mathbb{R}^n of dimension r .

Consider the lattice $\mathcal{L} \subset \mathbb{R}^3$ generated by the vectors $(1, 0, 0)$ and $(0, 1, 0)$. This lattice consists of all points of the form $(m, n, 0)$ where m and n are integers. The rank of this lattice is 2, because it is generated by two vectors, even though it is embedded in three-dimensional space.

2.2.5 Determinant

Definition 2.2 (Determinant of a lattice). Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, which is defined as a discrete subgroup of \mathbb{R}^n that spans the real vector space \mathbb{R}^n . Assume \mathcal{L} is generated by a set of linearly independent vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ in \mathbb{R}^n . The determinant of the lattice \mathcal{L} , denoted $\det(\mathcal{L})$, is defined as the absolute value of the determinant of the matrix M formed by taking $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ as its columns:

$$M = (\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n).$$

Explicitly, if $\mathbf{v}_i = (v_{i1}, v_{i2}, \dots, v_{in})^T$ for $i = 1, 2, \dots, n$, then

$$M = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix}.$$

The determinant of \mathcal{L} is then given by

$$\det(\mathcal{L}) = |\det(M)|.$$

Geometrically, the absolute value of the determinant of M represents the n -dimensional volume of the parallelepiped spanned by the generating vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$. For a lattice \mathcal{L} , this volume, given by $\det(\mathcal{L})$, is known as the *fundamental volume* or *covolume* of the lattice. It measures the "density" of the lattice points in \mathbb{R}^n . A smaller determinant indicates a denser lattice, while a larger determinant indicates a sparser lattice.

2.2.6 Dual Lattice

Definition 2.3 (Dual lattice). Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice generated by the basis B . The dual lattice \mathcal{L}^* of the lattice \mathcal{L} is defined as:

$$\mathcal{L}^* = \{\mathbf{y} \in \mathbb{R}^n \mid \langle \mathbf{y}, \mathbf{x} \rangle \in \mathbb{Z} \text{ for all } \mathbf{x} \in \mathcal{L}\},$$

where $\langle \mathbf{y}, \mathbf{x} \rangle$ denotes the standard inner product in \mathbb{R}^n . The dual lattice \mathcal{L}^* is generated by the columns of the matrix $(B^T)^{-1}$, where B^T is the transpose of the matrix B .

2.2.7 Basis Change

Any lattice \mathcal{L} has infinitely many different bases. If B and B' are two different bases for the same lattice \mathcal{L} , then there exists a unimodular matrix $U \in \mathbb{Z}^{k \times k}$ (a square integer matrix with determinant ± 1) such that $B' = BU$.

2.2.8 Quadratic Form

A quadratic form in the context of lattices is inherently represented by the Gram matrix G of the lattice basis. Given a lattice \mathcal{L} in \mathbb{R}^n with basis $B = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_k]$, the Gram matrix G captures the inner products of the basis vectors:

$$G = B^T B = (\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{i,j=1}^k,$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product.

The quadratic form $Q : \mathbb{R}^k \rightarrow \mathbb{R}$ associated with G is defined by:

$$Q(\mathbf{x}) = \mathbf{x}^T G \mathbf{x} = \sum_{i=1}^k \sum_{j=1}^k G_{ij} x_i x_j,$$

for any vector $\mathbf{x} = [x_1, x_2, \dots, x_k]^T \in \mathbb{R}^k$. This quadratic form evaluates the squared norm of the lattice vector $\mathbf{v} = B\mathbf{x}$:

$$Q(\mathbf{x}) = \|\mathbf{v}\|^2.$$

Thus, the Gram matrix G itself defines the quadratic form Q , encoding the geometric properties of the lattice. The quadratic form allows us to:

- Calculate the lengths of lattice vectors via $\|\mathbf{v}\| = \sqrt{Q(\mathbf{x})}$.
- Determine the angles between lattice vectors using the inner products encoded in G .
- Analyze the lattice's structure and classify it based on its quadratic form.

2.2.9 Relation Between Bases

If B and B' are two different bases for the same lattice \mathcal{L} , then there exists a unimodular matrix U such that $B' = BU$. The Gram matrices G and G' of the bases B and B' are related by

$$G' = U^T G U.$$

This relationship ensures that the quadratic form and hence the geometric properties of the lattice remain invariant under a change of basis.

2.2.10 Lattice Reduction

Lattice reduction is a process used to find a basis of a lattice that consists of shorter and more orthogonal vectors compared to the original basis. This process transforms the basis vectors of the lattice in such a way that the resulting basis provides a better representation for solving various lattice problems. The concept of lattice reduction is fundamental in computational number theory and cryptography.

The most well-known lattice reduction algorithm is the Lenstra-Lenstra-Lovász (LLL) algorithm, introduced by Lenstra, Lenstra, and Lovász in 1982 [LLL82]. The LLL algorithm takes a basis of a lattice and produces a new basis with the property that the vectors are relatively short and nearly orthogonal. This is achieved through a series of operations that iteratively reduce the size of the basis vectors while maintaining the lattice structure. The LLL algorithm operates in polynomial time and transforms a given lattice basis B into a reduced basis B' with certain approximation guarantees. The exact time complexity of the LLL algorithm is:

$$T_{\text{LLL}} = O\left(n^5 \left(\max_{1 \leq i \leq n} \log \|\mathbf{b}_i\|\right)^3\right),$$

where:

- n is the dimension of the lattice,
- \mathbf{b}_i are the basis vectors of B ,
- $\|\mathbf{b}_i\|$ denotes the Euclidean norm of \mathbf{b}_i .

The LLL algorithm produces a reduced basis $B' = [\mathbf{b}'_1, \mathbf{b}'_2, \dots, \mathbf{b}'_n]$ such that:

$$\|\mathbf{b}'_1\| \leq 2^{(n-1)/2} \lambda_1(\mathcal{L}),$$

where $\lambda_1(\mathcal{L})$ is the length of the shortest non-zero vector in the lattice \mathcal{L} . This means that LLL provides an exponential approximation factor in n and cannot find arbitrarily short vectors in high-dimensional lattices.

Another important lattice reduction algorithm is the Block Korkine-Zolotarev (BKZ) algorithm, which extends the ideas of the LLL algorithm to higher dimensions. The BKZ algorithm, introduced by Schnorr and Euchner in 1994 [SE94], enhances the LLL algorithm by introducing a block size parameter β , where $2 \leq \beta \leq n$. The exact time complexity of the BKZ algorithm is:

$$T_{\text{BKZ}} = O\left(n^{2+\varepsilon} \cdot T_{\text{SVP}}(\beta) \cdot \left(\max_{1 \leq i \leq n} \log \|\mathbf{b}_i\|\right)\right),$$

where:

- ε is a small constant (arising from polynomial factors),
- $T_{\text{SVP}}(\beta)$ is the time complexity of solving the Shortest Vector Problem (SVP) in dimension β ,
- $T_{\text{SVP}}(\beta)$ is typically exponential in β , i.e.,

$$T_{\text{SVP}}(\beta) = 2^{c\beta+o(\beta)},$$

with $c > 0$ being a constant dependent on the SVP solver used.

The BKZ algorithm achieves an improved approximation factor:

$$\|\mathbf{b}'_1\| \leq \delta_\beta^{(n-1)/(\beta-1)} \lambda_1(\mathcal{L}),$$

where δ_β is the root Hermite constant in dimension β , satisfying:

$$\delta_\beta = \left(\frac{\beta}{2\pi e}\right)^{1/(2\beta)}.$$

As β increases, the quality of the reduced basis improves, but the runtime grows exponentially due to the dependence on $T_{\text{SVP}}(\beta)$. Still, BKZ is useful for problems where the LLL algorithm alone is insufficient, such as when the approximation factor is important.

Lattice reduction is crucial for solving lattice problems because many of these problems are computationally hard with arbitrary bases. Reduced bases, such as those produced by LLL and BKZ, provide a more structured and manageable form, enabling efficient solutions to problems like the shortest vector problem (SVP) and the closest vector problem (CVP). These problems have significant applications in cryptography, particularly in constructing and analyzing lattice-based cryptosystems.

2.2.11 Lattice Isomorphism Problem (LIP)

In the context of lattice theory and its applications to cryptography, the Lattice Isomorphism Problem (LIP) is formalized as follows:

Definition 2.4 (Lattice Isomorphism). Two lattices $L, L' \subseteq \mathbb{R}^n$ are said to be *isomorphic* if there exists an orthonormal transformation $O \in O_n(\mathbb{R})$ such that

$$L' = O \cdot L.$$

Such an orthonormal O is sometimes called an isometry.

Computationally, we approach the problem by considering the bases of the lattices. For two bases $B, B' \in \mathbb{R}^{n \times m}$, the lattices they generate are isomorphic if there exists an orthonormal transformation $O \in O_n(\mathbb{R})$ and a unimodular transformation $U \in GL_m(\mathbb{Z})$ such that

$$B' = OBU.$$

This definition of lattice isomorphism underpins several computational problems in cryptography. Notably, the challenge of LIP lies in determining whether two given bases generate isomorphic lattices and, if so, finding the isomorphism represented by the matrices O and U .

To alleviate some of the computational inefficiencies associated with real-valued coordinates and orthonormal transformations, LIP can be reformulated using the language of quadratic forms. For a given lattice basis B , the Gram matrix $Q = B^t B$ defines a positive-definite quadratic form that encapsulates the geometric structure of the lattice. The reformulation focuses on determining the equivalence of two quadratic forms Q and Q' under unimodular transformations. This avoids the need for explicit orthonormal transformations.

Definition 2.5 (Lattice Isomorphism Problem, Quadratic Form Formulation). Given two quadratic forms $Q, Q' \in \mathbb{R}^{n \times n}$, determine whether there exists a unimodular matrix $U \in GL_n(\mathbb{Z})$ such that

$$Q' = U^t Q U.$$

This formulation shifts the problem to a purely algebraic domain while preserving the core geometric relationships. The equivalence class of a quadratic form Q , denoted $[Q]$, consists of all forms equivalent to Q under unimodular transformations. Importantly, the Gram matrix Q retains the intrinsic geometric properties of the lattice, such as vector norms and angles, allowing the quadratic form representation to fully characterize the lattice.

The quadratic form formulation not only simplifies the representation of the problem but also enables connections to other areas of mathematics, such as number theory and algebraic geometry. It highlights the structural richness of LIP and provides a pathway for leveraging algebraic tools in its study. Moreover, the theoretical results established in this thesis demonstrate that current cryptographic schemes based on LIP inherently rely on properties of these quadratic forms, further emphasizing their central role in both the theoretical and practical aspects of the problem.

3 Previous LIP Cryptanalysis

In the quest for cryptographic systems that can withstand the advent of quantum computing, significant efforts have been invested in exploring the robustness of lattice-based cryptography. Recently, the landscape of lattice-based cryptography has been notably enriched by rigorous investigations into the security and structure of lattices, including in the domain of LIP.

In this section we survey previous work that have significantly contributed to the current understanding of LIP, its complexity, its connections to other problems, and its potential cryptographic applications. This section highlights the focus of the previous work, namely, general investigations of LIP, and more restricted versions like module-LIP. This serves to highlight the notion of security of LIP in general and the possibility of drastic deviations in security in specific versions of LIP, thereby suggesting a more pointed focus on specific groups of lattices that are interesting for a variety of application or efficiency reasons. We present an algorithm by [HR13] that is the only currently known provable attack, while the other attacks, like the Hull Attack by [DG23], and the invariants iterated by [Dv22], are heuristic in nature.

Furthermore, the results presented here also bring forth the point of looking at the connections between different problems on different structures that share the inherent nature of the problem, as well as the different problems that share the underlying structure of the mathematical objects upon which it operates.

3.1 LIP Haviv Regev

Haviv and Regev [HR13] provide a significant contribution to the study of the Lattice Isomorphism Problem (LIP) by presenting a quasi-exponential time algorithm for LIP that relies on finding short vectors, and outputs all orthogonal transformations mapping one lattice to another. This follows the theme of being "lattice-based in the cryptanalytic sense," as Ducas and van Woerden [Dv22] called it, where the best algorithms rely on finding short vectors for a lattice. The algorithm runs in $n^{\mathcal{O}(n)}$ time and utilizes polynomial space, making it optimal up to constant factors in the exponent. This result not only deepens our understanding of the computational complexity of LIP but also highlights its structural parallels with other combinatorial and lattice-based problems.

In addition to their algorithmic advancements, the authors prove that LIP lies within the complexity class Statistical Zero-Knowledge (SZK). This is a compelling result, as it demonstrates the existence of a statistical zero-knowledge proof system for LIP. Consequently, this places LIP in the intersection of AM and coAM, indicating that it is unlikely to be NP-hard unless the polynomial hierarchy collapses. Such findings align LIP with other notable lattice problems, such as the Shortest Vector Problem (SVP) with polynomial approximation factors, which are also believed to be computationally challenging but not NP-hard.

Algorithm Summary

The algorithm has a runtime of $n^{\mathcal{O}(n)}$ and utilizes polynomial space, making it optimal up to constant factors in the exponent. The algorithm operates by distinguishing between two cases: the special case, where certain structural properties of the lattices allow for direct analysis, and the general case, which requires a recursive decomposition of the problem into smaller subproblems. Below we give a summary of the algorithm:

Special Case Algorithm: The special case applies when the lattices L_1 and L_2 have the same set of shortest nonzero vectors, denoted by $\lambda_1(L_1) = \lambda_1(L_2)$. The algorithm proceeds as follows:

1. For each lattice L_1 and L_2 , compute the sets of shortest vectors A_1 and A_2 , where $A_i = \{x \in L_i \mid \|x\| = \lambda_1(L_i)\}$, and an extended set of vectors $W_i = \{x \in L_i \mid \|x\| \leq 5n^{17/2} \cdot \lambda_1(L_i^*)\}$.
2. For each $w_1 \in W_1$, check if w_1 uniquely defines a linearly independent chain of length n in A_1 . If so, extract this chain (x_1, \dots, x_n) .
3. Similarly, for each $w_2 \in W_2$, check if w_2 uniquely defines a linearly independent chain of length n in A_2 . If so, extract this chain (y_1, \dots, y_n) .
4. Map each x_i to y_i via a candidate linear transformation O , and verify if O is orthogonal and satisfies $L_2 = O(L_1)$. Add O to the output set if it is valid.

The special case terminates when all candidate mappings are validated, outputting all orthogonal transformations that map L_1 to L_2 .

General Case Algorithm: When the special case conditions do not hold, the general case algorithm recursively decomposes the problem into smaller instances. The steps are as follows:

1. Compute the subspaces V_1 and V_2 spanned by the shortest vectors of L_1 and L_2 , respectively, and the projection of L_1 and L_2 onto the orthogonal complement of these subspaces.
2. Verify that the ranks of the intersections $L_1 \cap V_1$ and $L_2 \cap V_2$ are equal. If not, the lattices are not isomorphic, and the algorithm terminates.
3. Apply the special case algorithm to the intersections $L_1 \cap V_1$ and $L_2 \cap V_2$ to compute candidate transformations O_1 for the subspaces.
4. Recursively apply the general case algorithm to the projections $\pi_1(L_1)$ and $\pi_2(L_2)$ onto their orthogonal complements to compute candidate transformations O_2 for the complementary subspaces.
5. For each pair of transformations (O_1, O_2) , construct a combined transformation O defined on $\text{span}(L_1)$, where $O|_{V_1} = O_1$ and $O|_{\pi_1(\text{span}(L_1))} = O_2$.

6. Check the validity of each combined transformation O by verifying that O is orthogonal and satisfies $L_2 = O(L_1)$. If valid, add O to the output set.

The general case algorithm constructs the solution by recursively reducing the problem dimensionally until it reaches the base case.

The recursive nature of the general case algorithm ensures that the problem is broken down into tractable subproblems. The use of the special case algorithm at each step limits the combinatorial explosion of candidate transformations. The overall runtime is $n^{\mathcal{O}(n)}$, dominated by the number of recursive calls and the combinatorial complexity of mapping shortest vectors. This complexity is optimal up to constant factors in the exponent. Furthermore, the algorithm suggests the "lattice-based" nature of LIP from a cryptanalytic point of view, i.e., where the best-known methods of attack depend fundamentally on lattice reduction techniques. They also note that while this algorithm outputs all linear transformations, a problem left open is that of finding a possibly more efficient algorithm which only decides LIP.

3.2 A Formal and Convenient Framework for LIP-Based Cryptography

In their work, Léo Ducas and Wessel van Woerden [DvW21] provide a formal framework for the study of the Lattice Isomorphism Problem (LIP), presenting it through the lens of quadratic forms. This framework defines LIP in terms of equivalence classes of positive-definite quadratic forms, derived from a lattice basis as shown in the preliminaries. This enables the problem to be studied using tools from other areas, and additionally allows us to avoid the problems of floating-point arithmetic in practice. Below, we present the formal problem formulations and the associated invariants introduced in their work.

Problem Formulations.

1. *Search-LIP (s-LIP)*: Given two quadratic forms $Q_1, Q_2 \in \mathcal{S}_n^+$, the task is to determine the unimodular transformation under which Q_1 and Q_2 are equivalent. That is, find a unimodular matrix $U \in GL_n(\mathbb{Z})$ such that:

$$Q_2 = U^T Q_1 U.$$

2. *Distinguish-LIP (Δ -LIP)*: Given two quadratic forms $Q_0, Q_1 \in \mathcal{S}_n^+$ with the promise that one of them, say Q_b for $b \in \{0, 1\}$, belongs to a given equivalence class $[Q]$, determine the value of b . This is equivalent to distinguishing whether Q_0 or Q_1 is in the same equivalence class as Q , under unimodular transformations.

These formulations capture the essence of LIP in terms of quadratic forms, shifting the focus from real-valued orthogonal transformations to discrete unimodular equivalence. This perspective simplifies the computational framework while preserving the problem's inherent structure.

Arithmetic and Geometric Invariants

To analyze and distinguish quadratic forms, Ducas and van Woerden introduce several invariants that are preserved under unimodular transformations. These invariants are divided into two categories:

1. Arithmetic Invariants:

- **Determinant** ($\det(Q)$): The determinant of the quadratic form Q is given by $\det(Q)$, and it corresponds to the square of the lattice's volume.
- **GCD Invariant**: For a quadratic form Q represented by a symmetric matrix M , the GCD invariant is defined as:

$$\gcd(M_{11}, M_{22}, \dots, M_{nn}),$$

where M_{ii} are the diagonal entries of M .

- **Parity**: The parity of Q checks whether all diagonal elements of M are even. It is defined as:

$$\text{parity}(Q) = \begin{cases} 0, & \text{if all diagonal entries are even;} \\ 1, & \text{otherwise.} \end{cases}$$

- **Discriminant Modulo p** : The discriminant modulo a prime p is a finer classification invariant. For a matrix M associated with Q , the discriminant modulo p provides additional resolution for distinguishing equivalence classes.
- **Signature**: The signature of Q is a tuple (p, n) , where p and n denote the number of positive and negative eigenvalues of the matrix M , respectively.
- **Cassels-Hasse Invariant**: A local invariant computed using the Hilbert symbol:

$$\prod_{i < j} \left(\frac{M_{ii}, M_{jj}}{p} \right),$$

where p is a prime.

2. Geometric Invariants:

- **Successive Minima** ($\lambda_1(Q), \dots, \lambda_n(Q)$): The lengths of the shortest linearly independent vectors in the lattice corresponding to Q .
- **Theta Series** ($\Theta_Q(x)$): The formal series defined as:

$$\Theta_Q(x) = \sum_{v \in \mathbb{Z}^n} e^{-\pi x \|v\|_Q^2},$$

where $\|v\|_Q = v^\top Q v$ is the norm induced by the quadratic form.

- **Kissing Number** ($\kappa(Q)$): The number of shortest nonzero vectors in the lattice associated with Q , which corresponds to the number of points on the shortest sphere.

These invariants serve as tools for distinguishing equivalence classes of quadratic forms and give us a way to further understand the practical security of LIP when building cryptographic schemes.

Ducas and van Woerden’s framework highlights the cryptographic relevance of LIP by showing that the problem can be instantiated with structured lattices for secure cryptographic constructions. The reliance on arithmetic and geometric invariants facilitates the design of cryptosystems that are not only efficient but also robust against attacks exploiting structural properties of lattices.

In summary, the quadratic form framework and associated invariants formalize a practical formulation of LIP in a mathematically precise way, enabling its study in both theoretical and applied cryptographic contexts, such as the case of the lattice estimator. These contributions provide an important foundation for advancing the understanding and utilization of LIP in cryptographic schemes in practice.

3.3 Hull Attacks on Lattice Isomorphism

The cryptanalytic work by Léo Ducas and Shane Gibbons [DG23] casts a discerning light on the LIP, unearthing the potential for what are termed as Hull Attacks. This method exploits the intersection of a lattice with its scaled dual, known as the hull, to facilitate a cryptanalysis that poses a substantial threat to certain lattice configurations. Léo Ducas and Wessel van Woerden [DvW21] mention the possibility of such attacks but leave the actual finding of an attack open, which is the subject of this work.

Ducas and Gibbons’ investigation into the s-hull of a lattice—a variation adapted from the concept of a code’s hull—reveals a nuanced landscape where the hull’s geometry critically influences the security of lattice-based cryptosystems. They contend that while the s-hull does not lend itself to arithmetic distinguishers due to predictability of genus properties, it may considerably lower the bar for geometric attacks. Their insights culminate in an attack that, while remaining exponential in time complexity, significantly reduces the constant in the exponent, posing a challenge to the previously held conjectures regarding the hardness of LIP.

Their findings indicate that the geometry of hulls must be a paramount consideration when instantiating LIP for cryptographic purposes. Unimodular lattices, characterized by the equivalence of the lattice and its dual, emerge as a strategic choice to mitigate hull-based vulnerabilities. Intriguingly, Ducas and Gibbons note that existing proposals for LIP instantiations already exhibit this characteristic, thus inherently eschewing hull attacks.

The Hull Attack redefines the security narrative within lattice-based cryptography, underscoring the imperative for continuous scrutiny of lattice structures and their susceptibilities. The ramifications of Ducas and Gibbons’ work affects other ongoing

cryptographic research, as the possibility of using the hull to attack LIP forces us to consider a new avenue of vulnerabilities, and consider this in any security analysis.

3.4 Module LIP

The HAWK scheme, introduced in [DPPvW22], extends the Lattice Isomorphism Problem (LIP) paradigm by leveraging rank-2 module lattices over $R = \mathbb{Z}[X]/(X^n + 1)$, where n is a power of two. These lattices provide a balance between cryptographic security and computational efficiency, making them well-suited for practical applications. Unlike rank-1 module lattices (ideal lattices), which are vulnerable to polynomial-time attacks, rank-2 constructions mitigate these risks while maintaining efficient implementations.

HAWK achieves notable efficiency improvements by avoiding floating-point arithmetic and rejection sampling, enabling faster signature generation compared to schemes like Falcon. Additionally, it offers a worst-case to average-case reduction for module-LIP, further grounding its security in established theoretical hardness.

3.4.1 Cryptanalytic Insights and Structured Assumptions

HAWK operates on cyclotomic fields, which allow structured lattices to be used effectively. This design introduces practical efficiencies but also restricts the generality of its security assumptions. For instance, HAWK relies on the hardness of module-LIP in power-of-two cyclotomic fields, a more constrained setting than general LIP. Recent cryptanalysis of rank-2 module-LIP in totally real fields [MPMPW24] demonstrates that such structural assumptions can sometimes introduce vulnerabilities. While these specific attacks do not directly apply to cyclotomic fields, they highlight how inherent algebraic structures can be exploited to weaken security.

The methodology used in cryptanalysis of totally real fields, which involves decompositions of Gram matrices and relative norms, illustrates a broader principle: understanding specific structural properties can lead to significant cryptanalytic breakthroughs, potentially extendable to other cases. Although the specific setting of totally real fields differs from HAWK's cyclotomic construction, these techniques provide valuable insights that may inform future analyses of cyclotomic fields or other lattice structures.

3.4.2 Generalizing Cryptanalysis Through Parallel Cases

A key lesson from the cryptanalysis of rank-2 module-LIP in totally real fields is the value of exploring parallel but distinct cases. Even if the attacks on totally real fields do not directly apply to HAWK, they demonstrate how targeted investigations into specific algebraic structures can reveal vulnerabilities or guide the development of general attack methodologies. For example, cryptanalysis of q -ary lattices might uncover techniques that, with suitable adaptation, could be generalized to other lattice settings.

This principle underscores the importance of systematically studying LIP in diverse settings. Investigating specialized cases, such as module-LIP in other structured fields, may not always yield immediate attacks on schemes like HAWK but can lay the groundwork for broader cryptanalytic advances. By identifying and understanding parallel cases, researchers can build a deeper understanding of LIP's complexity and resilience, ultimately informing the design of more robust cryptographic systems. In the rest of this work, we intend to do exactly that.

4 Cryptanalysis of LIP

4.1 Hull Attacks and their Significance

In the study of cryptographic systems based on lattice structures and their correspondence to linear codes, the choice of modulus plays a crucial role in maintaining the mathematical and computational integrity of the systems. Ducas and Gibbons [DG23] specify the use of a prime modulus p for their techniques, primarily because structures formed by prime moduli exhibit properties that are critical for the ease of analysis, as the main purpose of their paper being providing a counter-example to the original security definition of LIP made by [DvW21]. This section elaborates on the reasons behind this choice in the context of the connection between codes and lattices, emphasizing the characteristics of the hull in the context of lattice-based cryptography. We then give a reduction for qLIP to pLIP, which allows us to use similar techniques as used by [DG23] to also solve LIP on certain q -ary lattices in quasi-polynomial time. Furthermore, we give a proof of an extension of the transformation space between q -ary lattices beyond just signed permutation matrices, reducing the significance of the reduction and giving more credence to the security of qLIP.

Below is brief summary of the hull attack and the associated important results:

Lemma 1 (Hull and Triviality, [DG23]). *Let $C \subseteq \mathbb{F}_p^n$ be an $[n, k]_p$ linear code. The p -hull of the associated Construction A lattice $L = C + p\mathbb{Z}^n$ satisfies:*

$$H_p(L) = H(C) + p\mathbb{Z}^n.$$

If $H(C) = \{0\}$, then $H_p(L) = p\mathbb{Z}^n$.

Lemma 2 (Reduction to Signed Permutations, [DG23]). *Let $L_1 = O_1(C + p\mathbb{Z}^n)$ and $L_2 = O_2(C + p\mathbb{Z}^n)$ be two isomorphic lattices defined by orthonormal transformations $O_1, O_2 \in O_n(\mathbb{R})$. An isometry between L_1 and L_2 can be reduced to a signed permutation equivalence between C_1 and C_2 , the respective reductions modulo p of L_1 and L_2 .*

Theorem 1 (Hull-Based Reduction of LIP, [DG23]). *Let $C \subseteq \mathbb{F}_p^n$ be an $[n, k]_p$ linear code whose hull is trivial, i.e., $H(C) = \{0\}$. Let $L_1 = O_1(C + p\mathbb{Z}^n)$ and $L_2 = O_2(C + p\mathbb{Z}^n)$ be two isomorphic lattices defined via orthonormal transformations $O_1, O_2 \in O_n(\mathbb{R})$. Solving LIP on L_1 and L_2 reduces to:*

1. Solving ZLIP (isomorphism of \mathbb{Z}^n).
2. Solving SPEP (signed permutation equivalence problem) on the associated code C .

Proof of the Theorem, adapted from [DG23]. Consider a lattice $L = O(C + p\mathbb{Z}^n)$ with determinant p^{n-k} . By Lemma 1, the hull $H_p(L)$ is equivalent to $p\mathbb{Z}^n$ under the same orthogonal transformation O . Solving ZLIP provides an isometry ϕ from $p\mathbb{Z}^n$ to $H_p(L)$, up to a signed permutation σ . Applying ϕ^{-1} to L_1 and L_2 yields lattices whose reductions modulo p define the codes C_1 and C_2 . By Lemma 2, their equivalence reduces to solving SPEP for C_1 and C_2 . \square

4.1.1 Field Characteristics

The primary reason for choosing a prime modulus lies in the algebraic structure of the set of integers modulo p , denoted as $\mathbb{Z}/p\mathbb{Z}$. Unlike rings formed by composite numbers or prime powers, $\mathbb{Z}/p\mathbb{Z}$ also forms a field, which ensures that each non-zero element has a multiplicative inverse and there are no zero divisors. Mathematically, these properties are denoted as:

$$\forall a \in \mathbb{Z}/p\mathbb{Z}, a \neq 0 \Rightarrow \exists a^{-1} : aa^{-1} \equiv 1 \pmod{p},$$

$$\forall a, b \in \mathbb{Z}/p\mathbb{Z}, ab \equiv 0 \pmod{p} \Rightarrow a = 0 \text{ or } b = 0.$$

These characteristics are crucial for the operations involved in constructing lattice structures with a correspondence to codes, facilitating clear and concise manipulations.

4.1.2 Properties of Lattices and Codes

The integrity and the simplicity of the correspondence between lattices and codes are best maintained under a prime modulus. This correspondence relies on the predictable behavior of arithmetic operations within fields, crucial for aligning lattice and code structures in cryptographic contexts. The use of a non-prime modulus complicates this correspondence due to the introduction of additional algebraic structures. This can be expressed as:

$$\text{If } n = pq, \text{ with } p, q > 1, \text{ then } \exists a, b \in \mathbb{Z}/n\mathbb{Z} : ab \equiv 0 \pmod{n}, a \neq 0, b \neq 0.$$

4.1.3 Trivial hulls in lattices

The hull attack requires the hull of the Construction A lattice to be isomorphic to the "trivial", that is $p\mathbb{Z}^n$. The connection between both the corresponding linear code and construction A lattice, as well as their respective hulls is given by [DG23] for p -ary lattices with a prime p , and is rather direct. However, for q -ary lattices, the connection is not necessarily as clear and direct. The following reduction gives an attack that applies the properties of the hull not only to p -ary and associated lattices, but for a significant amount of q -ary lattices, given some specific requirements.

4.1.4 Reduction of q LIP to p LIP

Here we give a reduction from q LIP to p LIP showing that under some reasonable assumptions we can reduce the problem of LIP on q -ary lattice to LIP on p -ary lattices.

Definition 4.1 (CRT-decomposition of a q -ary Lattice via the Chinese Remainder Theorem). Let $q \in \mathbb{Z}$ be a composite number with the factorization $q = p_1^{e_1} \times p_2^{e_2} \times \dots \times p_k^{e_k}$, where the $p_i^{e_i}$ are pairwise coprime. Consider a q -ary lattice $\Lambda_q(A) \subset \mathbb{Z}_q^n$ generated by the matrix $A_q \in \mathbb{Z}_q^{n \times m}$.

We define the set ϕ as the collection of lattices $\Lambda_{p_i^{e_i}}(A)$ for each $p_i^{e_i}$, where:

$$\phi = \left\{ \Lambda_{p_i^{e_i}}(A) \mid \Lambda_{p_i^{e_i}}(A) = \left\{ \mathbf{x} \in \mathbb{Z}_{p_i^{e_i}}^n \mid \mathbf{x} = A_{p_i^{e_i}} \mathbf{y} \pmod{p_i^{e_i}}, \mathbf{y} \in \mathbb{Z}_{p_i^{e_i}}^m \right\} \right\},$$

and $A_{p_i^{e_i}} = A_q \pmod{p_i^{e_i}}$ is the reduction of A_q modulo $p_i^{e_i}$.

The original q -ary lattice $\Lambda_q(A)$ is isomorphic to the direct product of the lattices in ϕ :

$$\Lambda_q(A) \cong \prod_{i=1}^k \Lambda_{p_i^{e_i}}(A),$$

where each $\Lambda_{p_i^{e_i}}(A) \in \phi$ is a lattice over $\mathbb{Z}_{p_i^{e_i}}$ generated by the matrix $A_{p_i^{e_i}} \in \mathbb{Z}_{p_i^{e_i}}^{n \times m}$. This decomposition allows computations in the larger ring \mathbb{Z}_q to be performed efficiently in the smaller rings $\mathbb{Z}_{p_i^{e_i}}$.

Using this definition of the decomposition of q -ary lattices, we can describe the method of reducing LIP of two q -ary lattices to LIP on the corresponding elements of their CRT-decompositions.

Theorem 2. *Let $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$ be two q -ary lattices that are isomorphic under a signed permutation O . Suppose q decomposes into prime powers as $q = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, and there exists a prime p_i such that $e_i = 1$. Then, for the CRT-decomposed components of $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$ modulo p_i , the components $\Lambda_{p_i}(A_1)$ and $\Lambda_{p_i}(A_2)$ are isomorphic under the same signed permutation O .*

Given an oracle \mathcal{A} that solves the signed permutation code equivalence problem for p_i -ary linear codes over \mathbb{F}_{p_i} with trivial hulls, in quasi-polynomial time, we can solve the isomorphism problem for $\Lambda_{p_i}(A_1)$ and $\Lambda_{p_i}(A_2)$ in quasi-polynomial time.

Proof. We start with two q -ary lattices $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$, generated by matrices $A_1, A_2 \in \mathbb{Z}^{m \times n}$. Assume there exists a signed permutation O such that:

$$\Lambda_q(A_1) = O \Lambda_q(A_2),$$

meaning the lattices are isomorphic under O . Our goal is to show that for a prime $p_i^{e_i}$ with exponent $e_i = 1$ in the prime factorization of q , the CRT-decomposed components modulo p_i , namely $\Lambda_{p_i}(A_1)$ and $\Lambda_{p_i}(A_2)$, are isomorphic under the signed permutation O modulo p_i , and have corresponding linear codes over \mathbb{F}_{p_i} .

First, from the definition above we have that any q -ary lattice $\Lambda_q(A)$ can be decomposed via the Chinese Remainder Theorem (CRT) into a direct product of lattices modulo each prime power factor of q :

$$\Lambda_q(A) \cong \prod_{j=1}^k \Lambda_{p_j^{e_j}}(A).$$

In particular, when $e_i = 1$, the p_i -ary component is defined as:

$$\Lambda_{p_i}(A) := \{x \in \mathbb{Z}^m \mid Ax \equiv 0 \pmod{p_i}\}.$$

Here, $\Lambda_{p_i}(A)$ is a lattice in \mathbb{Z}^m , defined by modular constraints modulo p_i .

Since p_i is prime, the inverse of Construction A corresponds to reduction mod p_i , which when applied on $\Lambda_{p_i}(A)$ gives us the corresponding linear code over the finite field \mathbb{F}_{p_i} . Thus, we have:

$$\mathcal{C}_{p_i}(A) := \Lambda_{p_i}(A) \pmod{p_i}.$$

Now, consider the action of the signed permutation matrix O modulo p_i . Since O is a signed permutation matrix, its entries are in $\{0, \pm 1\}$. When we reduce O modulo p_i , the entries become elements of \mathbb{F}_{p_i} :

$$O \pmod{p_i} = O_{p_i},$$

where each entry o_{jk} of O modulo p_i satisfies:

$$o_{jk} \equiv 0 \pmod{p_i} \text{ if } o_{jk} = 0, o_{jk} \equiv 1 \pmod{p_i} \text{ if } o_{jk} = 1, o_{jk} \equiv -1 \pmod{p_i} \text{ if } o_{jk} = -1.$$

Since $-1 \equiv p_i - 1 \pmod{p_i}$, the entries of O_{p_i} are in $\{0, 1, p_i - 1\}$.

Therefore, O_{p_i} is a matrix over \mathbb{F}_{p_i} with entries in $\{0, 1, p_i - 1\}$, corresponding to the reduction of the signed permutation matrix modulo p_i . Importantly, O_{p_i} acts as a coordinate permutation combined with scalar multiplication by 1 or -1 (modulo p_i).

Applying O_{p_i} to $\Lambda_{p_i}(A_2)$, we will have:

$$\Lambda_{p_i}(A_1) = O_{p_i} \Lambda_{p_i}(A_2),$$

meaning that the isomorphism between $\Lambda_{p_i}(A_1)$ and $\Lambda_{p_i}(A_2)$ is equivalent to the isomorphism between $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$. This reduces the lattice isomorphism problem for $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$ to the signed permutation code equivalence problem over \mathbb{F}_{p_i} .

Now, we have an oracle \mathcal{A} that can solve the signed permutations code equivalence problem for linear codes over \mathbb{F}_{p_i} with trivial hulls, and thus we can use \mathcal{A} to determine the signed permutation under which $\Lambda_{p_i}(A_1)$ and $\Lambda_{p_i}(A_2)$ are equivalent. A code has a trivial hull if the intersection of the code with its dual is trivial. In such cases, the code equivalence problem under signed permutations can be solved in quasi-polynomial time.

Therefore, by considering the CRT decomposition of q -ary lattices, we have shown that the lattice isomorphism problem reduces to the signed permutation code equivalence problem for linear codes over a finite field of prime order, given that q has a factor $p_i^{e_i}$ with $e_i = 1$. The reduction of the signed permutation matrix O modulo p_i results in entries in $\{0, 1, p_i - 1\}$, corresponding to $\{0, 1, -1\}$ modulo p_i . Furthermore, this connection allows us to solve the original lattice isomorphism problem for any two q -ary lattices $\Lambda_q(A_1)$ and $\Lambda_q(A_2)$, in quasi-polynomial time, given a non-power prime factor of q , trivial hull, and an SPEP oracle \mathcal{A} .

□

This of course requires us to have an algorithm that can find the unique prime factorization of an integer in polynomial time, which currently is not known to exist

in the non-quantum space. However, since we are considering a problem that is meant to be secure against quantum adversaries, it is reasonable to assume that the adversary has access to a quantum computer with a sufficient number of qubits, for which there exists an efficient algorithm to factor integers, namely, Shor's algorithm. Still, for the parameters generally considered secure, and thus what we consider here, the decomposition is tractable and can be computed without access to quantum computation.

It then follows that when there exists a factor $p_i^{e_i}$ where p_i is prime, and $e_i = 1$, we can also solve LIP using an approach similar to the hull attack outlined in [DG23]. The hull attack specifically concerns lattices that are isomorphic to a Construction A lattice with modulus p , then by finding this isomorphism through the hull the problem is solved in the corresponding codes by solving the Permutation Code Equivalence (PCE) problem, on the assumption of a trivial hull in the corresponding codes.

Then, since q -ary lattice may be isomorphic to different lattices than the original p -ary lattices defined in the original hull attack, this naturally extends the hull attack to the lattices isomorphic to some q -ary lattices, namely, those that are isomorphic by a signed permutation and have at least one prime factor $p_i^{e_i}$ with $e_i = 1$.

Next, we determine the amount of q -ary lattices this applies to for different q . We show that the conditions that allow for the reduction, as explained above, apply to most moduli q .

Lemma 3. *Let n be a positive integer. Denote q as an integer from the set $\{1, 2, 3, \dots, n\}$. Let $r(n)$ be the ratio of integers q that have a prime factorization, such that there exists at least one prime factor $p_i^{e_i}$ with $e_i = 1$. Then, $r(n)$ can be bounded from below as follows:*

$$r(n) \geq 1 - \frac{\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor + \left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2}{n}$$

Proof. Consider the set of integers $\{1, 2, 3, \dots, n\}$. We aim to count the number of integers in this set that have at least one prime factor p_i with $e_i = 1$, which means we need to exclude integers that are perfect powers and products of perfect prime powers.

A perfect power is a number of the form a^k where a and k are integers and $k \geq 2$. The number of perfect powers up to n can be approximated by summing up the counts for each exponent k :

$$\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor$$

This gives us the count of numbers that are perfect powers of some integer.

Next, we exclude products of perfect prime powers. A product of perfect prime powers is of the form $p_1^{e_1} p_2^{e_2} \dots p_m^{e_m}$ where all $e_i > 1$. These can be counted by considering the combinations of perfect prime powers up to n :

$$\left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2$$

The ratio $r(n)$ is given by:

$$r(n) = \frac{n - \sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor - \left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2}{n}$$

As n grows large, the terms $\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor$ and $\left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2$ grow slower than n , implying:

$$r(n) \approx 1 - \frac{\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor + \left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2}{n}$$

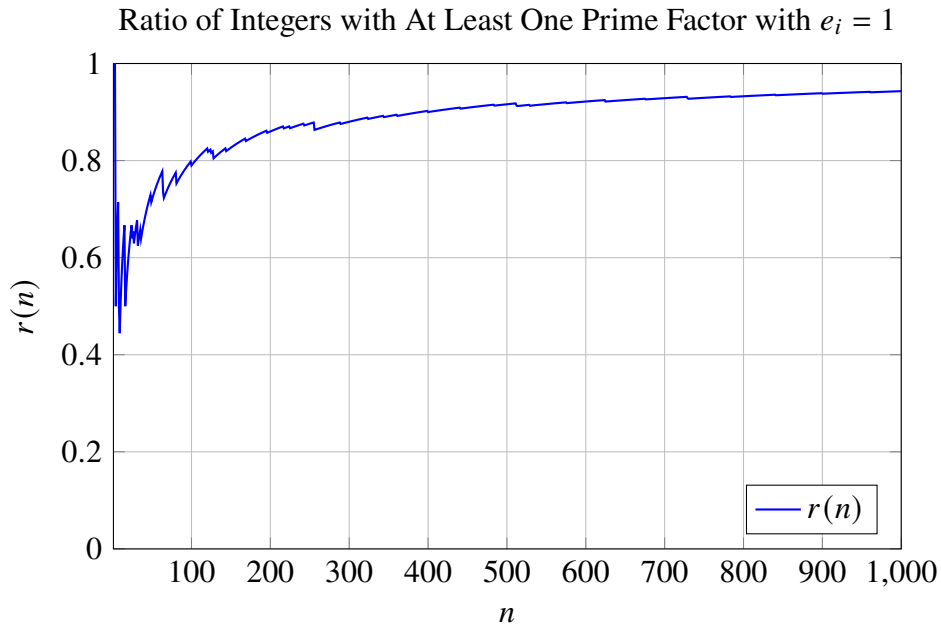
Thus, the ratio of integers with at least one prime factor p_i with $e_i = 1$ is asymptotically bounded by:

$$r(n) \geq 1 - \frac{\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor + \left(\sum_{k=2}^{\log_2(n)} \lfloor n^{1/k} \rfloor \right)^2}{n}$$

□

The evidence so far has shown that, as far as attacks depending on the hull go, q -ary lattices with q being a perfect prime power are seem to not be affected by the hull attack, by the current knowledge.

The graph below gives the ratio of integers with at least one prime factor $p_i^{e_i}$ with $e_i = 1$.



Given the prevalence of integers q for which the condition applies, as shown above, it significantly affects the choice of modulus when dealing with q LIP. How this relates

to parameter selection of other lattice problems used in cryptography, that are also generally based on q -ary lattices, is left as an open problem and further cryptanalysis in this is likely to affect the security of updatable key encryption based on the Learning With Errors (LWE) problem.

4.2 Avoiding Attacks through Non-trivial Hulls

Given that the hull attack relies on the hull being trivial, a question of what happens when the hull is non-trivial would naturally follow. Additionally, we would want to know how to guarantee that the hull of a lattice is non-trivial, and giving such an algorithm. For the case of a p -ary lattice, with p being prime or a prime power, such an algorithm \mathcal{A} exists (personal communication, Russell W. Lai), where we can guarantee a p -ary lattice has a non-trivial hull, by setting its corresponding construction A code to have a non-trivial hull. This is done by moving elements from the dual to the primal, giving us a code that must have a non-trivial hull.

However, the extension of the hull attack to q -ary lattices by Lemma 2 in the previous section causes some problems for a straight forward approach via codes in the case of composite q . Here, due to the fact that the dual code depends on the inner product used, which can vary between \mathbb{Z}_q and \mathbb{Z}_{p_i} , it is possible for $H(C)$ to be non-trivial due to one component having a non-trivial hull while others have trivial hulls.

Since each p_i -component of the CRT-decomposed lattice might have a different p_i -hull, the hull of the original lattice being non-trivial will not affect the effectiveness of the attack in each p_i -component. This would then lead to the requirement of having access to the unique prime decomposition of the modulus of the lattice, in order to construct the lattice with the properties we want. However, due to the small size of moduli used, finding the unique prime decomposition is tractable in this case, without access to quantum computation.

We can therefore use the same algorithm as for p -ary lattices, for the case of any q -ary lattice Λ_q . This can be done by ensuring a trivial hull for all the p -ary component lattices Λ_{p_i} of the original q -ary lattice, such that p_i is a prime factor of q . Thus, were we to for some reason want to use a q -ary lattice for LIP, for a composite q , such a lattices can be constructed efficiently using the same approach.

Next, we briefly go through the effects of ensuring a trivial hull in a code, and the possibility of this resulting in new vulnerabilities on the lattice side, when such a code is used for cryptography.

4.3 Constraints on Codes with Non-trivial Hulls

Here, we note that ensuring a non-trivial hull of a random linear code, will reduce the number of possible linear codes, and discuss the effects this could have on lattice problems in cryptography.

The hull of a code measures the overlap between a code and its dual. Results by Sendrier [Sen97], show that for a random linear code, the probability that the

hull is non-trivial quickly approaches zero as the code length n grows large. More precisely, the expected dimension of the hull of a random $[n, k]_q$ code approaches a small constant as $n \rightarrow \infty$, and for many parameter regimes it remains very close to zero. This implies that almost all large random linear codes have trivial hulls.

4.3.1 Combinatorial Constraints from Enforcing a Non-Trivial Hull

If one wishes to ensure that the hull $\text{Hull}(C)$ is non-trivial, i.e., $\dim(\text{Hull}(C)) = h > 0$, this introduces significant combinatorial constraints on the choice of C . The total number of k -dimensional subspaces of \mathbb{F}_q^n is

$$\binom{n}{k}_q = \prod_{i=0}^{k-1} \frac{q^n - q^i}{q^k - q^i}.$$

Since a random $[n, k]_q$ code is chosen uniformly at random from these subspaces, ensuring a non-trivial hull picks out a much smaller subset:

$$N_h = \#\{C : \dim(C) = k, \dim(\text{Hull}(C)) = h\}.$$

From Sendrier's asymptotic formulas (see [Sen97, Theorem 4.18, Corollary 4.20] and related remarks therein), we know that N_h is negligible compared to $\binom{n}{k}_q$ when $h > 0$. That is:

$$\frac{N_h}{\binom{n}{k}_q} \xrightarrow{n \rightarrow \infty} 0 \quad \text{for any fixed } h > 0.$$

In other words, codes with a prescribed positive hull dimension become extremely rare as n grows. This forces the code designer to sample from a set of measure (in the uniform distribution) asymptotically zero, breaking the initial randomness assumption commonly used in security arguments.

4.3.2 Impact on Construction A Lattices and Cryptographic Security

Construction A lattices, used in various cryptographic schemes (notably certain lattice-based protocols), are derived from linear codes. Suppose $C \subseteq \mathbb{F}_q^n$ is chosen at random and used to construct a lattice $\Lambda_C \subseteq \mathbb{Z}^n$ via a standard embedding (e.g., mapping \mathbb{F}_q to a subset of integers and forming a lattice by adding a scaled version of \mathbb{Z}^n). The security of such lattice-based schemes often relies on the hardness of problems believed to be generic in random lattices.

However, if the code C is not chosen uniformly at random, but rather from a heavily restricted subset (e.g., to ensure that $\text{Hull}(C)$ is non-trivial), then the resulting lattice Λ_C may exhibit additional structure. Such structure could manifest as symmetries, additional linear relations, or statistical biases that an adversary might exploit. Since codes with non-trivial hulls form a very special subset, there is no longer any a priori reason to believe that Λ_C resembles a "generic" random lattice.

In other words, by restricting to codes with non-trivial hulls, we introduce a non-uniform sampling bias. This bias might decrease entropy and potentially introduce

hidden vulnerabilities in cryptographic schemes relying on the hardness of certain lattice problems. Adversaries might leverage these structural properties, leading to potential cryptanalytic attacks that would be less likely if the codes were chosen from the full random ensemble.

4.3.3 Open problem

Sendrier's results show that almost all large random linear codes have trivial hulls, and that forcing a non-trivial hull severely restricts the code space. On the cryptographic side, particularly for constructions that rely on code-derived lattices, such a restriction may introduce additional structure or reduce the entropy of the system. This, in turn, could affect the security assumptions underlying lattice-based cryptographic protocols. While the exact cryptographic implications require further investigation, the combinatorial arguments and asymptotic results suggest that imposing a non-trivial hull is not free of consequences and might weaken the security properties assumed to hold for random instances.

4.4 Extension of q -ary Lattice Transformations

In this section, we present a proof of the existence of transformations between two q -ary lattices that are not signed permutations. This naturally extends the orthogonal transformation space for q -ary lattices.

Theorem 3. *Let $n \in \mathbb{N}$, $d \in \mathbb{N}$, and $q \in \mathbb{N}$ such that $d \mid q$. Define $q' = \frac{q}{d} = d$. Let $M \in \mathbb{Z}^{n \times n}$ be a full-rank integer matrix, that is not a scaled signed permutation matrix, satisfying*

$$M^\top M = d^2 I,$$

where I is the $n \times n$ identity matrix. Define the lattice $\Lambda \subseteq \mathbb{Z}^n$ as

$$\Lambda = \left\{ x \in \mathbb{Z}^n \mid Mx \in d\mathbb{Z}^n \right\}.$$

Then:

1. Λ is a q -ary lattice, i.e.,

$$q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n.$$

2. Λ is q' -ary, i.e.,

$$q'\mathbb{Z}^n \subseteq \Lambda.$$

3. Λ is non-trivial, i.e., $\Lambda \neq \mathbb{Z}^n$ and $\Lambda \neq \emptyset$.

4. The orthogonal matrix $O = \frac{1}{d}M$ maps Λ to another q' -ary lattice $\Lambda' = O\Lambda$, and O is not a signed permutation matrix.

Proof. 1. Λ is a q -ary lattice:

Containment $\Lambda \subseteq \mathbb{Z}^n$: By definition, $\Lambda \subseteq \mathbb{Z}^n$.

Containment $q\mathbb{Z}^n \subseteq \Lambda$: Take any $x \in q\mathbb{Z}^n$, so $x = qy$ for $y \in \mathbb{Z}^n$. Then:

$$Mx = M(qy) = q(My) = d \cdot \left(\frac{q}{d}My\right) = d \cdot q'My.$$

Since $q = d \cdot q'$ and M is an integer matrix, $q'My$ is an integer vector. Therefore, $Mx \in d\mathbb{Z}^n$, which implies $x \in \Lambda$. Hence, $q\mathbb{Z}^n \subseteq \Lambda$.

2. Λ is q' -ary:

Containment $q'\mathbb{Z}^n \subseteq \Lambda$: Take any $x \in q'\mathbb{Z}^n$, so $x = q'y$ for $y \in \mathbb{Z}^n$. Then:

$$Mx = M(q'y) = q'(My).$$

Since $q' = \frac{q}{d} = d$ and M is an integer matrix, $q'(My) \in d\mathbb{Z}^n$. Thus, $x \in \Lambda$, establishing $q'\mathbb{Z}^n \subseteq \Lambda$.

3. Λ is non-trivial:

Assume, for contradiction, that $\Lambda = \mathbb{Z}^n$. Then, for all $x \in \mathbb{Z}^n$, $Mx \in d\mathbb{Z}^n$. Since M is invertible over \mathbb{R} , this would imply $\mathbb{Z}^n = M^{-1}d\mathbb{Z}^n$, leading to $M\mathbb{Z}^n = d\mathbb{Z}^n$. However, M satisfies $M^\top M = d^2I$, which would require M to be a scaled signed permutation matrix to preserve the integer lattice structure. Unless M is a scaled signed permutation matrix, Λ cannot equal \mathbb{Z}^n . Therefore, for non-signed permutation matrices M , Λ is a proper subset of \mathbb{Z}^n , ensuring its non-triviality. Furthermore, since we have $q\mathbb{Z}^n \subseteq \Lambda$, it follows trivially that $\Lambda \neq \emptyset$.

4. Orthogonal Transformation $O = \frac{1}{d}M$:

Define $O = \frac{1}{d}M$. Since $M^\top M = d^2I$, we have:

$$O^\top O = \left(\frac{1}{d}M\right)^\top \left(\frac{1}{d}M\right) = \frac{1}{d^2}M^\top M = \frac{1}{d^2} \cdot d^2I = I.$$

Thus, O is orthogonal.

Define the transformed lattice:

$$\Lambda' = O\Lambda = \{Ox \mid x \in \Lambda\}.$$

Since O is orthogonal, it preserves the inner product and lattice structure. Moreover, Λ' inherits the q' -ary property:

$$q\mathbb{Z}^n \subseteq \Lambda \quad \Rightarrow \quad O(q\mathbb{Z}^n) = q'\mathbb{Z}^n \subseteq \Lambda'.$$

Therefore, Λ' is also a q' -ary lattice contained within \mathbb{Z}^n .

Non-signed Permutation Nature of O : A signed permutation matrix has exactly one non-zero entry (either +1 or -1) in each row and column, with all other entries being zero. Given that M is not a scaled signed permutation matrix, O inherits non-zero

entries in multiple positions per row and column, ensuring that O is not a signed permutation matrix.

Thus, O serves as an orthogonal transformation mapping the q -ary lattice Λ , that is also q' -ary, to another q' -ary lattice Λ' , without being a signed permutation matrix. This establishes the existence of non-trivial q' -ary lattices that map to some other q' -ary lattice by an orthogonal transformation that is not a signed permutation. \square

This only shows the existence of such transformation, but the construction and limitations of these transformations are not made clear in the previous proof. The structure would suggest that the denominators in O do not need to be the same, but that the possible denominators are related through the divisibility relations between them. Next, we give a concrete example of such a transformation in two dimensions.

4.5 Explicit Two-Dimensional Example

To concretely illustrate the theoretical construction, consider the following explicit example in two dimensions.

Example 1. Let:

$$n = 2, \quad d = 5, \quad q = 25, \quad q' = \frac{q}{d} = 5.$$

Define the integer matrix:

$$M = \begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix}.$$

Verification of $M^\top M = d^2 I$:

Compute $M^\top M$:

$$M^\top M = \begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix} = \begin{pmatrix} 4 \times 4 + 3 \times 3 & 4 \times 3 + 3 \times (-4) \\ 3 \times 4 + (-4) \times 3 & 3 \times 3 + (-4) \times (-4) \end{pmatrix} = \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix} = 25I.$$

Thus, M satisfies $M^\top M = d^2 I$.

Construction of Λ :

Define the lattice Λ as:

$$\Lambda = \left\{ x \in \mathbb{Z}^2 \mid Mx \in 5\mathbb{Z}^2 \right\}.$$

For $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{Z}^2$, the condition $Mx \in 5\mathbb{Z}^2$ translates to:

$$Mx = \begin{pmatrix} 4x_1 + 3x_2 \\ 3x_1 - 4x_2 \end{pmatrix} \in 5\mathbb{Z}^2.$$

This implies the system of congruences:

$$\begin{cases} 4x_1 + 3x_2 \equiv 0 \pmod{5}, \\ 3x_1 - 4x_2 \equiv 0 \pmod{5}. \end{cases}$$

Solving the Congruences:

Second Congruence:

$$3x_1 - 4x_2 \equiv 0 \pmod{5} \Rightarrow 3x_1 \equiv 4x_2 \pmod{5}.$$

Multiply both sides by the modular inverse of 3 modulo 5, which is 2:

$$x_1 \equiv 2 \times 4x_2 = 8x_2 \equiv 3x_2 \pmod{5}.$$

Thus:

$$x_1 = 3a + 5k, \quad a, k \in \mathbb{Z}.$$

Substituting into the First Congruence:

$$4(3a + 5k) + 3x_2 = 12a + 20k + 3a = 15a + 20k \equiv 0 \pmod{5}.$$

Simplifying:

$$15a + 20k \equiv 0 \pmod{5} \Rightarrow 0 \equiv 0 \pmod{5},$$

which holds for all $a, k \in \mathbb{Z}$. Therefore, the general solution is:

$$x_1 = 3a + 5k, \quad x_2 = a, \quad a, k \in \mathbb{Z}.$$

Parametrization of Λ :

Any vector $x \in \Lambda$ can be expressed as:

$$x = a \begin{pmatrix} 3 \\ 1 \end{pmatrix} + k \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \quad a, k \in \mathbb{Z}.$$

Thus, a basis for Λ is:

$$b_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 5 \\ 0 \end{pmatrix}.$$

Verification of Λ Being q -ary and q' -ary:

By construction, Λ contains $q'\mathbb{Z}^2 = 5\mathbb{Z}^2$ since:

For any $x = \begin{pmatrix} 5y \\ 5z \end{pmatrix} \in 5\mathbb{Z}^2$, choose $a = 5z$ and $k = y - 3z$. Then:

$$x = 5z \begin{pmatrix} 3 \\ 1 \end{pmatrix} + (y - 3z) \begin{pmatrix} 5 \\ 0 \end{pmatrix} = \begin{pmatrix} 15z + 5y - 15z \\ 5z \end{pmatrix} = \begin{pmatrix} 5y \\ 5z \end{pmatrix} \in \Lambda.$$

Thus, we have:

$$25\mathbb{Z}^2 \subseteq 5\mathbb{Z}^2 \subseteq \Lambda \subseteq \mathbb{Z}^2.$$

Verification of Non-Triviality:

The determinant of the basis matrix $B = \begin{pmatrix} 3 & 5 \\ 1 & 0 \end{pmatrix}$ is:

$$\det(B) = 3 \times 0 - 5 \times 1 = -5.$$

Taking the absolute value, $|\det(B)| = 5$, which implies that the index $[\mathbb{Z}^2 : \Lambda] = 5$. Therefore, Λ is a proper sublattice of \mathbb{Z}^2 , confirming its non-triviality.

Orthogonal Transformation O :

Define the orthogonal matrix:

$$O = \frac{1}{5}M = \frac{1}{5} \begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix} = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix}.$$

Verification of Orthogonality:

$$\begin{aligned} O^T O &= \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix} \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix} = \begin{pmatrix} \left(\frac{4}{5}\right)^2 + \left(\frac{3}{5}\right)^2 & \frac{4}{5} \times \frac{3}{5} + \frac{3}{5} \times \left(-\frac{4}{5}\right) \\ \frac{3}{5} \times \frac{4}{5} + \left(-\frac{4}{5}\right) \times \frac{3}{5} & \left(\frac{3}{5}\right)^2 + \left(-\frac{4}{5}\right)^2 \end{pmatrix} \\ &= \begin{pmatrix} \frac{16}{25} + \frac{9}{25} & \frac{12}{25} - \frac{12}{25} \\ \frac{12}{25} - \frac{12}{25} & \frac{9}{25} + \frac{16}{25} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I. \end{aligned}$$

Thus, O is indeed orthogonal.

Construction of Λ' :

Define:

$$\Lambda' = O\Lambda = \left\{ Ox \mid x \in \Lambda \right\}.$$

For any $x = ab_1 + kb_2 \in \Lambda$, where $a, k \in \mathbb{Z}$, we have:

$$Ox = aOb_1 + kOb_2.$$

Compute each term:

$$Ob_1 = O \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix} \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{12}{5} + \frac{3}{5} \\ \frac{9}{5} - \frac{4}{5} \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix},$$

$$Ob_2 = O \begin{pmatrix} 5 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix} \begin{pmatrix} 5 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}.$$

Thus:

$$\Lambda' = \left\{ a \begin{pmatrix} 3 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ 3 \end{pmatrix} \mid a, k \in \mathbb{Z} \right\}.$$

Verification of Λ' Being q' -ary:

Containment $q'\mathbb{Z}^2 \subseteq \Lambda'$: Take any $x = \begin{pmatrix} 5k \\ 5m \end{pmatrix} \in 5\mathbb{Z}^2$, where $k, m \in \mathbb{Z}$. We aim to express x as a linear combination of the basis vectors of Λ' :

$$x = a \begin{pmatrix} 3 \\ 1 \end{pmatrix} + b \begin{pmatrix} 4 \\ 3 \end{pmatrix}.$$

This leads to the system:

$$\begin{cases} 3a + 4b = 5k, \\ 1a + 3b = 5m. \end{cases}$$

Solving the second equation for a :

$$a = 5m - 3b.$$

Substituting into the first equation:

$$3(5m - 3b) + 4b = 15m - 9b + 4b = 15m - 5b = 5k \quad \Rightarrow \quad 3m - b = k \quad \Rightarrow \quad b = 3m - k.$$

Thus:

$$a = 5m - 3(3m - k) = 5m - 9m + 3k = -4m + 3k.$$

Therefore, for any $x = \begin{pmatrix} 5k \\ 5m \end{pmatrix} \in 5\mathbb{Z}^2$, choosing $a = -4m + 3k$ and $b = 3m - k$ yields:

$$x = (-4m + 3k) \begin{pmatrix} 3 \\ 1 \end{pmatrix} + (3m - k) \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \begin{pmatrix} -12m + 9k + 12m - 4k \\ -4m + 3k + 9m - 3k \end{pmatrix} = \begin{pmatrix} 5k \\ 5m \end{pmatrix}.$$

Thus, $5\mathbb{Z}^2 \subseteq \Lambda'$.

Containment $\Lambda' \subseteq \mathbb{Z}^2$: By construction, Λ' is generated by integer linear combinations of integer vectors, ensuring $\Lambda' \subseteq \mathbb{Z}^2$.

Verification of Non-Triviality:

The determinant of the basis matrix $B' = \begin{pmatrix} 3 & 4 \\ 1 & 3 \end{pmatrix}$ is:

$$\det(B') = 3 \times 3 - 4 \times 1 = 9 - 4 = 5.$$

Taking the absolute value, $|\det(B')| = 5$, which implies that the index $[\mathbb{Z}^2 : \Lambda'] = 5$. Therefore, Λ' is a proper sublattice of \mathbb{Z}^2 , confirming its non-triviality.

Orthogonal Transformation O is Not a Signed Permutation Matrix:

A signed permutation matrix has exactly one non-zero entry (either $+1$ or -1) in each row and column, with all other entries being zero.

$$O = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{4}{5} \end{pmatrix},$$

From above it is evident that none of the entries are in $\{0, \pm 1\}$, and multiple non-zero entries exist in each row and column. Therefore, O is not a signed permutation matrix.

The lattice Λ' satisfies:

$$5\mathbb{Z}^2 \subseteq \Lambda' \subseteq \mathbb{Z}^2,$$

making it a q' -ary lattice with $q' = 5$. Additionally, Λ' is a proper subset of \mathbb{Z}^2 with index 5, ensuring its non-triviality. The orthogonal matrix O is not a signed permutation matrix, as its entries are neither confined to $\{0, \pm 1\}$ nor do they form a single non-zero entry per row and column.

We have constructed a non-trivial q -ary lattice $\Lambda \subseteq \mathbb{Z}^n$ defined by the linear constraint $Mx \in d\mathbb{Z}^n$, where M is a full-rank integer matrix satisfying $M^\top M = d^2 I$. The orthogonal transformation $O = \frac{1}{d}M$ successfully maps Λ to another q' -ary lattice $\Lambda' \subseteq \mathbb{Z}^n$, while O itself is verified to be non-signed permutation.

4.6 General Rational Orthogonal Transformations

Above, we have given a proof of the existence of orthogonal transformations that are not signed permutations, and an example of such a transformation. The example we gave is quite limited, and a more general construction of such transformations would be needed for practical use. From the proof of existence, it naturally follows that rational orthogonal transformations, like the example given, extends to rational orthogonal transformation matrices with distinct denominators $d_1, d_2, d_3, \dots, d_{nm}$ for dimensions $n, m \in \mathbb{N}$.

Furthermore, due to the inclusion $q\mathbb{Z}^2 \subseteq \Lambda$, we can limit the possible denominators to $d \in \mathbb{N}$ such that $d \mid q$. The problem of finding a general framework and method of constructing such transformations is left to further research. Similarly, how much the existence of rational orthogonal transformations extends the transformation space for q -ary LIP remains unclear.

5 Estimating Concrete Hardness of LIP

In implementing software to estimate the security of lattice schemes, we care about estimating the complexity of attacks on problems with a specific set of parameters, to give an idea of the security of a scheme instantiated with these parameters. This is why we are mainly interested in the average case run-time of the algorithms solving the problems in question, as is the general case of concrete estimates.

In this section, we document the implementation of the Lattice Estimator module for LIP and PEP. The main concern here is the LIP estimation, which in some attacks such as the hull attack may use codes to solve the original problem. The PEP estimation is in this sense auxiliary, and only handled in this work in order to support the main task of estimating the security of LIP.

5.1 The Lattice Estimator

The Lattice Estimator [APS15] is maintained by Martin Albrecht, which is used to estimate the security for underlying lattice problems, given some specific parameters of interest. This allows a designer of cryptographic schemes to find parameters for which the problem in question is believed to be secure. The need for such software is quite evident with the increase in interest around lattice-based cryptography, and its standardization efforts, giving the designers of cryptographic schemes based on lattices an easy way to estimate the security of their schemes for some specific parameters.

In the case of LIP, the relevant parameters that we estimate the security on have unique invariants on each instance, and therefore the user has the option to provide two quadratic forms, or in the case of q -ary lattices, two generator matrices and associated modulo q . This trivially also include the standard lattice parameters such as dimension, as they are efficiently computable from each quadratic form, generator matrix. Further developments in identifying the relations of classes of lattices and their corresponding quadratic forms would allow for a more general estimation procedure.

The Lattice Estimator is mainly focused on lattices, and as of 2024 provides estimations for problems such as LWE, SIS, and NTRU. The estimations for these problems are based on attacks on the problems or some version of the problems. Here, we describe the extension of the Lattice Estimator to the case of LIP, which due to the nature of the available attacks also requires some security estimations being implemented for code problems as well. We begin by describing the parameters available to base the estimation on for the user.

5.2 LIP parameters

The primary parameter are the actual quadratic forms of a specific instance, which we define using the available Sage Math modules, which in the example below we assume are imported.

Below is an example of how such a quadratic form is instantiated, as is expected by the LIP module.

Suppose we start with a lattice basis B , represented as an integer matrix:

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix}.$$

The Gram matrix Q of the lattice is defined by:

$$Q = B^T B.$$

With SageMath, we can proceed as follows:

Step 1: Define the basis matrix B over the integers

```
>>> B = matrix(ZZ, [[b11, b12, ...],
                    [b21, b22, ...],
                    ...])
```

Step 2: Compute the Gram matrix $Q = B^T * B$

```
>>> Gram = B.transpose() * B
```

Step 3: Create the quadratic form from Q

```
>>> Q = QuadraticForm(ZZ, Gram)
```

This constructs a quadratic form Q from the given lattice basis B , in the form expected by the lattice estimator LIP module. In the case of construction A lattices, the modulus q is given as an integer, and the generator matrix is given as a matrix, defined exactly as the basis in the example above.

Define the generator matrix A over the integers as follows

```
>>> A = matrix(ZZ, [[b11, b12, ...],
                    [b21, b22, ...],
                    ...])
```

5.3 PEP estimator module

In order to estimate the complexity of the Hull Attack, we need to expand the Lattice Estimator beyond lattices, and include estimations on some problems for codes. In this section, we document the implemented functions and algorithms used in the estimation of the PEP/PCE module, that are used by the LIP module. Each algorithm described has an associated function implemented in the PCE module that estimates the hardness of solving PEP/PCE using said algorithm.

In this module there is a general interface that is for the user to interact with and estimate the complexity of a problem, some attacks on the problem that are the basis for the security estimation, and a parameter object that is used to give the specific parameters to be used for estimation. Below we detail each of these parts of the module as well as justify the choices made in the implementation.

5.3.1 Information Set Decoding (ISD)

This module functions independently as a method of estimating the complexity of the Information Set Decoding (ISD) problem, but is also used by some of the code attacks. While there exist many algorithms for solving ISD, the estimation for the complexity of solving ISD in this module is based on the algorithm by Lee-Brickell. The choice to use the Lee-Brickell algorithm is in large part due to it having a solid foundation of analysis and having been covered thoroughly since its inception. The algorithm, based on given parameters, tries to find the low-weight code words. Below, we give a brief overview of the algorithm, for purposes of illustrating the complexity of the algorithm as defined in [LB88].

The Lee-Brickell algorithm, introduced by Peter Lee and Ernest F. Brickell [LB88], is an efficient method for decoding linear error-correcting codes, particularly in the context of attacking the McEliece cryptosystem.

The Lee-Brickell algorithm improves upon previous information set decoding techniques by employing a probabilistic approach that reduces the expected number of operations required for decoding. The key idea is to find an information set—a set of coordinates in the codeword that are linearly independent and can be used to reconstruct the original message—and then perform Gaussian elimination to decode the message.

Steps of the Algorithm

1. Randomly select a subset of coordinates, known as the information set, from the received vector.
2. Apply Gaussian elimination to the submatrix formed by the columns corresponding to the information set. If the submatrix is invertible, proceed to the next step; otherwise, go back to step 1 and select a new information set.
3. Identify the error pattern by solving the linear system formed by the parity-check equations.
4. Use the identified error pattern to correct the received vector and retrieve the original message.

The efficiency of the Lee-Brickell algorithm lies in its probabilistic nature, which significantly reduces the number of trials needed to find a suitable information set compared to earlier methods. This makes it one of the most effective algorithms for information set decoding, especially for codes with high error-correcting capabilities.

The Lee-Brickell algorithm represents a crucial advancement in the field of error-correcting codes and cryptanalysis. Its probabilistic approach to finding an information set has paved the way for more efficient decoding methods and has quite important implications for the security of cryptographic systems based on linear codes.

For estimating the run-time of the algorithm, we have taken into account the parameters for the algorithm as follows:

- q : Modulus of the finite field.

- n : The length of the codeword.
- k : The dimension of the code.
- q : The target weight of the codewords.
- N_w : The estimated cardinality of set of codewords of weight w , up to scalar multiplication.
- L : The number of codewords to find.

Note that the algorithm solves ISD and not PCE, and thus these parameters are not based on the `PCEParameters` object used when interfacing with the attacks on PCE, even though they originate there in this case, when used as a sub-procedure for solving PCE.

Below we base our estimation of difficulty of solving ISD using Lee-Brickell on the estimations of Beullens [Beu20]. In the same vein as we are doing here, their analysis of Lee-Brickell is with the goal of solving PCE in mind. To this end, their analysis is the one that is most appropriate for us to use, considering our goals. We note that our implementation uses the N_w as a parameter here, although it can in fact also be computed from the rest of the parameters inside the Lee-Brickell estimator function. This is because the quantity N_w is used in other places, and we want to avoid computing it multiple times, so we do it before calling the Lee-Brickell estimator.

5.3.2 Leons' Algorithm

The algorithm given by Leon [Leo82] solves PCE by using the ISD algorithm we detailed above as a subroutine. It solves the permutation code equivalence problem for two codes C_1 and C_2 by leveraging invariants of codewords and finding permutations between subsets of codewords that span each code. The approach relies on geometric invariants such as weight, which are preserved under permutation. Specifically, if π is the permutation mapping C_1 to C_2 , the invariant property ensures that $\omega(x) = \omega(\pi(x))$ for all $x \in C_1$, where $\omega(x)$ is the weight of x .

The main steps of Leon's algorithm are as follows:

1. Compute the sets $B_w^{C_i} = B_w(C_i)$ for $i = 1, 2$, where $B_w(C_i)$ denotes the set of all codewords in C_i with Hamming weight w .
2. Determine the group generators of $S(B_w^{C_1}, B_w^{C_2})$, which is the subgroup of permutations $\sigma \in S_n$ such that $\sigma(B_w^{C_1}) = B_w^{C_2}$.
3. Verify whether $\sigma(C_1) = C_2$ for $\sigma \in S(B_w^{C_1}, B_w^{C_2})$.

A key insight is that the parameter w (the target weight) must be chosen carefully. If w is too small, the subgroup $S(B_w^{C_1}, B_w^{C_2})$ becomes too large to efficiently check all permutations. If w is too large, the sets $B_w^{C_1}$ and $B_w^{C_2}$ may become empty or prohibitively large to compute. To address this, w is typically chosen slightly larger than the minimum distance of the code.

Leon's algorithm operates efficiently when w is chosen properly. The complexity of the final two steps is polynomial in the size of $B_w^{C_i}$, which can be approximated as $(q-1)N_w$, where N_w is the number of codewords of weight w . To limit N_w , w should be small enough to avoid excessive computations but large enough to provide discriminant information.

The runtime is further influenced by the complexity of searching for codewords of weight w in the code. This search is often performed using an information-set decoding (ISD) algorithm, and therefore we make the assumption the algorithm used is the algorithm by Lee-Brickell detailed above, whose complexity is denoted by $C_{\text{ISD}}(q, n, k, w)$. The total complexity of Leon's algorithm is given by:

$$O\left(C_{\text{ISD}}(q, n, k, w) \cdot \sum_{i=1}^{N_w} \frac{1}{i}\right),$$

where the summation accounts for the verification of permutations. For practical purposes, the summation can be estimated as:

$$O(C_{\text{ISD}}(q, n, k, w) \cdot \ln(N_w)),$$

assuming $N_w \geq 2$.

Leon's algorithm effectively reduces the PCE problem to manageable computations by focusing on subsets of codewords of a specific weight. Its efficiency is highly dependent on the choice of w and the complexity of the underlying ISD algorithm. While the algorithm is practical for codes of moderate length and dimension, its performance deteriorates for larger instances due to the rapid growth of N_w . Due to this, the algorithm is of less importance in the context of solving lattice problems via code problems. Still, the estimation of using this to attack LIP is included in the module for PCE, for completeness.

5.3.3 Beullens'

Beullens' algorithm provides an improvement over Leon's algorithm for solving the permutation code equivalence (PCE) problem, particularly when the size of the finite field q is large. The algorithm leverages the concept of preserving the multiset of entries of codewords under permutations and avoids computing all minimal-weight codewords, which is a computational bottleneck in Leon's approach.

The algorithm solves the permutation equivalence problem, which asks whether two codes C_1 and C_2 are related by a permutation π . Its steps are as follows:

1. **Choose a Target Weight w :** Select w to maximize discriminant power while ensuring efficient computation. The value of w must satisfy:

$$\frac{n!}{(n-w)!q^{n-k}} < \frac{1}{4 \ln n},$$

where n is the code length, k is the code dimension, and q is the field size.

2. **Sample Codewords:** Use information set decoding (ISD) to generate a list L of size:

$$|L| = \sqrt{|B_n(w)|} \cdot q^{-n+k-1} \cdot 2 \ln n,$$

where $B_n(w)$ is the set of all vectors of weight w . Each entry in L is of the form $(x, \text{lex}(x))$, where $\text{lex}(x)$ represents the lexicographically smallest vector in the orbit of x under permutations.

3. **Identify Permutations:** For each sampled codeword $y \in C_2$ with weight w , check whether $\text{lex}(y)$ matches any entry in L . If so, append the pair (x, y) to a list P . Continue until P contains at least $2 \ln n$ such pairs.
4. **Recover the Permutation:** Use a backtracking algorithm to recover a permutation π such that $\pi(x) = y$ for all pairs $(x, y) \in P$. Verify that $\pi(C_1) = C_2$.

The runtime of Beullens' algorithm is dominated by the cost of generating codewords using ISD and verifying pairs. The key complexity factors are:

- **Sampling Codewords:** The ISD cost of generating the list L and matching entries is:

$$O\left(\frac{|B_n(w)|}{q^{n-k}} \cdot \ln n\right),$$

where $B_n(w)$ is the set of vectors of weight w .

- **Lexicographic Computations:** Computing $\text{lex}(x)$ for each sampled vector involves $O(k^2)$ row operations for Gaussian elimination and $O(k)$ operations to minimize entries. The total lex computation complexity for all sampled vectors is:

$$O(q \cdot |L|).$$

- **Permutation Recovery:** The backtracking step iterates over candidate permutations and typically completes in polynomial time for sufficiently large q , due to the high discriminant power of the algorithm.

The computational complexity of Beullens' algorithm is dominated by the cost of generating the list L of low-weight codewords using an ISD algorithm. This cost is:

$$2C_{|L|}(q, n, k, w) \approx C_\infty(q, n, k, w) \cdot \frac{|L|}{N_w},$$

where:

- $C_\infty(q, n, k, w)$ is the ISD complexity for finding a single weight- w codeword.
- $|L|$ is the size of the list L , and $N_w = |B_w^{C_1}|$ is the total number of weight- w codewords in C_1 .

Heuristically, the algorithm's runtime is dominated by the ISD search effort, which scales as:

$$O\left(\sqrt{|B_w^{C_1}|} \cdot q^{-n+k} \cdot \ln n\right).$$

Beullens' algorithm significantly outperforms Leon's algorithm for large field sizes q . While Leon's algorithm computes all minimal-weight codewords, Beullens' approach only samples a small fraction of codewords with non-minimal weights. This yields a substantial improvement in runtime, especially for large q .

Next we describe the SSA and BOS algorithms, both of which are specifically designed to exploit the properties of codes with small or trivial hulls, enabling them to achieve polynomial-time complexity for such instances. In contrast, Leon's and Beullens' algorithms, while effective, do not inherently leverage the trivial hull property as a core component of their strategy. Consequently, these algorithms rely on sampling and computational heuristics, which can introduce additional overhead. For codes with small or trivial hulls, SSA and BOS achieve superior performance by leveraging discriminant signatures (SSA) or graph isomorphism (BOS), both of which are computationally efficient and more targeted for these cases. This distinction makes SSA and BOS the preferred choices for solving the permutation equivalence problem in the context of trivial hulls.

Thus, the algorithms detailed in the following sections are the algorithms that are going to be the ones that make efficiently attacking LIP via code problems possible, and therefore the ones that are going to be of greatest interest for estimating LIP security.

5.3.4 Support Splitting Algorithm

A different approach to the one used in Leon and Beullen algorithms is used by the Support Splitting Algorithm (SSA), given by Sendrier [Sen00]. The SSA is a polynomial-time algorithm designed to determine the permutation between two permutation-equivalent linear codes. The algorithm is based on the concept of a *signature function*, which maps each coordinate of a code to an invariant property that remains unchanged under coordinate permutations. If the signature function is fully discriminant, meaning it uniquely identifies every coordinate of the code, the permutation can be directly recovered by matching the signatures of the two codes.

In the SSA, the signature function is defined as:

$$\mathcal{S}(C, i) = \{\omega(H(C^i)), \omega(H((C^\perp)^i))\},$$

where C^i is the code obtained by puncturing C at position i , C^\perp is the dual code of C , $H(\cdot)$ represents the hull of a code, and $\omega(\cdot)$ denotes the weight enumerator, which is defined as:

$$\omega(C)(x, y) = \sum_{w=0}^n N_w x^w y^{n-w}.$$

Here, N_w is the number of codewords in C with Hamming weight w .

The hull of a code, defined as the intersection of a code with its dual, plays a crucial role in reducing the computational complexity. As mentioned above, for random $[n, k]$ -linear codes over \mathbb{F}_q , the hull dimension h is typically small. This ensures that the cost of computing the hull and its weight enumerator remains manageable, making the algorithm practical for large code lengths.

The complexity of the SSA depends on both the length of the code n and the dimension of its hull h . Key computational steps include:

- Computing the hull of the code, which requires $O(n^3)$ operations in \mathbb{F}_q .
- Computing the weight enumerator of the hull, which has a complexity of $O(nq^h)$.

For random codes, heuristic observations suggest that using approximately $\ln(n)$ punctures is sufficient to construct a fully discriminant signature. Combining these steps, the overall complexity of SSA for recovering the permutation π between two equivalent codes C_1 and $C_2 = \pi(C_1)$ is estimated as:

$$O(n^3 + n^2q^h \ln(n)),$$

where $h = \min\{k, n - k\}$ is the dimension of the hull. This bound reflects the exponential dependence on h , which is typically small for random codes, making SSA efficient in most practical cases.

The Support Splitting Algorithm provides a polynomial-time solution for permutation equivalence problems in codes with small hull dimensions. Its efficiency hinges on the ability to compute and refine signatures using invariants such as the weight enumerator of the hull. While its complexity grows exponentially with the hull dimension, this dependency is mitigated for random codes, where the hull dimension is usually small. Therefore, SSA is a practical tool for code equivalence problems, especially in scenarios involving large code lengths and low-dimensional hulls. In our case, we are interested in the solving of PCE in the cases of construction A lattices, and given the lattice estimator parameters for a construction A lattice, we can efficiently estimate the hardness of solving the LIP problem via the PCE problem.

5.3.5 BOS Algorithm

The BOS algorithm, named after its authors Bardet, Otmani, and Saeed-Taha, is a deterministic method designed to solve the permutation code equivalence problem for linear codes. The BOS algorithm builds on a reduction of the code equivalence problem to the graph isomorphism problem, specifically leveraging the trivial hull property of certain codes to simplify the problem.

Where the SSA fails to perform on codes with trivial hulls, the BOS algorithm operates under the assumption that the codes A and B have trivial hulls, i.e., the intersection of the code with its orthogonal complement is $\{0\}$. This property allows the vector space \mathbb{F}_q^n to be expressed as the direct sum of the code and its dual:

$$\mathbb{F}_q^n = A \oplus A^\perp.$$

Using this property, the algorithm constructs symmetric adjacency matrices Σ_A and Σ_B representing graphs associated with the codes. These matrices are defined as:

$$\Sigma_A = G_A^T (G_A G_A^T)^{-1} G_A, \quad \Sigma_{A^\perp} = H_A^T (H_A H_A^T)^{-1} H_A,$$

where G_A is a generator matrix of A and H_A is its parity-check matrix.

The key idea of the BOS algorithm is to use the adjacency matrix Σ_A to construct a weighted graph G_A , where the matrix encodes the weights of edges between vertices. The algorithm then reduces the code equivalence problem to testing whether the graphs G_A and G_B are isomorphic. This reduction relies on the following theorem:

Theorem 4. [Sen00, Theorem 5] *Let A and B be two linear codes of length n over a field \mathbb{F} , both having a trivial hull. Then:*

$$B \sim_\pi A \quad \text{if and only if} \quad G_B \sim_\pi G_A,$$

where G_A and G_B are the weighted graphs associated with A and B , respectively, defined by their adjacency matrices Σ_A and Σ_B . The adjacency matrix for a code A is given by:

$$\Sigma_A = G_A^T (G_A G_A^T)^{-1} G_A,$$

where G_A is a generator matrix of A .

This reduction allows the BOS algorithm to leverage existing graph isomorphism solvers, such as Nauty, for the equivalence decision.

The time complexity of the BOS algorithm is primarily determined by two steps:

- Computing the adjacency matrices Σ_A and Σ_B involves matrix inversion and multiplication, with a complexity of $O(n^\omega)$, where ω is the exponent of matrix multiplication (currently $\omega < 2.373$).
- The complexity of testing isomorphism between two graphs with n vertices, denoted as $GI(n)$, depends on the chosen graph isomorphism algorithm.

For codes with non-trivial hulls, the algorithm must preprocess the codes to produce subcodes with trivial hulls. This preprocessing increases the overall complexity. For such cases, the total complexity is given by:

$$O(hn^{\omega+h+1} GI(n)),$$

where h is the dimension of the hull.

Experimental results show that for random binary codes with trivial hulls, the BOS algorithm can solve instances with lengths up to 50,000 in under a few seconds using practical graph isomorphism solvers. These results highlight the algorithm's efficiency for large codes, provided the hulls are trivial.

5.4 LIP estimator module

This section details the functioning of the LIP module of the lattice estimator, as well as reasoning for the choices made when implementing the module. The module covers two attacks on LIP. Firstly, the lattice reduction based attack by Haviv and Regev, which functions as a baseline, since it is based on lattice reduction. Secondly, the hull attack by Ducas and Gibbons which solves LIP using the connection to linear codes.

Like in the case of codes, the `LIPParameters` is included in the LIP module and should be accessed through the module itself. The actual LIP estimation module then contains all of the tools needed to estimate the concrete hardness of LIP on a lattice.

The LIP module, which acts as the interface, contains two methods that can be called through the LIP interface by the user for estimating hardness; `rough`, and `__call__`. The `rough` method provides a rough estimation of the hardness of the problems, based on some subset of attacks, while the `__call__` method runs all of the available algorithms and give a complete picture of the concrete hardness. The different attacks against LIP are separate modules that are imported and included in a list of algorithms, which are run. Each algorithm returns a cost dictionary which is then included in the final list of results returned to the user.

The algorithms included in the `rough` method is not determined by the user, but is defined in the interface itself. Then, since most of the best attacks for lattice problems like LIP tend to be relying of finding short vectors, it would be ideal to have such an attack included in the list of attacks run by the `rough` method. This gives an estimate of the hardness that is relatable to other lattice problems in a general sense. Currently there are only two attacks implemented against LIP, so the `rough` and `__call__` methods have the same list of algorithms which include both the hull attack, and the lattice reduction based attack by Haviv and Regev.

5.4.1 Haviv and Regevs attack

As noted in section 3.1, Haviv and Regev proposed an algorithm for solving LIP that runs in $n^{O(n)}$ time, where n is the rank of the lattices. The algorithm is based on identifying linearly independent shortest vectors and recursively reducing the problem to sublattices, leveraging structural properties of the input lattices. This attack is mainly included as a baseline attack, as it will be dominated by the lattice reduction complexity, and therefore can be considered not very efficient.

The implementation follows the theoretical work, providing tools to estimate the concrete complexity of the attack. This implementation relies on modeling the computational cost of lattice reduction, particularly the BKZ algorithm, using a specific block size β .

The algorithm proceeds through several structured steps:

1. **Identifying Shortest Vectors:** Compute the set of all shortest nonzero vectors in L_1 and L_2 . These vectors are candidates for generating a mapping between the two lattices.

2. **Isolation of Linearly Independent Vectors:** Use a generalized isolation lemma to identify n linearly independent vectors from the set of shortest vectors. These vectors serve as a basis for L_1 and L_2 , enabling further analysis.
3. **Mapping and Orthogonal Transformations:** For every possible mapping of basis vectors in L_1 to L_2 , construct an orthogonal transformation O . Verify whether O maps the entire lattice L_1 to L_2 .
4. **Recursive Problem Reduction:** If the identified basis does not span the full lattice, project the lattices onto the orthogonal complement of the span of these vectors and recursively solve the reduced problem.

The algorithm's recursive structure ensures that the problem dimension decreases in each step, leading to a total complexity of $n^{O(n)}$. The implementation estimates the concrete complexity of Haviv and Regev's algorithm by modeling the cost of lattice reduction using BKZ with block size β . Key components include:

The core function `costf` computes the cost of BKZ reduction, parameterized by the block size β . The cost model incorporates the following:

- **Root-Hermite Factor δ :** The quality of the reduced basis is measured by the root-Hermite factor, which depends on β . The function `delta` calculates δ as:

$$\delta = \beta / (2\pi e) \cdot (\pi\beta)^{1/\beta}.$$

- **Dimension d :** The lattice dimension directly impacts the complexity of reduction. For LIP, the dimension d is typically $2n$ to account for both input lattices.
- **Reduction Cost:** The cost of BKZ reduction scales exponentially with β , with specific models such as MATZOV or ABLR21 determining the precise growth rate.

The recursive nature of the algorithm is captured by splitting the lattice into smaller subproblems at each step. The implementation tracks the accumulated cost of reductions and other operations, with the total cost being a summation of these contributions across all recursive calls.

The implementation enables experimentation with key parameters, including:

- **beta:** Determines the quality of BKZ reduction and the associated computational overhead.
- **dimension:** Affects the number of recursive calls and the complexity of each reduction step.
- **cost models:** Various models (MATZOV, ABLR21, etc.) provide flexibility for analyzing classical and quantum reduction scenarios.

The complexity of Haviv and Regev's algorithm depends on two main factors:

- **Reduction Complexity:** The dominant cost arises from performing BKZ reductions at each recursion level. With block size β , the cost is estimated as $2^{c\beta}$, where c is a constant determined by the cost model.
- **Recursive Structure:** The number of recursive calls depends on the dimension reduction achieved in each step. The total complexity is bounded by $n^{O(n)}$, as each recursive step decreases the dimension by at least one.

The implementation’s cost estimates align with the theoretical predictions, making it a valuable and reliable tool for analyzing the practical feasibility of the attack under various parameter settings.

Haviv and Regev’s algorithm provides a baseline method for solving LIP with theoretical guarantees of $n^{O(n)}$ complexity. The implementation bridges the gap between theory and practice by modeling the cost of lattice reduction, allowing for precise complexity estimation. This combination of theoretical and practical tools offers deeper insights into the algorithm’s performance for lattices of varying parameters. While there has yet to be any practical implementation of the actual algorithm to attack LIP, the estimation of the concrete security for some given parameters give a good lower bound on concrete security for cryptographic scheme designers.

5.4.2 Hull Attack

The Hull Attack on a LIP instance leverages the trivial hull property of certain lattices to reduce the problem to a simpler Permutation Code Equivalence problem. The computational cost of the Hull Attack is divided into two primary components: estimating the cost of BKZ lattice reduction on two \mathbb{Z} LIP instances, and solving the PCE problem using code equivalence techniques. The intermediate representation transitions are computationally negligible and thus excluded from the estimation.

Viability of the Hull Attack. The attack’s feasibility depends on the modulus q of the q -ary lattice under consideration. The Hull Attack is restricted to lattices that are isomorphic to construction A lattices, which include q -ary lattices. Since there is no efficient method to verify whether a lattice is q -ary, the implementation assumes isomorphism to a q -ary lattice, so that the user can find secure parameters even if the attack does not apply. Additionally, the hull of a lattice is expected to be trivial for construction A lattices with high probability. The estimator assumes this triviality but does not verify it for general lattices, instead allowing the user to focus on cases where the trivial hull is a reasonable assumption. Thereby leaving it to the user’s discretion to decide to either find parameters resisting the attack, or ensure that the attack is not relevant in their specific case.

Estimation of the Complexity. The complexity estimation for the Hull Attack is implemented in the `HullAttack` class. The procedure involves the following steps:

1. *Dimension Determination:* The effective dimension d of the lattice for reduction is calculated as $d = 2n$, where n is the lattice dimension. This dimension corresponds to the two ZLIP instances being reduced.
2. *Lattice Reduction Cost:* Lattice reduction is simulated using the BKZ algorithm. The block size β is chosen based on the targeted root-Hermite factor δ . The *Geometric Series Assumption* (GSA) is used to model the reduction. Under the GSA, the squared Gram-Schmidt norms of the reduced basis vectors are assumed to decrease geometrically, with the i -th norm approximated as:

$$b_i^2 = \delta^{2(i-d)} \cdot \text{vol}(\Lambda)^{2/d},$$

where δ is the root-Hermite factor, d is the lattice dimension, and $\text{vol}(\Lambda)$ is the lattice volume. The reduction cost is estimated using the `costf` function from the reduction module.

3. *Code Attack Cost:* The attack on the PCE problem is handled via the `ssa` function from the PCE module. A `PCEParameters` instance is initialized with the q -ary field modulus, code length, and code dimension. The resulting cost of solving the PCE is added to the overall cost estimate.

The resulting cost is encapsulated in a cost dictionary, with key entries including:

- `rop`: Total number of word operations (approximately CPU cycles).
- `red`: Cost of lattice reduction operations.
- `δ` : Root-Hermite factor achieved by lattice reduction.
- `β` : BKZ block size.
- `d` : Lattice dimension.
- `non_red`: Non-reduction computational cost, primarily attributed to the PCE attack.

The implementation assumes a trivial hull for construction A lattices and estimates the cost of solving the PCE problem via the PCE attacks given in the section 5.3. The `ssa` function accounts for parameters such as code weight and hull dimension, which are initialized in the `PCEParameters` class.

This implementation of the estimation of the Hull Attack follows the description of the original attack, by combining lattice reduction and PCE algorithms to estimate the concrete complexity of solving the LIP using the attack as given by [DG23]. The attack's feasibility relies heavily on the triviality of the hull, which simplifies the problem by reducing it to a graph isomorphism problem via signed permutation equivalence. The computational cost is dominated by lattice reduction, with the BKZ block size being the key determining factor of complexity.

5.4.3 Arithmetic Invariants

A quadratic form $Q = B^t B$ defined by a lattice with basis B has several arithmetic invariants that provide insights into its structural properties. These invariants are fundamental in characterizing quadratic forms and can play a significant role in answering problems such as whether two lattices are isomorphic, in the negative. Below, we describe these invariants as implemented in the lattice estimator, to be used to check that the distinguishing version of LIP cannot be solved trivially using these invariants.

Discriminant (discriminant)

The *discriminant* of a quadratic form is a measure of the determinant of the associated matrix. Formally, if Q is represented by a symmetric matrix M , the discriminant is given by:

$$\text{discriminant}(Q) = \text{disc}(M).$$

In the implementation, the discriminant is computed using the `discriminant(form)` method, where `form` is the quadratic form.

Rank (rank)

The *rank* of a quadratic form corresponds to the dimension of the associated vector space. If M is the matrix representation of Q , then:

$$\text{rank}(Q) = \text{dimension of the row space of } M.$$

This is calculated in the implementation using the `rank(form)` method.

GCD Invariant (gcd_invariant)

The *GCD invariant* is the greatest common divisor (GCD) of the diagonal entries of the matrix M associated with Q . Formally:

$$\text{GCD invariant}(Q) = \text{gcd}(M_{11}, M_{22}, \dots, M_{nn}),$$

where M_{ii} are the diagonal elements of M . This invariant is computed using the `gcd_invariant(form)` method.

Parity (parity)

The *parity* of a quadratic form indicates whether all diagonal entries of the matrix M are even. It is defined as:

$$\text{parity}(Q) = \begin{cases} 0 & \text{if all diagonal entries are even,} \\ 1 & \text{otherwise.} \end{cases}$$

This is implemented in the `parity(form)` method.

Signature (signature)

The *signature* of a quadratic form is a tuple (p, n) , where p and n denote the number of positive and negative eigenvalues of the matrix M , respectively. It provides geometric insights into the quadratic form's behavior, as:

$$\text{signature}(Q) = (\#\text{positive eigenvalues}, \#\text{negative eigenvalues}).$$

This invariant is computed using the `signature(form)` method.

Hasse Invariant (`hasse_invariant`)

The *Hasse invariant* of a quadratic form at a prime p is a local invariant that captures the structure of the form modulo p . It is computed using the Hilbert symbol:

$$\text{Hasse invariant}(Q, p) = \prod_{i < j} \left(\frac{M_{ii}, M_{jj}}{p} \right),$$

where $\left(\frac{a, b}{p} \right)$ is the Hilbert symbol. This is implemented in the `hasse_invariant(form, p)` method, where p is the prime.

Witt Invariant (`witt_invariant`)

The *Witt invariant*, or Witt index, quantifies the maximal dimension of a totally isotropic subspace of the quadratic form. It is closely related to the signature and discriminant and provides additional information about the equivalence class of the form. This invariant is computed using the `witt_invariant(form)` method.

Spinor Norm and Spinor Genus (`spinor_norm`)

The *spinor norm* is a fundamental invariant in the classification of quadratic forms and lattices, placing them into equivalence classes known as the *spinor genus* [LLM24]. Two lattices L_1 and L_2 in a quadratic space V are said to be spinor equivalent if there exists an element g in the proper orthogonal group $O^+(V)$ such that for every prime p , there exists a local transformation f_p with spinor norm 1 satisfying:

$$L_2 = g f_p L_1.$$

This equivalence refines the genus but remains coarser than proper equivalence, as it allows for a broader classification of lattices based on local-global principles. The *spinor genus* groups lattices that share these equivalence properties.

The computation of the spinor norm leverages the underlying algebraic structures of the quadratic form. The primary computational task involves determining the Hilbert symbols for pairs of values derived from the quadratic form at each prime p . Formally, the spinor norm of a lattice is calculated by evaluating products of local invariants:

$$\text{Spinor Norm}(L) = \prod_p \text{Hilbert Symbol}((a, b)_p),$$

where $(a, b)_p$ denotes the Hilbert symbol at prime p , and the values a and b are derived from the Gram matrix of the quadratic form.

The `spinor_norm(form)` method in the implementation serves as a placeholder for future computation. While the actual computation is not currently implemented due to its computational complexity, this method would ideally iterate over all relevant primes, evaluate the Hilbert symbols, and aggregate them into the spinor norm for the given quadratic form. This methodology aligns with known mathematical techniques for determining spinor equivalence in lattice theory.

The spinor genus and spinor norm are particularly significant in lattice isomorphism problems, as they provide a robust framework for distinguishing between isomorphic and non-isomorphic lattices. Future implementations of the `spinor_norm` method would enhance the precision of lattice classification, extending the computational toolkit available for analyzing the security of cryptographic schemes based on lattice structures. As mentioned above, this invariant was also recently investigated in more detail in the LIP context by [LLM24].

The arithmetic invariants of quadratic forms, as described above, provide a comprehensive framework for analyzing the structure and properties of lattices. These invariants, implemented in the estimator module, play a critical role in efficiently distinguishing non-isomorphic lattices.

5.4.4 Geometric Invariants

Geometric invariants of quadratic forms provide insights into the structure of lattices by capturing properties related to vector norms and inner products. These invariants, like the arithmetic invariants, play a crucial role in distinguishing quadratic forms and lattices. Importantly, though, evaluating these invariants typically requires finding or enumerating short vectors, making their concrete computational cost tied to lattice reduction algorithms. Below, we outline the key geometric invariants for LIP, given by [DvW21].

First Minimum (λ_1)

The *first minimum* $\lambda_1(Q)$ of a quadratic form Q is defined as the smallest norm of a nonzero vector in the lattice:

$$\lambda_1(Q) = \min_{x \in \mathbb{Z}^n \setminus \{0\}} \|x\|_Q,$$

where $\|x\|_Q = \sqrt{x^T Q x}$. This invariant provides a measure of the shortest vector in the lattice defined by Q . Finding $\lambda_1(Q)$ is computationally hard and is typically approximated using lattice reduction algorithms such as BKZ.

Successive Minima (λ_i)

The *successive minima* $\lambda_i(Q)$ generalize the first minimum by considering the smallest radius r such that i linearly independent lattice vectors lie within a ball of radius r :

$$\lambda_i(Q) = \min\{r \mid \dim(\text{span}\{x \in \mathbb{Z}^n : \|x\|_Q \leq r\}) \geq i\}.$$

This invariant captures the geometry of the lattice at increasing dimensions and is similarly difficult to compute exactly.

Kissing Number ($\kappa(Q)$)

The *kissing number* $\kappa(Q)$ is the number of shortest vectors in the lattice, given by:

$$\kappa(Q) = |\{x \in \mathbb{Z}^n : \|x\|_Q = \lambda_1(Q)\}|.$$

It represents the number of lattice points on the surface of the smallest enclosing sphere. This invariant provides insights into the local structure of the lattice.

Set of Shortest Vectors ($\text{Min}(Q)$)

The *set of shortest vectors* $\text{Min}(Q)$ includes all lattice vectors achieving the first minimum:

$$\text{Min}(Q) = \{x \in \mathbb{Z}^n : \|x\|_Q = \lambda_1(Q)\}.$$

This set is essential for understanding the symmetries and structure of the lattice.

Theta Series ($\Theta_Q(q)$)

The *theta series* of a quadratic form Q encodes the number of lattice vectors at each norm squared and is formally defined as:

$$\Theta_Q(q) = \sum_{k \geq 0} N_k q^k,$$

where $N_k = |\{x \in \mathbb{Z}^n : \|x\|_Q = k\}|$. The theta series is a generating function that provides a holistic view of the lattice's geometry.

The computation of geometric invariants is intrinsically tied to the enumeration or approximation of short vectors in the lattice. This involves solving hard problems such as the Shortest Vector Problem (SVP) or Closest Vector Problem (CVP). Practically, these computations rely on lattice reduction techniques like BKZ or enumeration within a bounded radius.

The cost function for these computations is dominated by the block size parameter β in BKZ, which governs the quality of the reduction. For large values of β , the computational complexity grows exponentially:

$$\text{Cost}(\text{BKZ}) = O(2^{c\beta}),$$

where c is a constant that depends on the specific implementation. For practical estimations, the complexity of finding short vectors is often approximated using heuristic models for lattice reduction.

In summary, while geometric invariants provide valuable structural insights into quadratic forms and lattices, their computation is challenging and depends heavily on efficient algorithms for lattice reduction. Therefore, the estimations of these invariants will behave in a similar way to the algorithm by Haviv and Regev.

6 Conclusions

This thesis has explored the Lattice Isomorphism Problem (LIP) as a foundational problem in post-quantum cryptography, providing both theoretical insights and practical tools for evaluating its security. By bridging the gap between computational hardness results and concrete cryptanalytic techniques, this work has made advancements in the understanding of LIP and its cryptographic potential.

Theoretical Results and Structural Vulnerabilities

One of our intended goals was to find out whether the increased structure in lattices can be leveraged to attack LIP on other similarly structured lattices, with a focus on the q -ary to p -ary reduction and the extension of the transformation space between q -ary lattices. Previous research has suggested that simpler lattices, such as those generated over \mathbb{Z}^n , might exhibit greater resilience to certain attacks due to their reduced structure [DvW21]. This serves as an important context for evaluating the findings presented in this work.

The q -ary to p -ary reduction demonstrates that the structure of q -ary lattices can, under the assumption of signed permutation transformations, enable connections to p -ary lattices. This reduction simplifies the problem space and allows cryptanalytic techniques for p -ary lattices to be applied to q -ary lattices. However, this indicates a potential vulnerability, as the additional structure in q -ary lattices might be exploited by attackers, making them less secure under certain circumstances.

In contrast, this thesis also establishes that the transformation space between q -ary lattices extends beyond signed permutations. This discovery suggests that q -ary lattices are governed by a broader set of equivalence transformations, thereby mitigating the impact of the q -ary to p -ary reduction. The presence of these additional transformations expands the theoretical framework of lattice isomorphism and highlights that q -ary lattices may not be as inherently vulnerable as previously thought.

Taken together, these results provide a nuanced perspective. While the q -ary to p -ary reduction hints at potential vulnerabilities introduced by inherent structural properties, the extended transformation space demonstrates that these vulnerabilities might be less significant in practice. This aligns with the broader theme that structural complexity in lattices, when properly understood and accounted for, does not necessarily equate to reduced security. These insights emphasize the importance of continuing to explore the interplay between lattice structure and cryptographic security to inform the development of resilient cryptographic primitives.

Estimating the Concrete Security of LIP

In the beginning of the thesis we also posed the question of: "How well can we estimate the concrete security of LIP?" Through the development and application of the Lattice Estimator and the analysis of various cryptanalytic techniques, we have demonstrated that the concrete security of LIP can be effectively estimated based on the current

attacks described in the literature. The primary observation is that the problem of LIP is "lattice-based" in a cryptanalytic sense; the most effective attacks, such as those by Haviv and Regev or those leveraging the hull structure, rely fundamentally on lattice reduction techniques.

Compared to other problems like Learning with Errors (LWE), where security estimates can often be derived from general parameters such as modulus and dimension, LIP presents a unique challenge. Each instance of LIP may involve quadratic forms with highly varying properties, leading to instances that are significantly different in terms of cryptanalytic difficulty. This variability necessitates the use of the actual quadratic forms for security estimation, as general parameters alone are insufficient for accurately assessing the complexity of the problem.

Additionally, the implementation of arithmetic invariants as a distinguishing method provides an efficient tool for identifying and filtering out easily distinguishable lattices. This contributes to a layered understanding of security: while the broader attack space is dominated by lattice reduction, the invariants offer a complementary, algebraic perspective on lattice equivalence.

The implications of these findings are twofold. First, they underline the necessity of parameter selection that balances resistance to reduction-based attacks with robustness against invariant-based filtering. Second, they highlight the inherent structural hardness of LIP, suggesting that it may serve as a robust foundation for cryptographic schemes, provided that care is taken in designing against specific attack vectors.

In conclusion, while LIP's security estimation framework is well-developed based on existing techniques, it requires instance-specific analysis due to the variability of quadratic forms. This distinguishes LIP from other lattice problems like LWE and emphasizes the importance of tailored approaches to estimating its concrete security. Furthermore, this suggests an approach to achieve more general parameters for estimation of LIP security would be to further investigate and classify properties of quadratic forms as they relate to lattices.

Key Contributions. The primary contributions of this thesis include:

- *Analysis of LIP Security:* A comprehensive examination of cryptanalytic techniques for LIP, including the Hull Attack, which reduces LIP to the PCE problem. The study highlights the critical role of the trivial hull property in determining the hardness of specific LIP instances, and the possibility of the dimension of the hull of a code having effects on construction A lattices.
- *Reduction from q -ary to p -ary LIP:* A novel reduction demonstrating structural connections between q -ary and p -ary LIP instances. This reduction broadens the applicability of existing cryptanalytic tools to additional lattice classes.
- *Extension of the Transformation Space:* A proof that orthogonal transformations between q -ary lattices extend beyond signed permutations. This result reveals a larger transformation space for q -ary lattices, thereby increasing the security of LIP on these lattices compared to previous assumptions. The existence of these

transformations reduces the effectiveness of certain attacks, particularly those reliant on signed permutation equivalence.

- *Development of the Lattice Estimator:* The implementation and integration of modules for estimating the security of LIP-based cryptographic schemes. These extensions incorporate state-of-the-art cryptanalytic methods, such as Haviv and Regev’s algorithm, Beullens’ approach, and the Support Splitting Algorithm, enabling detailed evaluations of parameter selection and security.

Significance of Results. The findings of this thesis demonstrate that LIP offers a robust foundation for cryptographic primitives, when applied to the right lattices. By proving the existence of non-signed permutation transformations, this work establishes that the security for q -ary lattices might be stronger than previously thought, giving hope for possible uses of LIP in applications of updatable encryption with q -ary lattices. Additionally, the extended analysis of parameter selection and cryptanalytic methods provides practical guidance for designing secure LIP-based cryptosystems.

Future Work

This thesis opens several avenues for further research:

1. *Hull Properties in Construction A Lattices:* While this work assumes trivial hulls for q -ary lattices, further investigation into the implications of non-trivial hulls could reveal additional strengths or weaknesses in LIP-based cryptography.
2. *Comprehensive Analysis of Transformation Spaces:* The discovery of non-signed permutation transformations raises questions about the complete classification of orthogonal transformations for q -ary lattices. Developing systematic methods to enumerate and analyze these transformations is a promising direction.
3. *Broader Cryptographic Applications:* Extending the applicability of LIP to other cryptographic primitives could enhance its utility, such as using LIP for updatable encryption by considering the group action framework used in isogeny-based updatable encryption. Further, exploring connections between LIP and other lattice-based problems may yield new cryptographic constructions.
4. *Optimization of Cryptanalytic Tools:* Enhancing the efficiency of the Lattice Estimator and integrating new cryptanalytic techniques could provide deeper insights into the concrete security of LIP under various parameter settings.
5. *Quadratic Form Properties and General Parameters:* Further investigating the properties of quadratic forms as they relate to lattices could lead to the identification of more general parameters for estimating the concrete security of LIP. Understanding these relationships may help reduce the dependence on instance-specific analysis and provide a broader framework for evaluating security.

By addressing LIP's complexity, extending its cryptanalytic analysis, and proving the existence of a larger transformation space for q -ary lattices, this thesis makes contributions to the field of post-quantum cryptography. These results not only enhance the theoretical understanding of LIP but also offer practical tools and guidance for its deployment in secure cryptosystems. This work lays the foundation for future research into the resilience and versatility of LIP in a quantum-resistant cryptographic landscape.

References

- [AGPS19] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. Estimating quantum speedups for lattice sieves. Cryptology ePrint Archive, Paper 2019/1161, 2019.
- [Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.
- [APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046, 2015. <https://eprint.iacr.org/2015/046>.
- [Bab16] László Babai. Graph isomorphism in quasipolynomial time, 2016.
- [Beu20] Ward Beullens. Not enough less: An improved algorithm for solving code equivalence problems over \mathbb{F}_q . Cryptology ePrint Archive, Paper 2020/801, 2020. <https://eprint.iacr.org/2020/801>.
- [DG23] Léo Ducas and Shane Gibbons. Hull attacks on the lattice isomorphism problem. Cryptology ePrint Archive, Paper 2023/194, 2023. <https://eprint.iacr.org/2023/194>.
- [DPPvW22] 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, Paper 2022/1155, 2022.
- [Dv22] 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.
- [DvW21] Léo Ducas and Wessel van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. Cryptology ePrint Archive, Paper 2021/1332, 2021. <https://eprint.iacr.org/2021/1332>.
- [HR13] Ishay Haviv and Oded Regev. On the lattice isomorphism problem, 2013.
- [LB88] Pil Joong Lee and Ernest F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In *International Conference on the Theory and Application of Cryptographic Techniques*, 1988.
- [Leo82] J. Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 1982.

- [LLL82] Arjen K Lenstra, Hendrik W Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
- [LLM24] Cong Ling, Jingbo Liu, and Andrew Mendelsohn. On the spinor genus and the distinguishing lattice isomorphism problem. Cryptology ePrint Archive, Paper 2024/1475, 2024.
- [Mic11] Daniele Micciancio. Lattice-based cryptography. In *Encyclopedia of Cryptography and Security*, 2011.
- [MPMPW24] Guilhem Mureau, Alice Pellet-Mary, Heorhii Pliatsok, and Alexandre Wallet. Cryptanalysis of rank-2 module-lip in totally real number fields. Cryptology ePrint Archive, Paper 2024/441, 2024. <https://eprint.iacr.org/2024/441>.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.
- [SE94] Claus-Peter Schnorr and Michael Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *International Symposium on Fundamentals of Computation Theory*, pages 68–85. Springer, 1994.
- [Sen97] Nicolas Sendrier. On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293, 1997.
- [Sen00] N. Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.
- [Sen06] N. Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Trans. Inf. Theor.*, 46(4):1193–1203, September 2006.
- [Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.