Exploiting the complexity of Lattice Isomorphism Problem via Irreducible Decomposition

Kaijie Jiang[1], Yinchen Liu[2]

[1] Institute for Advanced Study, Tsinghua University, Beijing, China
jkj21@mails.tsinghua.edu.cn
[2] Institute for Interdisciplinary Information Sciences, Tsinghua University
liuyinch23@mails.tsinghua.edu.cn

Abstract. The Lattice Isomorphism Problem (LIP) is a computational problem that has recently been introduced into cryptography and is believed to be hard. Its search version, Search Lattice Isomorphism Problem (SLIP), is considered even harder than the Shortest Vector Problem (SVP), yet its complexity is still not well understood. Haviv and Regev (SODA 2014) showed that the decisional version (DLIP) lies in a statistical zero-knowledge class and is therefore unlikely to be NP-hard. This result does not apply to the search version, which motivates the question of whether NP can reduce to SLIP.

Our main result answers this question negatively. We show that every language reducible to SLIP lies in AM and coAM, by analyzing the direct-sum structure of irreducible lattices. Consequently, NP cannot reduce to SLIP unless the polynomial hierarchy collapses, and there is no reduction from SVP to SLIP unless the polynomial hierarchy collapses.

We also study several problems closely related to LIP and establish reductions between its search, counting, and decisional variants. These connections mirror known relationships for graph isomorphism. Finally, we propose a new algorithm that uses a KZ basis to compute an orthogonal decomposition of a lattice.

1 Introduction

Lattices are discrete additive subgroups of $\mathbb{R}^m$. A lattice is usually defined by a set of n linearly independent basis vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^m$, such that any vector in the lattice can be expressed as an integer linear combination of the basis vectors. Despite their simplicity, lattices offer a rich geometric structure that can be used to define various computationally hard problems, the most important of which are the *Shortest Vector Problem* (SVP), which involves finding the shortest non-zero vector in a given lattice, and the *Closest Vector Problem* (CVP), which involves finding the lattice point closest to a given target point. Both of these problems are known to be NP-hard, and their theoretical complexity and algorithms have been extensively studied [37,13,16,2,4]. A major challenge in the area lies in comprehending the computational complexity of SVP, CVP and

related lattice problems for polynomial approximation factors, which constitute the foundation of lattice-based cryptography.

Apart from SVP and CVP, several other significant lattice-related problems have gained significant interest. One such problem is the *Lattice Isomorphism Problem* (LIP). Two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ are said to be isomorphic if there exists an orthogonal transformation that maps $\mathcal{L}_1$ to $\mathcal{L}_2$. The *Decisional Lattice Isomorphism Problem* (DLIP) is to decide whether there exists an orthogonal linear transformation mapping $\mathcal{L}_1$ to $\mathcal{L}_2$, and the *Search Lattice Isomorphism Problem* (SLIP) is to find such an orthogonal transformation given two isomorphic lattices $\mathcal{L}_1$ and $\mathcal{L}_2$. In the formal definitions below, these lattices are represented by integral Gram matrices, and an isomorphism is output as the corresponding coordinate change matrix. Research on LIP can be traced back to the work of Plesken and Souvignier [56,57], focusing on the creation of algorithms to address low-dimensional instances of LIP. The asymptotic complexity of LIP was later considered by Dutour Sikirić, Schürmann, and Vallentin [58]. They prove that DLIP is at least as hard as the *Graph Isomorphism Problem* (GIP)[3]. In [38], Haviv and Regev propose an $n^{O(n)}$-time algorithm for SLIP by using a new generalized isolation lemma [60] and an SVP oracle, which remains the fastest known algorithm for solving SLIP, and they prove that DLIP $\in$ SZK (statistical zero knowledge). In [27], Sikirić et al. demonstrate that with access to an SVP oracle, an SLIP instance can be reduced to a GIP instance. While GIP possesses a quasi-polynomial time algorithm, as demonstrated in [10], the number of shortest vectors within a lattice may be exponential in the worst case, potentially resulting in an exponential-sized graph, as discussed in [27].

Lattice Isomorphism Problems have recently been used to construct cryptographic schemes, including public key encryption [15,1], fully-homomorphic encryption [22,48], signature [25,26], key encapsulation mechanism [26], identification [26] and commitment schemes [41]. Most of these schemes are based on lattice isomorphism problems with special algebraic (module-LIP) or geometric properties ($\mathbb{Z}$LIP), sparking renewed interest in the domain of lattice isomorphism problems. The module-LIP, an algebraic variant of LIP, was introduced by Ducas et al. [25]. There is a series of cryptanalytic works on module-LIP [53,50,8]. For $\mathbb{Z}$LIP, the goal is to find an orthogonal transformation that maps $\mathbb{Z}^n$ to $\mathcal{L}$, provided that $\mathcal{L}$ is isomorphic to $\mathbb{Z}^n$. Indeed, $\mathbb{Z}$LIP is a well-known problem, and determining its computational complexity has remained a long-standing open question. This challenge has inspired a rich and elegant body of research [34,59,31,43,44,47,40,18,15,42,24].

Although both SLIP and DLIP are employed in various cryptographic schemes, their computational complexity is not well understood, particularly compared to the well-studied graph isomorphism problem (GIP). As mentioned above, all existing algorithms for SLIP require the SVP oracle as a subroutine [38,27] and in [38], Haviv and Regev prove that DLIP $\in$ SZK, and it is well known that SZK $\subseteq$ AM $\cap$ coAM [30,5], thus DLIP is unlikely to be NP-hard, unless the poly-

[3] Given two graphs G_1 and G_2 one has to decide whether there exists an edge-preserving bijection from the vertex set of G_1 to that of G_2.

nomial hierarchy (PH) collapses [20]. Therefore, a natural question is whether SLIP can be NP-hard, and in particular, is there a reduction from SVP to SLIP? Careful readers may wonder whether there exists a reduction from SLIP to DLIP. If such a reduction does exist, it would imply that SLIP is unlikely to be NP-hard, unless PH collapses. However, our understanding of the reduction from SLIP to DLIP remains limited. Currently, the only related works are [59,32]. In [59], Szydlo established a polynomial-time reduction from the Search $\mathbb{Z}$LIP problem to DLIP. In [32], van Gent and van Woerden extended this result to general lattices with direct sum structures and modular lattices. In this respect, LIP is different from GIP and *Code Equivalence Problem*[4] (CEP). For CEP, Petrank and Roth show that it lies in NP $\cap$ coAM [55], Biasse and Micheli show that its search version is polynomially equivalent to the decision version [17], thus the search version of CEP is unlikely to be NP-hard. For GIP, it is shown in [35] that GIP lies in NP $\cap$ coAM and the search version is polynomially equivalent to the decision version [51]. In addition, it is shown in [51] that GIP, calculating a set of generators of the automorphism group of a graph (GAP), counting all the isomorphisms between the graphs (#GIP), calculating the order of the automorphism group of a graph (#GAP) are all polynomially equivalent. Similarly, we can define these problems on lattice isomorphisms, but the reductions between these problems are not well studied.

Therefore, in this paper, we delve into the complexity of LIP and focus on two key questions, namely,

Question 1. Is SLIP an NP-hard problem?

Question 2. What are the reductions among LAGP, SLIP, #LAP, and DLIP?

1.1 Our Results and Techniques

On Basing SLIP on NP-hardness

For the first question, we provide a negative answer by proving that SLIP cannot be NP-hard unless PH collapses. Specifically, we prove the following theorem:

Theorem 1.1 (see also Theorem 3.1). *If there exists a randomized Turing reduction from a decision problem L to* SLIP*, then $L \in$ AM $\cap$ coAM.*

This result leads to the following interesting corollary, which uses the result that SVP is NP-hard [6].

Corollary 1.1. *If there exists a randomized Turing reduction from the shortest vector problem* SVP *to* SLIP*, then* NP $\subseteq$ coAM*.*

In order to obtain a black-box separation between SLIP and NP-hardness, let R be a reduction from a decision problem L to SLIP. We aim to construct an

[4] Given two n-dimensional linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ over some field $\mathbb{F}$, the goal is to decide whether there exists a permutation on the coordinates mapping $\mathcal{C}_1$ to $\mathcal{C}_2$.

AM protocol for L using the reduction R. A natural approach is for the verifier to sample a random tape and send it to the prover. The prover then simulates the reduction R and answers any queries to SLIP by leveraging its unbounded computational power. Subsequently, the prover sends the simulation transcript along with the answers to the verifier, who checks their correctness. However, since SLIP is a search problem, there is not always a unique correct answer. The prover thus has the freedom to choose which answer to provide for each query. This freedom enables a malicious prover to potentially deceive the verifier. This issue can be addressed by shifting the search problem to a sampling problem, since in this setting the correct answer distribution for each query is unique. The idea was originally proposed by Bogdanov and Brzuska [19]. We adopt their general proof framework, which mainly consists of two steps.

The first step is to shift from a search problem to a sampling problem $\mathcal{S}$. The second step is to prove that this sampling problem possesses certain properties that can be leveraged to separate NP-hardness. In our proof, the sampling problem $\mathcal{S}$ we use has a *size-verifiable* property.

Inspired by [42], we defined a new sampling problem related to lattice automorphisms, denoted as the Sampling Lattice Automorphism Problem (SaLAP): on input an n-dimensional lattice $\mathcal{L}$, SaLAP outputs a uniform sample $\varphi \xleftarrow{\$} \mathrm{Aut}(\mathcal{L})$.

We then proved that there exists a randomized Turing reduction from SLIP to SaLAP. A key technique in the reduction is a lemma concerning the structure of the automorphism group of the direct sum of irreducible lattices. Informally, this lemma states that for any automorphism ϕ on such a lattice, ϕ can be viewed as a permutation of the irreducible lattice components. Here, an irreducible lattice is defined as one that cannot be further orthogonally decomposed into two sublattices. Moreover, each irreducible lattice plays a role analogous to a connected component in a graph. In addition, according to the unique orthogonal decomposition theorem proved by Eichler in 1952 [28], every lattice $\mathcal{L}$ has a unique orthogonal decomposition $\mathcal{L} = \mathcal{N}_1 \oplus \cdots \oplus \mathcal{N}_s$, where each $\mathcal{N}_i$ is an irreducible lattice. [5]

Let us now outline the main idea. The reduction essentially follows a divide-and-conquer approach. For an SLIP instance $(\mathcal{L}, \mathcal{L}')$, we first construct a lattice $\mathcal{N} = \mathcal{L} \oplus \mathcal{L}'$, and then aim to find a "lucky" automorphism in $\mathrm{Aut}(\mathcal{N})$. From this we can extract an isomorphism between $\mathcal{L}$ and $\mathcal{L}'$. However, the proportion of such "lucky" automorphisms in the entire automorphism group depends on the number of the irreducible components of $\mathcal{L}$. When $\mathcal{L}$ is an irreducible lattice, we can apply the aforementioned lemma to show that the proportion of "lucky" automorphisms in $\mathrm{Aut}(\mathcal{N})$ is $1/2$. If $\mathcal{L}$ has more irreducible components, then the proportion of "lucky" automorphisms may be exponentially small. Fortunately, if $\mathcal{L}$ is not an irreducible lattice, we can always use the uniform distribution on $\mathrm{Aut}(\mathcal{N})$ to orthogonally decompose $\mathcal{L}$ and $\mathcal{L}'$ into $\mathcal{L}_1 \oplus \mathcal{L}_2$ and $\mathcal{L}'_1 \oplus \mathcal{L}'_2$ with high probability, thereby reducing SLIP instance $(\mathcal{L}, \mathcal{L}')$ to two SLIP instances:

[5] Later, Kneser gave a constructive and simpler proof using an Errathostenes sieve [45].

$(\mathcal{L}_1, \mathcal{L}_1')$ and $(\mathcal{L}_2, \mathcal{L}_2')$, where $\mathcal{L}_1$ and $\mathcal{L}_2$ have a smaller number of irreducible components, thus obtaining a divide-and-conquer approach. Therefore, if a decision problem L can be reduced to SLIP, then it can also be reduced to SaLAP.

For the second step, we demonstrate that the size of the automorphism group of a lattice is *verifiable*, meaning that the decision problem which, given a Gram matrix $\mathbf{G}$ and an integer s, asks whether $|\mathrm{Stab}(\mathbf{G})| = s$ lies in AM. We actually construct an IP protocol and then use the well-known results [36,11] to obtain an AM protocol. The main techniques used in this protocol are the randomized self-reduction algorithm on LIP [26] and the BPP algorithm proposed by Beals and Babai [12] for computing the order of finite subgroups on $\mathrm{GL}_n(\mathbb{Q})$. For a common input $(\mathbf{G}, s)$, the prover outputs a generating set $\{g_1, \ldots, g_k\}$ of $\mathrm{Stab}(\mathbf{G})$ and sends the set to the verifier. The verifier can use the randomized self-reduction algorithm on LIP and the BPP algorithm proposed by Beals and Babai to determine whether $\{g_1, \ldots, g_k\}$ generates $\mathrm{Stab}(\mathbf{G})$, and then use the algorithm of Beals and Babai to determine whether its order is equal to s.

In the protocol, we represent lattices by Gram matrices; see Section 2 for the common input model used throughout the paper.

In addition, we reduce the computation of the orthogonal decomposition of a lattice (LODP, see Definition 2.13) to calculating the Korkine-Zolotareff (KZ) basis of a lattice.

Theorem 1.2 (see also Theorem 3.2). *Let $\mathcal{L}$ be a full-rank lattice with a* KZ *basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$. Define a graph $G_{\mathcal{L}} = (V, E)$ where $V = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ and $(\mathbf{b}_i, \mathbf{b}_j) \in E$ if and only if $\langle \mathbf{b}_i, \mathbf{b}_j \rangle \neq 0$. Then, the unique orthogonal decomposition is given by $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$, where each $\mathcal{L}_i$ is spanned by the basis vectors in the i-th connected component of $G_{\mathcal{L}}$, and is irreducible.*

It is a standard fact that KZ basis can be found using SVP oracle at most n times. Thus, if we plug in the fastest known algorithm for SVP [3], we have the following corollary.

Corollary 1.2. *There is a randomized $2^{n+o(n)}$-time and space algorithm for* LODP.

The Reductions among LAGP, SLIP, #LAP, and DLIP

For the second question, we have some positive answers. Specifically, we prove that the reduction results on LIP correspond to most of the results on the GIP, except for the reduction from #GAP to GIP and the reduction from Search GIP to #GAP. These directions can be considered as open problems for future research.

Our first reduction result corresponds to the reduction that GAP is a polynomial-time equivalent to the search GIP, except that our reduction is a randomized reduction.

Theorem 1.3 (see also Theorem 4.1). *In the context of randomized Turing reduction,* LAGP *is equivalent to* SLIP.

This reduction uses a basic fact in computational group theory: computing the generating set of a finite group is almost equivalent to uniform sampling on this finite group [9,23]. Specifically, in [42], Jiang et al. use the randomized self-reduction of LIP to prove that a reduction from SaLAP to SLIP. Combining the reduction from SLIP to SaLAP we mentioned earlier, we can complete the reduction. However, one thing to note is that when using a set of group generators to uniformly sample group elements, there will be some errors in the distribution (see Theorem 2.4). Therefore, we need to make a small modification to the reduction proof from SLIP to SaLAP.

From the above reduction and the algorithm of Beals and Babai [12] mentioned above, we can get the following reduction.

Corollary 1.3. *There is a randomized Turing reduction from* #LAP *to* SLIP*.*

Combining this corollary with Theorem 1.1, we can obtain the result that #LAP cannot be #P-complete unless PH collapses.

Moreover, we demonstrate that there is an efficient reduction from DLIP to #LAP.

Theorem 1.4 (see also Theorem 4.2). *There is a Turing reduction from* DLIP *to* #LAP*.*

Recall the reduction from GIP to #GAP, for a GIP instance (G_1, G_2), we can assume that G_1 and G_2 are connected graphs, because computing the connected components of a graph is efficient. Then the reduction determines whether G_1 and G_2 are isomorphic by analyzing the relationship among the quantities $\#\mathrm{Aut}(G_1)$, $\#\mathrm{Aut}(G_2)$, and $\#\mathrm{Aut}(G_1 \oplus G_2)$. For lattices, however, it is not known whether the orthogonal decomposition of $\mathcal{L}$ can be computed efficiently, even with access to a #LAP oracle. Interestingly, given a DLIP instance $(\mathcal{L}_0, \mathcal{L}_1)$, we find that we can determine whether $\mathcal{L}_0$ is isomorphic to $\mathcal{L}_1$ by analyzing the relationship among the quantities $n_{00} = |\mathrm{Aut}(\mathcal{L}_0 \oplus \mathcal{L}_0)|$, $n_{01} = |\mathrm{Aut}(\mathcal{L}_0 \oplus \mathcal{L}_1)|$, and $n_{11} = |\mathrm{Aut}(\mathcal{L}_1 \oplus \mathcal{L}_1)|$. Specifically, $\mathcal{L}_0$ is isomorphic to $\mathcal{L}_1$ if and only if $n_{00} = n_{01} = n_{11}$ holds. The proof of the reduction essentially relies on the lemma concerning the structure of the automorphism group of the direct sum of irreducible lattices that we mentioned earlier.

Therefore, a direct corollary of the above theorem is that #LIP is polynomial-time equivalent to #LAP, and we use similar reduction techniques to prove that #LAP can be used to distinguish whether a lattice is irreducible.

Corollary 1.4. *In the context of Turing reduction,* #LIP *is equivalent to* #LAP*.*

See Fig. 1 for a summary of our results.

1.2 Related Works

The potential of basing cryptography on the minimal assumption NP $\not\subseteq$ BPP lies at the core of complexity-theoretic cryptography. Previous investigations

into constructing cryptography on worst-case NP-hardness have produced several negative results for black-box reduction. The work of Akavia, Goldreich, Goldwasser and Moshkovitz [7] and Bogdanov and Brzuska [19] showed that a special class of one-way functions called *size-verifiable* one-way functions cannot be based on NP-hardness. Liu [49] then generalized the proof framework in [19] and proved that the search $\mathsf{SIVP}_{\omega(n\log n)}$ [6] is not NP-hard, unless PH collapses.

Although we showed that SLIP is not NP-hard, an interesting result is that in [14], Bennett, Dadush, and Stephens-Davidowitz introduced the *Lattice Distortion Problem* (LDP), which serves as a generalization of LIP. LDP aims to determine the degree of "similarity" between two lattices: Given two n-dimensional lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, the goal is to compute a bijective linear transformation $\mathbf{T}$ that maps $\mathcal{L}_1$ to $\mathcal{L}_2$ while minimizing the condition number $\|\mathbf{T}\| \cdot \|\mathbf{T}^{-1}\|$ of $\mathbf{T}$, where $\|\mathbf{T}\| = \sup_{\|\mathbf{x}\|=1} \|\mathbf{T}\mathbf{x}\|$. They showed that LDP is NP-hard to approximate to within any constant factor (under randomized reductions), by a reduction from SVP.

In [39], Hemkemeier and Vallentin presented an algorithm based on Kneser's method to solve the Lattice Orthogonal Decomposition Problem (LODP, see Definition 2.13). Given a generating system $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_s\} = \{\mathbf{v} \in \mathcal{L} : \|\mathbf{v}\| \leq B\}$ for an n-dimensional lattice $\mathcal{L}$, $s = |S|$, their algorithm solves the LODP in $O(n^6 \log^2 B + sn^2)$ time. However, their analysis does not account for the complexity of constructing this set S. This pre-computation can be a significant bottleneck, as the number of vectors s can be exponentially large. In fact, the parameter B needs to be at least larger than λ_n. Thus, the time complexity of precomputing this part is at least $2^{n+o(n)}$, since it requires solving at least one instance of SVP.

1.3 Outline

The rest of the paper is organized as follows. In Section 2, we present the basic definitions and preliminaries. In Section 3, we prove our main result and its corollary. In Section 4, we show the reduction results corresponding to the known results in the graph isomorphism problem.

2 Preliminaries

2.1 Notations

– Matrices and column vectors are denoted by bold letters, such as $\mathbf{A}$ and $\mathbf{a}$. For a matrix $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ we denote its Gram-Schmidt orthogonalization by $\widetilde{\mathbf{A}} = (\widetilde{\mathbf{a}}_1, \ldots, \widetilde{\mathbf{a}}_n)$. The Euclidean norm of $\mathbf{a} \in \mathbb{R}^n$ is denoted by $\|\mathbf{a}\|$. The transpose of $\mathbf{A}$ is denoted by $\mathbf{A}^\top$, and $(\mathbf{A}^{-1})^\top$ is abbreviated as $\mathbf{A}^{-\top}$.

[6] $\omega(n\log n)$-*Approximation Shortest Independent Vector Problem*: given an n-dimensional lattice $\mathcal{L}$, find a set of n linearly independent vectors whose length is at most $\omega(n\log n) \cdot \lambda_n(\mathcal{L})$, where the length of a vector set is defined as the length of the longest vector in the set.

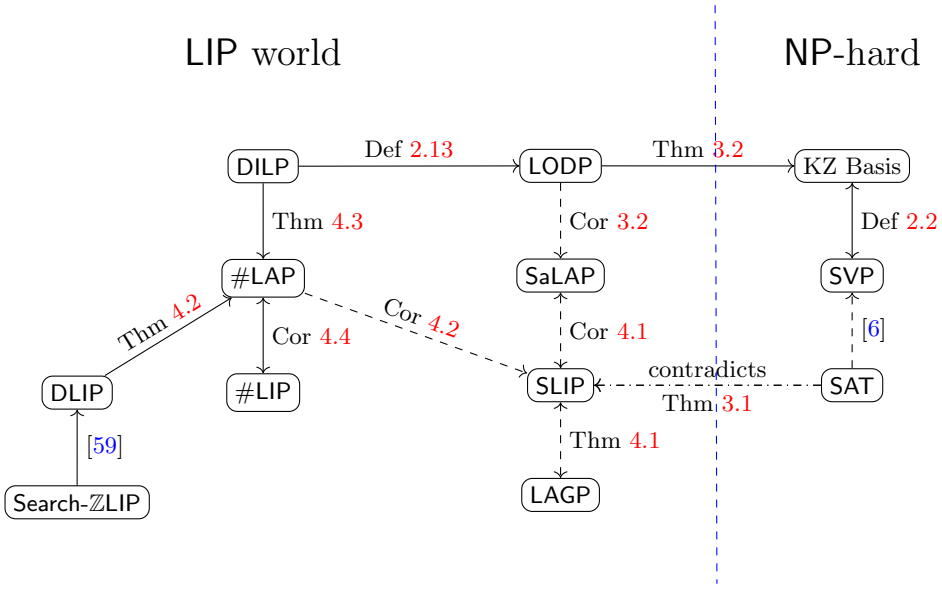

Fig. 1. An illustration of the known results together with our new reduction results. Solid arrows denote Turing reductions, dashed arrows denote randomized Turing reductions, and the dash-dotted arrow denotes a black-box separation.

- We use $x \leftarrow \mathcal{D}$ to denote that x is sampled from a distribution $\mathcal{D}$. For a finite set S, we write $s \xleftarrow{\$} S$ to denote that s is drawn uniformly from S.
- Let $[n] = \{1, 2, \ldots, n\}$ for a positive integer n. The size of a finite set A is denoted by $|A|$.
- Let $\mathrm{GL}_n(\mathbb{R})$ and $\mathrm{GL}_n(\mathbb{Z})$ be the general linear group of rank n over $\mathbb{R}$ and $\mathbb{Z}$, respectively. We use $\mathrm{O}_n(\mathbb{R})$ to represent the group of orthogonal matrices $\mathbf{O} \in \mathrm{GL}_n(\mathbb{R})$ such that $\mathbf{O}^\top \mathbf{O} = \mathbf{I}_n$, where $\mathbf{I}_n$ is the identity matrix.
- Let $\mathcal{S}_n^{>0}(\mathbb{Z})$ denote the set of $n \times n$ symmetric positive definite matrices with integer entries. For a $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, we use $[\mathbf{G}]$ to represent the set $\{\mathbf{U}^\top \mathbf{G} \mathbf{U} : \mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})\}$. For Gram matrices $\mathbf{G}_1, \ldots, \mathbf{G}_s$, we write $\mathbf{G}_1 \oplus \cdots \oplus \mathbf{G}_s$ for their block diagonal direct sum.
- We denote the group of permutation matrices of size $n \times n$ as $\mathcal{S}_n$.
- For a finite collection of lattices $\{\mathcal{L}_i\}_{i \in [s]}$ we write $\mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$ for their orthogonal sum. For a lattice $\mathcal{L}$ and its sublattices $\mathcal{L}_1$ and $\mathcal{L}_2$, we write $\mathcal{L} = \mathcal{L}_1 \oplus \mathcal{L}_2$ if and only if $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2, \mathcal{L}_1 \cap \mathcal{L}_2 = \{0\}$ and $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, for all $\mathbf{x} \in \mathcal{L}_1$, $\mathbf{y} \in \mathcal{L}_2$.

2.2 Lattice and Related

A lattice $\mathcal{L}$ of rank n and dimension m is a set of points in $\mathbb{R}^m$ that can be expressed as integer combinations of n linearly independent basis vectors

$\mathbf{b}_1, \ldots, \mathbf{b}_n$. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be the corresponding basis matrix; then $\mathcal{L} = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^n\}$. Throughout the remainder of this paper, we restrict our attention to full-rank integral lattices (unless stated otherwise). Here, full rank means that $m = n$, i.e., there exists a basis $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$ of $\mathcal{L}$. By integral, we mean that for any $\mathbf{x}, \mathbf{y} \in \mathcal{L}$, the inner product satisfies $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$. For a matrix $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$, we denote $\mathcal{L}(\mathbf{B})$ as the lattice generated by $\mathbf{B}$. Let $\lambda_i(\mathcal{L}) = \inf\{r : \dim(\mathrm{span}(\mathcal{L} \cap \overline{\mathbf{B}}(0, r))) \geq i\}$ denote the i-th successive minimum of the lattice $\mathcal{L}$, where $\overline{\mathbf{B}}(0, r) = \{\mathbf{x} \in \mathbb{R}^m : \|\mathbf{x}\| \leq r\}$ is the closed ball of radius r around 0.

Next, we review the definition of the Korkine-Zolotareff (KZ) basis. To define it precisely, we first need to define the Gram-Schmidt orthogonalization process.

Definition 2.1 (Gram-Schmidt orthogonalization). *Given a sequence of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, the corresponding Gram-Schmidt orthogonalized vectors $\widetilde{\mathbf{b}}_i$ are defined by:*

$$\widetilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \widetilde{\mathbf{b}}_j, \quad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2},$$

where $\mu_{i,j}$ are called the Gram-Schmidt coefficients. Geometrically, $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to the subspace spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

With this orthogonalization process defined, we can now introduce the KZ basis:

Definition 2.2 (Korkine-Zolotareff Basis). *Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a basis of a full-rank lattice $\mathcal{L}$ and let $\{\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n\}$ be the corresponding Gram-Schmidt orthogonalized vectors. For each $i \in [n]$, define the projection operator $\pi_i^{(\mathbf{B})}$ onto the subspace $\mathrm{span}\{\widetilde{\mathbf{b}}_i, \ldots, \widetilde{\mathbf{b}}_n\} = \mathrm{span}\{\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_{i-1}\}^\perp$ by:*

$$\pi_i^{(\mathbf{B})}(\mathbf{x}) = \sum_{j=i}^{n} \frac{\langle \mathbf{x}, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2} \cdot \widetilde{\mathbf{b}}_j.$$

A basis $\mathbf{B}$ is called a Korkine-Zolotareff basis *if it satisfies the following two conditions for all i:*

1. ***Shortest vector condition:*** *$\widetilde{\mathbf{b}}_i$ is a shortest non-zero vector in the projected lattice $\pi_i^{(\mathbf{B})}(\mathcal{L}) = \{\pi_i^{(\mathbf{B})}(\mathbf{v}) : \mathbf{v} \in \mathcal{L}\}$.*
2. ***Coefficient condition:*** *For all $j < i$, the Gram-Schmidt coefficients satisfy $-\frac{1}{2} < \mu_{i,j} \leq \frac{1}{2}$.*

The KZ basis is a stronger notion than the more computationally tractable LLL reduced basis. A KZ basis exists for every lattice $\mathcal{L}$ and, by definition, can be computed using at most n calls to the SVP oracle. Many of its properties have been studied in [46]; however, its fundamental relationship with the orthogonal decomposition of a lattice has not been previously identified in the literature, as we will demonstrate in Theorem 3.2.

2.3 Lattice Isomorphism and Automorphism

The automorphism group $\mathrm{Aut}(\mathcal{L})$ of an n-dimensional lattice $\mathcal{L}$ consists of all orthogonal matrices that preserve $\mathcal{L}$, i.e.,

$$\mathrm{Aut}(\mathcal{L}) = \{\varphi \in \mathrm{O}_n(\mathbb{R}) : \varphi(\mathcal{L}) = \mathcal{L}\}.$$

It is clear that $\mathrm{Aut}(\mathcal{L})$ contains the automorphisms $\pm\mathbf{I}_n$, called *trivial automorphisms* of $\mathcal{L}$.

The set of isomorphisms $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$ between two n-dimensional lattices $\mathcal{L}_1, \mathcal{L}_2$ consists of all orthogonal matrices that send $\mathcal{L}_1$ to $\mathcal{L}_2$, i.e.,

$$\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2) = \{\varphi \in \mathrm{O}_n(\mathbb{R}) : \varphi(\mathcal{L}_1) = \mathcal{L}_2\}.$$

If $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2) \neq \emptyset$, take an arbitrary element $\varphi \in \mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$. It is straightforward to see $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2) = \varphi \cdot \mathrm{Aut}(\mathcal{L}_1)$. Hence $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$ is either empty or a coset of $\mathrm{Aut}(\mathcal{L}_1)$ in $\mathrm{O}_n(\mathbb{R})$. Two lattices are said to be isomorphic, written $\mathcal{L}_1 \cong \mathcal{L}_2$, if $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2) \neq \emptyset$.

Lemma 2.1 ([42, Lemma 2.8])**.** *For any two isomorphic lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, we have:*

1) $\mathrm{Aut}(\mathcal{L}_1) \cong \mathrm{Aut}(\mathcal{L}_2)$. *For any $\varphi \in \mathrm{O}_n(\mathbb{R})$ such that $\mathcal{L}_2 = \varphi(\mathcal{L}_1)$, the map $\mathcal{F}$ defined by $\mathcal{F}(\phi) = \varphi \cdot \phi \cdot \varphi^{-1}$, for all $\phi \in \mathrm{Aut}(\mathcal{L}_1)$, is an isomorphism from $\mathrm{Aut}(\mathcal{L}_1)$ to $\mathrm{Aut}(\mathcal{L}_2)$.*
2) *There is a one-to-one correspondence between $\mathrm{Aut}(\mathcal{L}_1)$ and $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$. For any $\varphi \in \mathrm{O}_n(\mathbb{R})$ such that $\mathcal{L}_2 = \varphi(\mathcal{L}_1)$, the map $\mathcal{G}$ defined by $\mathcal{G}(\phi) = \varphi \cdot \phi$, for all $\phi \in \mathrm{Aut}(\mathcal{L}_1)$, is a bijection between $\mathrm{Aut}(\mathcal{L}_1)$ and $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$.*

For complexity-theoretic purposes, all algorithmic problems below use Gram matrices as inputs. Thus an input lattice is given by an integral Gram matrix $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, and isomorphisms are represented by coordinate change matrices $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$. This convention avoids bit-complexity issues caused by arbitrary real bases and also includes every integral lattice. The same representation is standard in the quadratic-form treatment of lattice isomorphism; see, e.g., [26,61,40]. For convenience and readability, we use explicit basis vectors only for basis-dependent notions such as Gram–Schmidt orthogonalization and Korkine–Zolotareff bases.

For Gram matrices, we use a different symbol for coordinate stabilizers. We write

$$\mathrm{Stab}(\mathbf{G}) = \{\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z}) : \mathbf{U}^\top \mathbf{G} \mathbf{U} = \mathbf{G}\}$$

for the stabilizer of the quadratic form $\mathbf{G}$, and for $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{S}_n^{>0}(\mathbb{Z})$,

$$\mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2) = \{\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z}) : \mathbf{U}^\top \mathbf{G}_1 \mathbf{U} = \mathbf{G}_2\}$$

for the corresponding coordinate isomorphisms, so that $\mathrm{Stab}(\mathbf{G}) = \mathrm{Iso}(\mathbf{G}, \mathbf{G})$. The one-argument form is a group; the two-argument form is either empty or a coset of $\mathrm{Stab}(\mathbf{G}_1)$. Throughout the rest of the paper, automorphism outputs lie in $\mathrm{Stab}(\mathbf{G})$ and isomorphism outputs lie in $\mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2)$; we keep $\mathrm{Aut}(\mathcal{L})$ and $\mathrm{Iso}(\mathcal{L}_1, \mathcal{L}_2)$ only for geometric statements about the represented lattices.

Lemma 2.2 ([42, Lemma 2.9]). *Let $\mathcal{L}$ be a lattice with basis $\mathbf{B}$, and let $\mathbf{G} = \mathbf{B}^\top\mathbf{B}$. Then $\mathrm{Stab}(\mathbf{G}) \cong \mathrm{Aut}(\mathcal{L})$. The map ϕ defined by $\phi(\mathbf{U}) = \mathbf{B}\mathbf{U}\mathbf{B}^{-1}$, for all $\mathbf{U} \in \mathrm{Stab}(\mathbf{G})$, is an isomorphism from $\mathrm{Stab}(\mathbf{G})$ to $\mathrm{Aut}(\mathcal{L})$.*

2.4 Computational Problems related to Lattice Isomorphism

We now define the computational problems used throughout the paper. We will use the Gram matrix representation for all problems, and all input matrices below are assumed to lie in $\mathcal{S}_n^{>0}(\mathbb{Z})$ and to have the appropriate dimension.

When a lattice $\mathcal{L}$ is represented by a Gram matrix $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, we use the norm induced by $\mathbf{G}$ on coordinate vectors. Namely, for $\mathbf{z} \in \mathbb{Z}^n$, we write

$$\|\mathbf{z}\|_{\mathbf{G}} = \sqrt{\mathbf{z}^\top\mathbf{G}\mathbf{z}}.$$

Equivalently, if $\mathbf{B}^\top\mathbf{B} = \mathbf{G}$ and $\mathcal{L} = \mathbf{B}\mathbb{Z}^n$, then the coordinate vector $\mathbf{z}$ represents the lattice vector $\mathbf{B}\mathbf{z}$, and

$$\|\mathbf{z}\|_{\mathbf{G}} = \|\mathbf{B}\mathbf{z}\|_2.$$

Accordingly, lattice parameters defined via Euclidean lengths can be written equivalently in terms of $\mathbf{G}$. For example,

$$\lambda_1(\mathcal{L}) = \lambda_1(\mathbf{G}) := \min_{\mathbf{z}\in\mathbb{Z}^n\setminus\{0\}} \|\mathbf{z}\|_{\mathbf{G}}.$$

Definition 2.3 (Decisional Shortest Vector Problem (γ-GapSVP)). *For $\gamma = \gamma(n) \geq 1$, the decision version of the γ-approximate Shortest Vector Problem is defined as follows. On input $(\mathbf{G}, r)$, where $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$ is the Gram matrix of an integral lattice $\mathcal{L}$ and $r \in \mathbb{Q}_{>0}$, decide whether*

- (YES instance) $\lambda_1(\mathbf{G}) \leq r$,
- (NO instance) $\lambda_1(\mathbf{G}) > \gamma r$,

when one of these cases is promised to hold. [7]

Definition 2.4 (Search Shortest Vector Problem (γ-SVP)). *For any approximation factor $\gamma = \gamma(n) \geq 1$, the search version of the γ-approximate Shortest Vector Problem is defined as follows. Given $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, output a nonzero coordinate vector $\mathbf{z} \in \mathbb{Z}^n$ such that $\|\mathbf{z}\|_{\mathbf{G}} \leq \gamma\lambda_1(\mathbf{G})$. In particular, when $\gamma = 1$, we recover the exact Shortest Vector Problem, denoted by* SVP.

Currently, the fastest algorithm for solving SVP is proposed by Aggarwal, Dadush, Regev and Stephens-Davidowitz.

Theorem 2.1 ([3]). *There is a randomized $2^{n+o(n)}$-time and space algorithm for* SVP.

[7] Although SVP is often stated with a lattice basis as input, this is only a matter of representation. The lattice reduction algorithms work through Gram matrix representations as well.

Next, we define the decision and search versions of the Lattice Isomorphism Problem.

Definition 2.5 (Decision Lattice Isomorphism Problem (DLIP)). *Given* $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{S}_n^{>0}(\mathbb{Z})$, DLIP *is to decide whether* $\mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2) \neq \emptyset$.

Definition 2.6 (Search Lattice Isomorphism Problem (SLIP)). *Given* $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{S}_n^{>0}(\mathbb{Z})$ *with* $\mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2) \neq \emptyset$, SLIP *is to output a matrix* $\mathbf{U} \in \mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2)$.

Next, we define two new computationally hard problems about lattice automorphisms. The first is a sampling problem SaLAP, which will play a key role in this work. The second is to find a set of generators of the automorphism group of a lattice LAGP, which corresponds to the graph automorphism problem GAP.

Definition 2.7 (Sampling Lattice Automorphism Problem (SaLAP)). *Given* $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, SaLAP *is to output a uniform sample* $\mathbf{U} \xleftarrow{\$} \mathrm{Stab}(\mathbf{G})$.

Definition 2.8 (Lattice Automorphism Group Problem (LAGP[8])). *Given* $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, LAGP *is to output a generating set of* $\mathrm{Stab}(\mathbf{G})$.

Next, we introduce two counting problems related to LIP.

Definition 2.9 (Counting Lattice Automorphism Problem (#LAP)). *Given* $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, #LAP *is to compute* $|\mathrm{Stab}(\mathbf{G})|$.

Definition 2.10 (Counting Lattice Isomorphism Problem (#LIP)). *Given* $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{S}_n^{>0}(\mathbb{Z})$, #LIP *is to compute* $|\mathrm{Iso}(\mathbf{G}_1, \mathbf{G}_2)|$.

Next, we introduce the definition of irreducible lattice, which plays a key role in our paper. In addition, we need to use the unique decomposition theorem proved by Eichler in 1952. We also propose the problem of determining whether a lattice is irreducible, together with its corresponding search version, LODP.

Definition 2.11 (Irreducible Lattices). *A full-rank lattice* $\mathcal{L}$ *is called irreducible if and only if there do not exist non-trivial sublattices* $\mathcal{L}_1 \neq \{0\}$ *and* $\mathcal{L}_2 \neq \{0\}$, *such that* $\mathcal{L} = \mathcal{L}_1 \oplus \mathcal{L}_2$.

Theorem 2.2 ([28,45]). *Every lattice* $\mathcal{L}$ *has a unique orthogonal decomposition* $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$ *with irreducible lattices* $\mathcal{L}_i$. *We denote these* $\mathcal{L}_i$ *as the irreducible component of* $\mathcal{L}$, *and we call such* $\mathcal{L}$ *an s-direct sum lattice.*

Definition 2.12 (Distinguish Irreducible Lattice Problem (DILP)). *Given* $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, DILP *is to decide whether the represented integral lattice is irreducible.*

Definition 2.13 (Lattice Orthogonal Decomposition Problem (LODP)). *Given* $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, LODP *is to output a unimodular matrix* $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ *and integral Gram matrices* $\mathbf{G}_1, \ldots, \mathbf{G}_s$ *such that* $\mathbf{U}^\top \mathbf{G} \mathbf{U} = \mathbf{G}_1 \oplus \cdots \oplus \mathbf{G}_s$, *where each* $\mathbf{G}_i$ *represents an irreducible lattice.*

[8] The reason we use LAGP instead of LAP here is that LAP has already been defined in [42] as the problem of finding a non-trivial automorphism of a given lattice.

2.5 Randomized Self Reduction for Lattice Isomorphism Problem

In this section, we introduce the randomized self-reduction algorithm for the lattice isomorphism problem, whose key technique is the Gaussian sampling algorithm over the lattice.

Let

$$\rho_s(\boldsymbol{y}) = \exp\left(-\pi\|\boldsymbol{y}\|^2/s^2\right), \qquad \boldsymbol{y} \in \mathbb{R}^n,$$

be the Gaussian function centered at the origin with parameter s. The discrete Gaussian distribution with parameter s over a lattice $\mathcal{L}$ is defined by

$$\mathcal{D}_{\mathcal{L},s}(\boldsymbol{y}) = \rho_s(\boldsymbol{y})/\rho_s(\mathcal{L}), \boldsymbol{y} \in \mathcal{L}.$$

For a set $A \subseteq \mathcal{L}$, we denote $\rho_s(A) = \sum_{\boldsymbol{x}\in A} \rho_s(\boldsymbol{x})$.

In [33], Gentry et al. present an efficient approach that produces a sample distribution that is statistically close to the $\mathcal{D}_{\mathcal{L},s}$ for sufficiently large parameter s. Furthermore, Brakerski et al. provide an algorithm that samples exactly according to $\mathcal{D}_{\mathcal{L},s}$ [21].

Lemma 2.3 ([21])**.** *Suppose $\mathcal{L}$ is a lattice of dimension n with a basis $\mathbf{B}$. Then there exists a polynomial sampling algorithm* SampleD *which inputs $\mathbf{B}$ and outputs a vector from $\mathcal{D}_{\mathcal{L},s}$ for any $s \geq \sqrt{\ln(2n+4)/\pi} \cdot \max_i \left\|\widetilde{\boldsymbol{b}}_i\right\|$.*

Intuitively, the randomized self-reduction for the lattice isomorphism problem process leverages the Gaussian sampling algorithm on lattices, which effectively conceals the details of the lattice basis while preserving the geometric structure of the lattice. Such an algorithm is used in theoretical reductions and cryptographic constructions [26,15,42,41] related to the lattice isomorphism problem.

The randomized self-reduction algorithm can be given in basis form [15] or in quadratic form [26]. For the sake of simplicity, we give a sketch of the algorithm in quadratic form below. Given a $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, we first apply the LLL algorithm to $\mathbf{G}$, eliminating the length information. Then, the Gaussian sampling algorithm in Lemma 2.3 is used to sample n linearly independent lattice vectors $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. By leveraging [52, Lemma 7.1] (which exploits the uniqueness of the Hermite Normal Form), these vectors are converted into the Gram matrix $\mathbf{G}'$, yielding the transition matrix $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$, such that $\mathbf{G}' = \mathbf{U}^\top\mathbf{G}\mathbf{U}$. The output is $(\mathbf{G}', \mathbf{U})$. For further details, please refer to [26].

Lemma 2.4 (Adapted from [26, Lemma 3.4])**.** *There exists a randomized polynomial algorithm $\mathcal{R}$ that takes any $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$ as input and outputs $\mathcal{R}(\mathbf{G}) = (\mathbf{R}, \mathbf{U})$ such that $(\mathbf{R} = \mathbf{U}^\top\mathbf{G}\mathbf{U}, \mathbf{U}) \in [\mathbf{G}] \times \mathrm{GL}_n(\mathbb{Z})$, with the following properties:*

- *For any quadratic form $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, $\mathbf{G}' \in [\mathbf{G}]$, then the marginal distributions of the first coordinate are identical for $\mathcal{R}(\mathbf{G})$ and $\mathcal{R}(\mathbf{G}')$;*
- *For any quadratic form $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$, conditioning on the first coordinate $\mathbf{G}''$ of $(\mathbf{G}'', \mathbf{U}) \leftarrow \mathcal{R}(\mathbf{G})$, $\mathbf{U}$ is uniformly distributed in* $\mathrm{Iso}(\mathbf{G}, \mathbf{G}'')$.

2.6 Group-Theoretic Algorithms

Next, we need to list some results for computational problems in group theory that will be used in this paper.

Theorem 2.3 ([12]). *There exist* BPP *(in fact, Las Vegas) algorithms for the following problems: given a generating set* $\{g_1, .., g_k\}$ *of a finite group of matrices in* $\mathrm{GL}_n(\mathbb{Q})$ *as input,*

(a) determine the order of the group $\langle g_1, .., g_k\rangle$*,*
(b) Check if an input matrix g *is a member of* $\langle g_1, .., g_k\rangle$ *or not.*

Theorem 2.4 ([9]). *Suppose that a group* G *is generated by* $\{g_1, \ldots, g_k\}$ *and* ℓ *is an upper bound on* $\log |G|$*,* $k \in \mathrm{poly}(\ell)$*. There is a sampling algorithm that takes* 1^ℓ*,* ϵ*, and* $\{g_1, \ldots, g_k\}$ *as input and runs in* $\mathrm{poly}(\ell, \log(1/\epsilon))$ *time*[9] *that produces each group element with probability* $(1 \pm \epsilon)/|G|$*.*

We recall the following standard result from group theory, which asserts that a logarithmic number of random elements is sufficient to generate the entire group with high probability.

Theorem 2.5 ([23]). *For any finite group* G*,* $\epsilon \in (0, 1)$ *and an integer* $k > \log_2(|G|) + \max\{3, 2\log_2(1/\epsilon)\}$ *we have*

$$\Pr_{g_1, g_2, \ldots, g_k \xleftarrow{\$}_{i.i.d.} G} [\langle g_1, g_2, \ldots, g_k\rangle = G] \geq 1 - \epsilon.$$

2.7 Basic Complexity Theory

Finally, we review the basic complexity theory that will be frequently used throughout this paper.

Reductions

A language L is polynomial-time Turing reducible to a language L', denoted $L \leq_T L'$, if there exists a polynomial-time algorithm for L that can make oracle calls to L', i.e., $L \in \mathsf{P}^{L'}$. In this paper, we only consider (randomized) polynomial Turing reductions, and omit the word 'polynomial' when referring to Turing reductions.

We now extend the types of problems a little bit, a decision, search or sampling problem M is Turing reducible to another problem M' if there exists a polynomial-time algorithm for solving M that can make oracle calls to an oracle for M'. A randomized Turing reduction from M to M' is a reduction that uses randomness and can make oracle calls to an oracle for M' to solve M correctly with probability at least $2/3$ for any instance of M.

[9] The original paper was computed under the computational model of "black box groups": group elements were encoded as $(0, 1)$-strings of uniform length, and an oracle performs group operations at unit cost.

Specifically, a reduction from a decision problem L to a sampling problem $\mathcal{S}$ is a randomized oracle algorithm R^I such that for every oracle I that solves $\mathcal{S}$, R^I decides L with probability at least $2/3$ over the randomness of R^I. That is, if $x \in L$, $\Pr[R^I(x) = 1] \geq 2/3$, if $x \notin L$, $\Pr[R^I(x) = 1] \leq 1/3$.

Let R^I be a reduction, I is an oracle of the sampling problem $\mathcal{S}$. For an input z, a valid transcript of $R^I(z)$ is a string of the form $(r, x_1, x_2, \ldots, x_q)$, where r is the randomness of the reduction and $x_1, \ldots, x_q$ are the oracle responses in the order produced by I. We will assume, without loss of generality, that the length of r and the number of queries q depend only on the length of z.

Interactive Proof Systems and Arthur-Merlin Protocols

Definition 2.14 (Interactive Proofs (IP)). *The class* $\mathsf{IP}[k]$ *consists of languages L for which there exists an interactive proof system between a probabilistic polynomial-time verifier (Verifier) and an all-powerful prover (Prover). The Verifier and Prover exchange k rounds of messages, where the Verifier uses polynomial-time computation and private randomness. Two properties are required for the protocol:*

- *(Completeness) If $x \in L$, there exists a strategy for Prover such that* $\Pr[\textit{Verifier accepts}] \geq 2/3$.
- *(Soundness) If $x \notin L$, for all strategies of Prover,* $\Pr[\textit{Verifier accepts}] \leq 1/3$.

The error bounds can be made exponentially small through standard amplification techniques (at a cost of increasing rounds).

If we require Verifier to use public randomness, the class corresponds to AM (Arthur-Merlin) protocols, which are a special case of interactive proofs.

Definition 2.15 (Arthur–Merlin (AM)). *The class* AM *consists of languages L for which there exists a two-round public-coin interactive proof system between a randomized polynomial-time verifier (Arthur) and an unbounded prover (Merlin). On input x, Arthur first samples random coins r and sends them to Merlin; Merlin then replies with a message m, and Arthur decides whether to accept or reject by running a polynomial-time verifier $V(x, r, m)$.*

The protocol satisfies the following properties:

- Completeness. *If $x \in L$, then there exists a (possibly randomized) strategy M for Merlin such that*

$$\Pr_r[V(x, r, M(r)) = 1] \geq \tfrac{2}{3}.$$

- Soundness. *If $x \notin L$, then for every strategy M of Merlin,*

$$\Pr_r[V(x, r, M(r)) = 1] \leq \tfrac{1}{3}.$$

The error bounds can be made exponentially small through standard amplification techniques.

It is known that $\mathsf{AM} := \mathsf{AM}[2] = \mathsf{AM}[k]$ for any constant $k \geq 2$ [11], where $\mathsf{AM}[k]$ denotes a k-round Arthur-Merlin protocol. Moreover, it is shown in [36] that $\mathsf{IP}[k] \subseteq \mathsf{AM}[k+2]$ for any $k : \mathbb{N} \to \mathbb{N}$ with $k(n)$ computable in $\mathrm{poly}(n)$, where $\mathsf{IP}[k]$ denotes an interactive proof system with k rounds of interaction.

The class coAM is the complement of AM, consisting of languages whose complements are in AM. It is known that if $\mathsf{NP} \subseteq \mathsf{coAM}$, then PH collapses to the second level, i.e., $\mathsf{PH} = \Sigma_2^P$ [20]. Hence, it is not likely that coAM contains a complete language for NP.

In the sequel, we will make use of the following standard tool due to Goldwasser and Sipser [36].

The Goldwasser–Sipser lower-bound protocol. Goldwasser and Sipser [36] introduced a public-coin Arthur–Merlin protocol for certifying lower bounds on the size of an implicitly defined finite set. Let S be a finite set whose membership can be certified[10]. For any threshold t, there exists a two-round public-coin AM protocol that decides the following promise problem with constant completeness and soundness gap[11]:

- **YES instances:** $|S| \geq t$;
- **NO instances:** $|S| \leq t/2$.

3 On Basing SLIP on NP-hardness

In this section, we will prove our result Theorem 1.1. First, we establish a fundamental lemma related to irreducible lattices. In our proof, we will rely on the unique orthogonal decomposition Theorem 2.2 [28,45], and we find a new algorithmic proof for the theorem via KZ basis with improved running time (see Theorem 3.2).

Lemma 3.1 (Isomorphisms of orthogonal direct sums)**.** *Let $\mathcal{A} = \bigoplus_{i=1}^{r} \mathcal{A}_i$ and $\mathcal{B} = \bigoplus_{j=1}^{s} \mathcal{B}_j$ be n-dimensional full-rank lattices with orthogonal decompositions of irreducible lattices $\{\mathcal{A}_i\}_{i\in[r]}$ of dimension n_i and $\{\mathcal{B}_j\}_{j\in[s]}$ of dimension m_j. Clearly $n = \sum_{i=1}^{r} n_i = \sum_{j=1}^{s} m_j$.*

If there exists a lattice isomorphism $\phi \in \mathrm{O}_n(\mathbb{R}) : \mathcal{A} \to \mathcal{B}$, then $r = s$, and there exists a permutation $\sigma : [r] \to [r]$ such that $n_i = m_{\sigma(i)}$, and isomorphisms $\phi_i : \mathcal{A}_i \to \mathcal{B}_{\sigma(i)}$ for each $i \in [r]$ such that $\phi = \bigoplus_{i=1}^{r} \phi_i$.

Proof. The unique orthogonal decompositions of $\mathcal{A}$ and $\mathcal{B}$ are given by $\mathcal{A} = \bigoplus_{i=1}^{r} \mathcal{A}_i$ and $\mathcal{B} = \bigoplus_{j=1}^{s} \mathcal{B}_j$.

[10] We assume that membership in S admits a constant-round public-coin AM protocol. Under this assumption, a Goldwasser–Sipser-style argument yields a constant-round public-coin protocol for distinguishing $|S| \geq t$ from $|S| \leq t/c$ for any constant $c > 1$. By the collapse $\mathsf{AM}[k] = \mathsf{AM}$ for constant k, the resulting promise problem lies in AM.

[11] The original formulation in [36] uses a gap of t versus $t/2$, but any constant-factor gap suffices.

Consider the decomposition of $\mathcal{B}$ given by the image of $\mathcal{A}$ as follows: $\mathcal{B} = \phi(\mathcal{A}) = \phi(\bigoplus_{i=1}^{r} \mathcal{A}_i) = \bigoplus_{i=1}^{r} \phi(\mathcal{A}_i)$. Note that since ϕ is an orthogonal transformation, it preserves inner products and the orthogonal decomposition $\bigoplus_{i=1}^{r} \phi(\mathcal{A}_i)$.

Furthermore, each $\phi(\mathcal{A}_i)$ must be irreducible. If $\phi(\mathcal{A}_i)$ had a non-trivial orthogonal decomposition $\phi(\mathcal{A}_i) = \mathcal{D}_1 \oplus \mathcal{D}_2$, then $\mathcal{A}_i = \phi^{-1}(\mathcal{D}_1) \oplus \phi^{-1}(\mathcal{D}_2)$ would also be a non-trivial orthogonal decomposition, contradicting the irreducibility of $\mathcal{A}_i$. Therefore, $\bigoplus_{i=1}^{r} \phi(\mathcal{A}_i)$ is an orthogonal decomposition of $\mathcal{B}$ into irreducible components. By the uniqueness part of Theorem 2.2, this decomposition must match the given decomposition $\mathcal{B} = \bigoplus_{j=1}^{s} \mathcal{B}_j$ up to reordering.

Thus, $r = s$, and there exists a permutation $\sigma : [r] \to [r]$ such that $\phi(\mathcal{A}_i) = \mathcal{B}_{\sigma(i)}$ for each $1 \le i \le r$. Since the isomorphic lattices have the same rank, we must have $n_i = m_{\sigma(i)}$ for all i. For each $1 \le i \le r$, define $\phi_i = \phi|_{\mathcal{A}_i \otimes \mathbb{R}} : \mathcal{A}_i \to \mathcal{B}_{\sigma(i)}$ as the restriction of ϕ to the subspace spanned by $\mathcal{A}_i$. Then $\phi = \bigoplus_{i=1}^{r} \phi_i$, as desired. □

The preceding lemma shows that irreducible lattices can only be isomorphic to other irreducible lattices. A direct consequence is the following corollary, which describes the automorphism group structure for a direct sum of such lattices.

Corollary 3.1. *For any k-dimensional irreducible lattice $\mathcal{L}$ and any positive integer m, we have* $\mathrm{Aut}(\bigoplus_{i=1}^{m} \mathcal{L}) = \{(\mathbf{S} \otimes \mathbf{I}_k) \cdot \mathrm{diag}(\mathbf{A}_1, \cdots, \mathbf{A}_m) \mid \mathbf{S} \in \mathcal{S}_m, \mathbf{A}_i \in \mathrm{Aut}(\mathcal{L}), i \in [m]\}$.

Next, we use the above lemma to prove the randomized Turing reduction from SLIP to SaLAP.

Proposition 3.1. *There is a randomized Turing reduction from* SLIP *to* SaLAP.

Proof. Let $(\mathbf{G}, \mathbf{G}')$ be the given SLIP instance, where $\mathrm{Iso}(\mathbf{G}, \mathbf{G}') \neq \emptyset$, and let $\mathcal{L} \cong \mathcal{L}'$ be represented lattices of rank n. We will use lattice notation below for readability. However, we also emphasize that the reduction itself only manipulates Gram matrices. In particular, the direct sum is represented by the block diagonal form $\mathbf{N} = \mathbf{G} \oplus \mathbf{G}'$, and every call to SaLAP is made on such Gram matrix $\mathbf{N}$, returning uniformly random transition matrix sampled from $\mathrm{Stab}(\mathbf{N})$ [12].

From Theorem 2.2, assume that the unique orthogonal decomposition of $\mathcal{L}$ is $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_k$, where $\mathcal{L}_i$, $i \in [k]$, are irreducible lattices and $1 \le k \le n$. Let $\mathcal{N} = \mathcal{L} \oplus \mathcal{L}'$. According to Lemma 3.1 and Corollary 3.1, for any $\phi \in \mathrm{Aut}(\mathcal{N})$, $\phi(\mathcal{L}_1)$ is an irreducible component of $\mathcal{N}$. Since $\mathcal{L} \cong \mathcal{L}'$, for a random $\phi \xleftarrow{\$} \mathrm{Aut}(\mathcal{N})$,

$$\Pr_{\phi \xleftarrow{\$} \mathrm{Aut}(\mathcal{N})} [\phi(\mathcal{L}_1) \subseteq \mathcal{L}'] = \frac{1}{2}.$$

[12] Moreover, a sublattice of a lattice with Gram matrix $\mathbf{H}$ is represented by an integer coordinate matrix $\mathbf{C}$, with induced Gram matrix $\mathbf{C}^\top \mathbf{H} \mathbf{C}$. The sum, intersection and orthogonal complement of sublattices can be obtained by computing, via Hermite normal form.

If this event occurs, then $\mathrm{rank}(\phi(\mathcal{L}) \cap \mathcal{L}) \leq n-1$. Thus, an automorphism of $\mathcal{N}$ can be used to obtain a partial orthogonal decomposition of $\mathcal{L}$ and $\mathcal{L}'$, with a probability of at least $1/2$.

By Lemma 3.1, after writing the orthogonal decomposition of $\mathcal{N}$ induced by $\mathcal{L} \cong \mathcal{L}'$ and the automorphism ϕ, we have

$$\begin{aligned}
\mathcal{L} &= (\phi(\mathcal{L}') \cap \mathcal{L}) \oplus (\phi(\mathcal{L}) \cap \mathcal{L}), \\
\mathcal{L}' &= (\phi(\mathcal{L}) \cap \mathcal{L}') \oplus (\phi(\mathcal{L}') \cap \mathcal{L}'), \\
\phi(\mathcal{L}') \cap \mathcal{L} &\cong \phi(\mathcal{L}) \cap \mathcal{L}', \\
\phi(\mathcal{L}') \cap \mathcal{L}' &\cong \phi(\mathcal{L}) \cap \mathcal{L}.
\end{aligned}$$

Therefore, if we find the isomorphisms $\psi_0 \in \mathrm{Iso}(\phi(\mathcal{L}') \cap \mathcal{L}, \phi(\mathcal{L}) \cap \mathcal{L}')$ and $\psi_1 \in \mathrm{Iso}(\phi(\mathcal{L}) \cap \mathcal{L}, \phi(\mathcal{L}') \cap \mathcal{L}')$, then we get an isomorphism $\psi = \psi_0 \oplus \psi_1 \in \mathrm{Iso}(\mathcal{L}, \mathcal{L}')$.

In addition, if $\phi \in \mathrm{Aut}(\mathcal{N})$ satisfies $\phi(\mathcal{L}) = \mathcal{L}'$, we call ϕ a "lucky" automorphism. Its restriction to the first summand gives, in coordinate form, an element of $\mathrm{Iso}(\mathbf{G}, \mathbf{G}')$. If $\mathcal{L}$ is irreducible (in other words $k = 1$), the proportion of lucky automorphisms in $\mathrm{Aut}(\mathcal{N})$ is $1/2$. Indeed,

$$\mathrm{Aut}(\mathcal{N}) = \left\{ \begin{pmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{pmatrix} : \substack{\mathbf{A}_1 \in \mathrm{Aut}(\mathcal{L}) \\ \mathbf{A}_2 \in \mathrm{Aut}(\mathcal{L}')} \right\} \bigcup \left\{ \begin{pmatrix} 0 & \mathbf{B}_1 \\ \mathbf{B}_2 & 0 \end{pmatrix} : \substack{\mathbf{B}_1 \in \mathrm{Iso}(\mathcal{L}, \mathcal{L}') \\ \mathbf{B}_2 \in \mathrm{Iso}(\mathcal{L}', \mathcal{L})} \right\},$$

since Lemma 3.1 and Corollary 3.1 imply that any automorphism of $\mathcal{N}$ must map $\mathcal{L}$ to either $\mathcal{L}$ or $\mathcal{L}'$.

Therefore, the overall reduction process is as follows: we call the SaLAP on lattice $\mathcal{N}$ to generate $m = \mathrm{poly}(n)$ independently and identically distributed $\mathbf{U}_1, \ldots, \mathbf{U}_m \xleftarrow{\$}_{i.i.d.} \mathrm{Stab}(\mathbf{N})$, which corresponds to $\phi_1, \ldots, \phi_m \xleftarrow{\$}_{i.i.d.} \mathrm{Aut}(\mathcal{N})$ in the lattice view. Then, we determine whether there is a "lucky" automorphism in $\{\phi_1, \ldots, \phi_m\}$ ($\mathrm{rank}(\phi_i(\mathcal{L}) \cap \mathcal{L}) = 0$), if true, output the isomorphism $\phi_i \mid_{\mathcal{L}} \in \mathrm{Iso}(\mathcal{L}, \mathcal{L}')$ directly. When $\mathcal{L}$ is irreducible, this occurs with probability larger than $1 - 2^{-m}$. If $\mathcal{L}$ is not irreducible, then determine whether $\exists\, \phi_i$, $i \in [m]$ such that $1 \leq \mathrm{rank}(\phi_i(\mathcal{L}) \cap \mathcal{L}) \leq n-1$, this occurs with a probability at least $1 - 2^{-m}$.

Thus, in the second case, we reduce the k-direct sum SLIP instance $(\mathcal{L}, \mathcal{L}')$ to two SLIP instances $(\phi_i(\mathcal{L}) \cap \mathcal{L}, \phi_i(\mathcal{L}') \cap \mathcal{L}')$ and $(\phi_i(\mathcal{L}') \cap \mathcal{L}, \phi_i(\mathcal{L}) \cap \mathcal{L}')$, where $\phi_i(\mathcal{L}) \cap \mathcal{L}$ is a k_1-direct sum lattice, $\phi_i(\mathcal{L}') \cap \mathcal{L}$ is a k_2-direct sum lattice and $k_1 + k_2 = k$, $1 \leq k_1, k_2 \leq k-1$. Therefore, iteratively using the same method for these two lower-direct sum SLIP instances in the above paragraph, we can find $\psi \in \mathrm{Iso}(\mathcal{L}, \mathcal{L}')$ with a probability of $1 - \mathrm{negl}(n)$ by calling the SaLAP at most $2nm$ times. □

In fact, in the above reduction process, we have proved the following corollary.

Corollary 3.2. *There is a randomized Turing reduction from* LODP *to* SaLAP.

The following proposition means that, for a Gram matrix $\mathbf{G}$, the order of $\mathrm{Stab}(\mathbf{G})$ is *verifiable* in AM.

IP Protocol for AutVer

Common Input: $(\mathbf{G}, s) \in \mathcal{S}_n^{>0}(\mathbb{Z}) \times \mathbb{Z}$, where $s \leq 2^n \times n!$ is given in binary.

1. **Prover:** Output a generating set $\{g_1, g_2, \ldots, g_\ell\}$ of $\mathrm{Stab}(\mathbf{G})$ to the Verifier, where $g_i \in \mathrm{GL}_n(\mathbb{Z})$, $i \in [\ell]$, $\ell = \Theta(n^2)$.
2. **Verifier:** Verify that each $g_i, i \in [\ell]$ is in $\mathrm{Stab}(\mathbf{G})$ and use the algorithm in Theorem 2.3 to calculate whether s is equal to the order of $\langle g_1, \ldots, g_\ell \rangle$ (with a probability greater than $1 - 2^{-\Theta(n)}$).
3. **Verifier:** Calculate $\mathbf{G}'$ using randomized algorithm $\mathcal{R}$ in Lemma 2.4 and send it to the Prover, where $\mathbf{G}' = \mathbf{U}^\top \mathbf{G} \mathbf{U}$, $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$.
4. **Prover:** Output an isomorphism $\mathbf{V} \in \mathrm{GL}_n(\mathbb{Z})$ from $\mathbf{G}$ to $\mathbf{G}'$, that is, $\mathbf{V}^\top \mathbf{G} \mathbf{V} = \mathbf{G}'$.
5. **Verifier:** Check $\mathbf{V}$ is an isomorphism from $\mathbf{G}$ to $\mathbf{G}'$, then calculate $g_{\ell+1} = \mathbf{U}\mathbf{V}^{-1} \in \mathrm{Stab}(\mathbf{G})$ and verify membership of $g_{\ell+1}$ in the group $\langle g_1, \ldots, g_\ell \rangle$ using the algorithm from Theorem 2.3 (with a probability greater than $1 - 2^{-\Theta(n)}$).
6. **Verifier, Prover:** Repeat steps $3 - 5$ once.

Fig. 2. IP Protocol for AutVer

Proposition 3.2. *Let*

$$\mathsf{AutVer} := \{(\mathbf{G}, s) \mid \mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z}) \ \wedge \ |\mathrm{Stab}(\mathbf{G})| = s\}.$$

We show that deciding membership in AutVer *is in* AM.

Proof. In this protocol, we work directly with the quadratic form $\mathbf{G} \in \mathcal{S}_n^{>0}(\mathbb{Z})$. This avoids representation issues. Feit [29] showed that for $n > 10$, $\mathbb{Z}^n$ has the largest automorphism group among all n-dimensional lattices, with $|\mathrm{Aut}(\mathbb{Z}^n)| = |\mathrm{Stab}(\mathbf{I}_n)| = 2^n \cdot n!$. Therefore, s is at most $2^{O(n \log n)}$ and can be represented efficiently.

We now describe the protocol in detail and analyze its correctness as in Fig. 2.

In step 1, the prover sends a purported generating set $\{g_1, g_2, \ldots, g_\ell\}$ represented in $\mathrm{GL}_n(\mathbb{Z})$ for $\mathrm{Stab}(\mathbf{G})$, where $\ell = \Theta(n^2)$. A legitimate prover is indeed able to send an ℓ-size generating set, since any finite group has a generating set of logarithmic size in its order by Theorem 2.5.

In step 2, the verifier checks that each $g_i \in \mathrm{Stab}(\mathbf{G})$ by verifying $g_i^\top \mathbf{G} g_i = \mathbf{G}$. Then the verifier uses the algorithm from Theorem 2.3 to calculate the order of the group $\langle g_1, \ldots, g_\ell \rangle$ and checks if it equals s. This can be done with probability of correctness at least $1 - 2^{-\Theta(n)}$ and also in polynomial time.

In steps $3 - 6$, the protocol verifies that the prover cannot extend the generating set, which ensures that $\{g_1, g_2, \ldots, g_\ell\}$ generates the full group $\mathrm{Stab}(\mathbf{G})$. This is done by:

1. The verifier randomly transforms $\mathbf{G}$ to $\mathbf{G}'$ using the randomized polynomial algorithm $\mathcal{R}$ in Lemma 2.4, which produces a new quadratic form that is isomorphic to $\mathbf{G}$.

2. The prover provides an isomorphism $\mathbf{V}$ from $\mathbf{G}$ to $\mathbf{G}'$.
3. The verifier computes a new element $g_{\ell+1} = \mathbf{U}\mathbf{V}^{-1} \in \mathrm{Stab}(\mathbf{G})$, where $\mathbf{U}$ is the transformation matrix used in step 3.
4. The verifier verifies membership of $g_{\ell+1}$ in the group $\langle g_1, \ldots, g_\ell \rangle$ using the BPP algorithm from Theorem 2.3.

Clearly the verifier runs in polynomial time of the input length, so we now focus on the completeness and soundness of the protocol.
Completeness: If $(\mathbf{G}, s) \in \mathsf{AutVer}$, i.e., $|\mathrm{Stab}(\mathbf{G})| = s$, then:

1. The prover can provide a valid generating set $\{g_1, \ldots, g_\ell\}$ of $\mathrm{Stab}(\mathbf{G})$ in step 1.
2. The verifier will confirm in step 2 that the order of $\langle g_1, \ldots, g_\ell \rangle$ equals s with probability at least $1 - 2^{-\Theta(n)}$.
3. In step 3, the randomized algorithm $\mathcal{R}$ produces $\mathbf{G}'$ that is isomorphic to $\mathbf{G}$.
4. In step 4, the prover can provide a valid isomorphism $\mathbf{V}$.
5. In step 5, since $\{g_1, \ldots, g_\ell\}$ already generates the full group $\mathrm{Stab}(\mathbf{G})$, $g_{\ell+1}$ should be in $\langle g_1, \ldots, g_\ell \rangle$.

Therefore, $\Pr[\text{Verifier accepts}] \geq 1 - 2^{-\Theta(n)}$.
Soundness: If $(\mathbf{G}, s) \notin \mathsf{AutVer}$, i.e., $s \neq |\mathrm{Stab}(\mathbf{G})|$, then:

1. In step 2, the verifier checks the order of $\langle g_1, \ldots, g_\ell \rangle$ equals s and also $\langle g_1, \ldots, g_\ell \rangle \subseteq \mathrm{Stab}(\mathbf{G})$. Then the prover passes step 2 with high probability only if $\langle g_1, \ldots, g_\ell \rangle$ has order s and is a subgroup of $\mathrm{Stab}(\mathbf{G})$. Thus, $|\mathrm{Stab}(\mathbf{G})|/s \in \mathbb{Z}_{>1}$ (cannot be 1 by the assumption $s \neq |\mathrm{Stab}(\mathbf{G})|$).
2. In step 3, by the property of Lemma 2.4, when given the output $\mathbf{G}'$, $\mathbf{U}$ is uniformly distributed in $\mathrm{Iso}(\mathbf{G}, \mathbf{G}')$.
3. In step 5, conditioned on any output of the Prover, $g_{\ell+1} = \mathbf{U}\mathbf{V}^{-1}$ is uniformly distributed in $\mathrm{Stab}(\mathbf{G})$, since $\mathbf{U}$ is uniformly distributed in $\mathrm{Iso}(\mathbf{G}, \mathbf{G}')$ conditioned on $\mathbf{V}$ being an isomorphism from $\mathbf{G}$ to $\mathbf{G}'$. Therefore, the probability that $g_{\ell+1}$ is in the subgroup generated by $\langle g_1, \ldots, g_\ell \rangle$ is at most $|\langle g_1, \ldots, g_\ell \rangle|/|\mathrm{Stab}(\mathbf{G})| = s/|\mathrm{Stab}(\mathbf{G})| \leq \frac{1}{2}$.
4. By repeating steps $3-5$ once, the probability that both $g_{\ell+1}$ and another such element are in $\langle g_1, \ldots, g_\ell \rangle$ is at most $\frac{1}{4} + 2^{-\Theta(n)} \leq \frac{1}{3}$ for sufficiently large n.

Since $\mathsf{IP}[k] \subseteq \mathsf{AM}[k+2]$ for any $k : \mathbb{N} \to \mathbb{N}$ with $k(n)$ computable in $\mathrm{poly}(n)$ [36] and $\mathsf{AM}[k] = \mathsf{AM}$ for any constant $k \geq 2$ [11], we conclude that AutVer is in AM. □

Next, we present the proof of the main theorem. As outlined in Section 1.1, the key idea is to adapt the set lower bound protocol of Goldwasser and Sipser [36]. For each input $z \in L$, we define a set $C(z)$ whose membership can be verified in AM on input z. Moreover, the construction ensures that if $z \in L$, then $|C(z)|$ exceeds $2/3$ of a fixed public constant, while if $z \notin L$, its size is at most $1/2$ of that constant. Concretely, $C(z)$ consists of all tuples $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p)$, where $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q)$ forms a valid accepting transcript of $R^{\mathcal{O}}(z)$, and p ranges over a carefully chosen interval that enforces the required size gap.

Theorem 3.1. *If there exists a randomized Turing reduction from a decision problem L to* SLIP*, then* $L \in \mathsf{AM} \cap \mathsf{coAM}$.

Proof. By Proposition 3.1, if there exists a randomized Turing reduction from a decision problem L to SLIP, then there exists a randomized Turing reduction from L to SaLAP. Let $R^{\mathcal{O}}$ be such a reduction, where the SaLAP oracle $\mathcal{O}$ works as follows: on query a Gram matrix $\mathbf{G}$, return a matrix $\mathbf{U}$ chosen uniformly at random from $\mathrm{Stab}(\mathbf{G})$. Without loss of generality, we assume that the number of queries q is a fixed polynomial in $|z|$, which can be ensured by padding with meaningless queries to the trivial lattice $\mathbb{Z}$. Similarly, we assume that the length of r, denoted $|r|$, is a fixed polynomial in $|z|$, which can be achieved by padding with additional unused random coins if necessary.

Let the language C consist of all tuples $(z, r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p)$ such that $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q)$ is an accepting valid transcript of $R^{\mathcal{O}}(z)$, and p is an integer between 1 and $\lceil K/(s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)) \rceil$. Here,

- $\mathbf{G}_i$ is the i-th Gram-matrix query of the reduction,
- $s(\mathbf{G}) = |\mathrm{Stab}(\mathbf{G})|$,
- $K = 2^{ql+3}$, where l is a predetermined polynomial in $|z|$, greater than the running time of the reduction R. Hence, l serves as an upper bound on the length of answers that $\mathcal{O}$ replies to each query $R^{\mathcal{O}}$ on inputs of length $|z|$ asks.

Claim. The language C is in AM.

Proof of Claim. On input $(z, r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p)$, the AM verifier for C runs the reduction on input z with randomness r, simulates oracle answers using $\mathbf{U}_i$ sequentially and records all queries $\mathbf{G}_i$. It checks for each query $\mathbf{G}_i$ that the reduction makes, the answer $\mathbf{U}_i$ is indeed in $\mathrm{Stab}(\mathbf{G}_i)$ and finally the reduction algorithm R accepts. Then, the verifier asks the prover to provide $s(\mathbf{G}_1), \ldots, s(\mathbf{G}_q)$. Both parties then run the AM protocol for AutVer in parallel to verify that the numbers $s(\mathbf{G}_1), \ldots, s(\mathbf{G}_q)$ provided by the prover are correct. Finally, the Verifier checks $1 \leq p \leq \lceil K/(s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)) \rceil$ and accepts if all checks pass. For a detailed protocol, see Fig. 3.

The verifier runs in polynomial time of input: the efficiency of Steps 2 and 5 follows from the fact that R is a randomized polynomial-time Turing reduction from L to SaLAP, while the efficiency of Step 4 is guaranteed by Proposition 3.2. And the completeness of the protocol is immediate. We briefly check soundness as follows:

Soundness: If $(z, r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p) \notin C$, then at least one of the following hold:

1. The transcript $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q)$ is invalid, or not accepted for $R^{\mathcal{O}}(z)$, or for some $i \in [q]$, $\mathbf{U}_i \notin \mathrm{Stab}(\mathbf{G}_i)$, in which case the verifier rejects in step 2.
2. The prover provides incorrect sizes for the automorphism groups in step 4. The soundness of the AutVer protocol ensures that for a single $i \in [q]$, the probability that the verifier accepts an incorrect claim about $|\mathrm{Stab}(\mathbf{G}_i)|$ is exponentially small. Therefore, the overall probability of accepting is exponentially small, since q is polynomial in $|z|$.

AM Protocol for C

1. **Common Input:** $(z, r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p)$.
2. **Verifier:** Run the reduction algorithm R on input z with randomness r, simulating the oracle answers with $\mathbf{U}_1, \ldots, \mathbf{U}_q$ in sequence using common input, while recording each query $\mathbf{G}_i$ made by R. Verify the answer $\mathbf{U}_i$ is indeed in $\mathrm{Stab}(\mathbf{G}_i)$ and that the reduction algorithm R accepts with the given transcripts $\mathbf{U}_i$.
3. **Prover:** Run the reduction algorithm R on input z with randomness r, simulating the oracle answers with $\mathbf{U}_1, \ldots, \mathbf{U}_q$ in sequence. Record each query $\mathbf{G}_i$ made by R. Output $s(\mathbf{G}_i) = |\mathrm{Stab}(\mathbf{G}_i)|$ for each $1 \leq i \leq q$.
4. **Prover, Verifier:** Run the AM protocol (Proposition 3.2) for AutVer (with two-sided exponential small error in $|z|$) in parallel to check $s(\mathbf{G}_1), \ldots, s(\mathbf{G}_q)$ provided by the prover are correct.
5. **Verifier:** Computes $\lceil K/(s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)) \rceil$ and checks that $1 \leq p \leq \lceil K/(s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)) \rceil$.

Fig. 3. AM Protocol for C

3. $p < 1$ or $p > \left\lceil \frac{K}{s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)} \right\rceil$, in which case verifier rejects in step 5.

Hence, we conclude that the AM protocol has soundness error $1/3$ and hence $C \in \mathsf{AM}$. □

Let $C(z)$ denote the set of all $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p)$ such that $(z, r, \mathbf{U}_1, \ldots, \mathbf{U}_q, p) \in C$.

Claim. $C(z)$ has size at least $\frac{2}{3}2^{|r|}K$ if $z \in L$, and size at most $\frac{1}{2}2^{|r|}K$ if $z \notin L$.

Proof of Claim. Fix the input z. Conditioned on the randomness r, every valid transcript $(r, \mathbf{U}_1, \ldots, \mathbf{U}_q)$ appears with probability exactly $1/(s(\mathbf{G}_1) \cdots s(\mathbf{G}_q))$ over the choice of randomness of the SaLAP oracle $\mathcal{O}$. These conditional probabilities sum to 1 over all valid answer sequences. Consequently,

$$\Pr[R^{\mathcal{O}}(z) \text{ accepts}] = 2^{-|r|} \sum_{\substack{(r, \mathbf{U}_1, \ldots, \mathbf{U}_q) \\ \text{accepting valid transcript}}} \frac{1}{s(\mathbf{G}_1) \cdots s(\mathbf{G}_q)}.$$

If $z \in L$, then the accepting valid transcripts have total probability at least $2/3$ over the joint choice of r and oracle randomness. Hence,

$$\begin{aligned}
|C(z)| &= \sum_{\substack{(r,\mathbf{U}_1,\ldots,\mathbf{U}_q) \\ \text{accepting valid transcript}}} \left\lceil \frac{K}{s(\mathbf{G}_1)\cdots s(\mathbf{G}_q)} \right\rceil \\
&\geq K \sum_{\substack{(r,\mathbf{U}_1,\ldots,\mathbf{U}_q) \\ \text{accepting valid transcript}}} \frac{1}{s(\mathbf{G}_1)\cdots s(\mathbf{G}_q)} \\
&\geq \frac{2}{3} 2^{|r|} K.
\end{aligned}$$

If $z \notin L$, then the accepting valid transcripts have total probability at most $1/3$. Also, for each fixed r, there are at most 2^{ql} possible answer strings. Therefore,

$$\begin{aligned}
|C(z)| &\leq \sum_{\substack{(r,\mathbf{U}_1,\ldots,\mathbf{U}_q) \\ \text{accepting valid transcript}}} \left(\frac{K}{s(\mathbf{G}_1)\cdots s(\mathbf{G}_q)} + 1 \right) \\
&\leq K \sum_{\substack{(r,\mathbf{U}_1,\ldots,\mathbf{U}_q) \\ \text{accepting valid transcript}}} \frac{1}{s(\mathbf{G}_1)\cdots s(\mathbf{G}_q)} + \sum_{\substack{(r,\mathbf{U}_1,\ldots,\mathbf{U}_q) \\ \text{accepting valid transcript}}} 1 \\
&\leq \frac{1}{3} 2^{|r|} K + 2^{|r|} 2^{ql} \\
&= \left(\frac{1}{3} + \frac{1}{8} \right) 2^{|r|} K \leq \frac{1}{2} 2^{|r|} K,
\end{aligned}$$

by the choice $K = 2^{ql+3}$. □

Thus, using the set lower bound protocol of Goldwasser and Sipser [36] and also the fact that $C \in \mathsf{AM}$, we conclude that L is in AM. On the other hand, if L is randomized Turing reducible to SaLAP, then its complement $\overline{L}$ is also randomized Turing reducible to SaLAP: use the same reduction algorithm $R^{\mathcal{O}}$ but finally output the opposite answer. Hence, it follows that L is also in coAM. □

Corollary 3.3. *If there exists a randomized Turing reduction from* SVP *to* SLIP*, then* $\mathsf{NP} \subseteq \mathsf{coAM}$*.*

Proof. In [6], Ajtai showed that GapSVP is NP-hard, and $\mathsf{GapSVP} \leq \mathsf{SVP}$. Hence, by Theorem 3.1, if there is a randomized reduction from SVP to SLIP, then $\mathsf{NP} \subseteq \mathsf{coAM}$. □

Unique Orthogonal Decomposition Algorithm via KZ basis

In the above proof, we relied on the unique orthogonal decomposition theorem for lattices (Theorem 2.2, [28,45]). In this subsection, we give a new algorithmic

proof of the theorem and highlight the role of the KZ basis in revealing irreducible components.

While algorithmic approaches to lattice decomposition have been studied before [45,39,54], existing methods require enumerating all lattice vectors of length below a threshold B (with $B \geq \lambda_n$).

Our approach avoids this computational bottleneck by bypassing the search for short decomposable vectors, and instead uncovers a novel structural connection between KZ bases and irreducible decompositions.

Theorem 3.2. *Let $\mathcal{L}$ be a full-rank lattice with a* KZ *basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$. Define a graph $G_{\mathcal{L}} = (V, E)$ where $V = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ and $(\mathbf{b}_i, \mathbf{b}_j) \in E$ if and only if $\langle \mathbf{b}_i, \mathbf{b}_j \rangle \neq 0$. Then, we have an orthogonal decomposition $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$, where each $\mathcal{L}_i$ is spanned by the basis vectors in the i-th connected component of $G_{\mathcal{L}}$, and is irreducible. Moreover, this decomposition is the unique orthogonal decomposition of $\mathcal{L}$ into irreducible lattices.*

Proof. We first prove an intermediate result using the property of a KZ basis.

Claim. For any orthogonal decomposition of the lattice $\mathcal{L} = \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_t$, where the $\mathcal{M}_i$ are not necessarily irreducible, every KZ basis vector $\mathbf{b}_j$ lies in a single summand $\mathcal{M}_i$.

Proof of Claim. We prove this by induction on index j of the KZ basis vectors.

Base case ($j = 1$): For the first KZ basis vector $\mathbf{b}_1$, by property (1) of the KZ basis, $\mathbf{b}_1 = \widetilde{\mathbf{b}}_1$ is a shortest non-zero vector in the entire lattice $\mathcal{L}$. Write $\mathbf{b}_1 = \sum_{i=1}^{t} \mathbf{m}_{i,1}$, where $\mathbf{m}_{i,1} \in \mathcal{M}_i$. Since the decomposition is orthogonal, $\|\mathbf{b}_1\|^2 = \sum_{i=1}^{t} \|\mathbf{m}_{i,1}\|^2$. If more than one component $\mathbf{m}_{i,1}$ is non-zero, then any such nonzero $\mathbf{m}_{i,1}$ satisfies $\|\mathbf{m}_{i,1}\| < \|\mathbf{b}_1\|$. But since $\mathbf{m}_{i,1} \in \mathcal{L}$, this contradicts the fact that $\mathbf{b}_1$ is a shortest non-zero vector in $\mathcal{L}$. Therefore, $\mathbf{b}_1$ must lie entirely within a single component $\mathcal{M}_i$.

Induction step: Assume the claim holds for KZ basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}$, each of these vectors lies within some $\mathcal{M}_i$ (i's are not necessarily the same for different base vectors).

Then, we denote the $\mathbb{R}$-vector space spanned by vectors in $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1}\} \cap \mathcal{M}_i$ as $\mathbf{U}_i$, and its orthogonal complement in the $\mathbb{R}$-vector space spanned by $\mathcal{M}_i$ as $\mathbf{V}_i$. By induction hypothesis, $\mathrm{span}\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1}\} = \mathbf{U}_1 \oplus \cdots \oplus \mathbf{U}_t$, and its orthogonal complement is $\mathrm{span}\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1}\}^{\perp} = \mathbf{V}_1 \oplus \cdots \oplus \mathbf{V}_t$.

Now consider $\mathbf{b}_j$. Recall that $\tilde{\mathbf{b}}_j$ is the component of $\mathbf{b}_j$ orthogonal to the span of $\{\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}\}$, and we write $\mathbf{b}_j = \sum_{i=1}^{t} \mathbf{m}_{i,j}$, where $\mathbf{m}_{i,j} \in \mathcal{M}_i$. We take the orthogonal decomposition of $\mathbf{m}_{i,j}$ with respect to the vector spaces $\mathbf{U}_i$ and $\mathbf{V}_i$: $\mathbf{m}_{i,j} = \mathbf{u}_{i,j} + \mathbf{v}_{i,j}$, where $\mathbf{u}_{i,j} \in \mathbf{U}_i$ and $\mathbf{v}_{i,j} \in \mathbf{V}_i$. As a result, we have $\tilde{\mathbf{b}}_j = \sum_{i=1}^{t} \mathbf{v}_{i,j}$, since $\sum_{i=1}^{t} \mathbf{v}_{i,j} \in \mathbf{V}_1 \oplus \cdots \oplus \mathbf{V}_t$ and is orthogonal to the span of $\{\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}\} = \mathbf{U}_1 \oplus \cdots \oplus \mathbf{U}_t$.

By the KZ basis property (1), $\tilde{\mathbf{b}}_j$ is a shortest non-zero vector in the projected lattice $\pi_j^{(\mathbf{B})}(\mathcal{L})$. Again, by orthogonality of the decomposition, $\|\tilde{\mathbf{b}}_j\|^2 = \sum_{i=1}^{t} \|\mathbf{v}_{i,j}\|^2$. If more than one $\mathbf{v}_{i,j}$ is non-zero, then such non-zero $\mathbf{v}_{i,j}$ has

$\|\mathbf{v}_{i,j}\| < \|\tilde{\mathbf{b}}_j\|$. We choose such vector and consider $\mathbf{m}_{i,j} \in \mathcal{M}_i = \mathbf{u}_{i,j} + \mathbf{v}_{i,j}$. Since $\mathbf{u}_{i,j} \in \mathbf{U}_i$ and $\mathbf{v}_{i,j} \in \mathbf{V}_i$, $\pi_j^{(\mathbf{B})}(\mathbf{m}_{i,j}) = \mathbf{v}_{i,j}$ has smaller length than $\tilde{\mathbf{b}}_j$, contradicting the fact that $\tilde{\mathbf{b}}_j$ is a shortest non-zero vector in this projection.

Therefore, at most one of $\mathbf{v}_{i,j}$ is non-zero. Now, we make use of the KZ basis property (2). We shall use the following uniqueness observation. Fix a projected vector $\widetilde{\mathbf{v}} \in \pi_j^{(\mathbf{B})}(\mathcal{L})$. There is at most one vector $\mathbf{v} = \widetilde{\mathbf{v}} + \sum_{k<j} \mu_k \widetilde{\mathbf{b}}_k \in \mathcal{L}$ whose Gram–Schmidt coefficients satisfy $-1/2 < \mu_k \leq 1/2$ for all $k < j$. Indeed, if $\mathbf{v}$ and $\mathbf{v}'$ are two such vectors, then $\mathbf{v} - \mathbf{v}' \in \mathcal{L} \cap \mathrm{span}_{\mathbb{R}}\{\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}\} = \sum_{k<j} \mathbb{Z}\mathbf{b}_k$. Writing this difference in the Gram–Schmidt basis and using the triangular relation between $(\mathbf{b}_1, \ldots, \mathbf{b}_{j-1})$ and $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_{j-1})$, one sees inductively from $k = j-1$ down to 1 that all coefficient differences are integers. Since each of them lies in $(-1, 1)$, they are all zero. Hence $\mathbf{v} = \mathbf{v}'$.

Now, we consider the vector $\mathbf{m}_{i_0,j}$, which has the same Gram–Schmidt orthogonalization result $\tilde{\mathbf{b}}_j = \mathbf{v}_{i_0,j}$. Since $\mathbf{m}_{i_0,j} \in \mathcal{M}_{i_0}$, it is orthogonal to all basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1}\}$ inside $\mathcal{M}_k$ for $k \neq i_0$. Hence, by adding integer combination of basis vectors inside $\mathcal{M}_{i_0}$, there exists a vector $\mathbf{v}$ such that its Gram–Schmidt coefficients on the basis vectors $\tilde{\mathbf{b}}_k$ lie in $(-\frac{1}{2}, \frac{1}{2}]$, for all $\mathbf{b}_k \in \mathcal{M}_{i_0}$ and all zero on the basis vectors $\tilde{\mathbf{b}}_k$, for $\mathbf{b}_k \in \mathcal{M}_i$ for $i \neq i_0$. Therefore, $\mathbf{v}$ has the same Gram–Schmidt orthogonalization result as $\tilde{\mathbf{b}}_j$ and Gram–Schmidt coefficients all in $(-\frac{1}{2}, \frac{1}{2}]$, and hence by the KZ basis property (2), $\mathbf{v}$ must be equal to $\mathbf{b}_j$, and hence $\mathbf{b}_j = \mathbf{v}$ lies within a single component $\mathcal{M}_{i_0}$. □

Let $\mathcal{L}_i$ be the lattice spanned by the basis vectors in the i-th connected component of the graph $G_{\mathcal{L}}$ for $1 \leq i \leq s$. By linearity, $\mathcal{L}_i \perp \mathcal{L}_j$ for $i \neq j$, since basis vectors from different connected components must be orthogonal. Thus, $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$ is an orthogonal decomposition.

Next, we show that each $\mathcal{L}_i$ is irreducible. Suppose, for contradiction, that some $\mathcal{L}_i$ is reducible. Then there exists a non-trivial orthogonal decomposition $\mathcal{L}_i = \mathcal{M}_1 \oplus \mathcal{M}_2$. Consider the direct sum decomposition of $\mathcal{L}$ as $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_{i-1} \oplus \mathcal{M}_1 \oplus \mathcal{M}_2 \oplus \mathcal{L}_{i+1} \oplus \cdots \oplus \mathcal{L}_s$. The claim states that every basis vector $\mathbf{b}_j$ of $\mathcal{L}_i$ must lie in either $\mathcal{M}_1$ or $\mathcal{M}_2$. Moreover, since the decomposition is nontrivial, the basis vectors cannot all lie entirely within $\mathcal{M}_1$ or entirely within $\mathcal{M}_2$. Then, since $\mathcal{M}_1 \oplus \mathcal{M}_2$ is an orthogonal decomposition, the induced graph on the basis vectors in $\mathcal{L}_i$ must be disconnected, a contradiction to the fact that the basis vectors spanning $\mathcal{L}_i$ form a connected component of $G_{\mathcal{L}}$. Thus, each $\mathcal{L}_i$ is irreducible, and $\mathcal{L} = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_s$ gives the unique orthogonal decomposition into irreducible components by Theorem 2.2. □

It is a standard fact that KZ basis can be found using SVP oracle at most n times. Using the fastest known algorithm for SVP as stated in Theorem 2.1, we have the following.

Corollary 3.4. *There is a randomized $2^{n+o(n)}$-time and space algorithm for* LODP.

4 The reductions among LAGP, SLIP, #LAP, and DLIP

In this section, we give some reductions related to LIP that correspond to existing reductions in GIP.

In Proposition 3.1, we showed that there is a randomized Turing reduction from SLIP to SaLAP, but in fact the reverse is also true, and the reduction preserves dimension. The reduction essentially uses the random self-reduction property of LIP, as described in Lemma 2.4.

Lemma 4.1 (Corollary 4.2 in [42])**.** *There is a randomized Turing reduction from* SaLAP *to* SLIP *in the same dimension.*

In combination with Proposition 3.1, this gives the following corollary.

Corollary 4.1. *In the context of randomized Turing reduction,* SLIP *is equivalent to* SaLAP*.*

In graph isomorphism, we know that the graph isomorphism problem is equivalent to the graph automorphism problem, that is, to find a set of generators of the automorphism group of a graph.

We give the same reduction for lattice isomorphisms, in which we use Corollary 4.1 and the fact that a uniform distribution on a finite group can be used to obtain a set of generators by Theorem 2.5, and conversely, a set of generators can be used to approximate a uniform distribution on a finite group by Theorem 2.4.

Theorem 4.1. *In the context of randomized Turing reduction,* LAGP *is equivalent to* SLIP*.*

Proof. The randomized reduction from LAGP to SLIP is straightforward, given an LAGP instance $\mathbf{G}$ and an SLIP oracle $\mathcal{O}$, we can use $\mathcal{O}$ to get an SaLAP oracle $\mathcal{P}$ by Lemma 4.1, then we call $\mathcal{P}$ on the $n \times n$ Gram matrix $\mathbf{G}$ k times to get a set of elements in $\mathrm{Stab}(\mathbf{G})$: $\mathbf{U}_1, \ldots, \mathbf{U}_k \xleftarrow{\$}_{i.i.d.} \mathrm{Stab}(\mathbf{G})$, where $k = n^2$. Since $|\mathrm{Stab}(\mathbf{G})| < 2^{n^2}$ for $n > 10$ due to [29], thus by Theorem 2.5, we have $\langle \mathbf{U}_1, \ldots, \mathbf{U}_k \rangle = \mathrm{Stab}(\mathbf{G})$ with probability larger than $1 - 2^{-n}$.

For the randomized reduction from SLIP to LAGP, it is essentially a combination of Theorem 2.4 and error versions of Proposition 3.1. By [29], for an $n \times n$ Gram matrix $\mathbf{G}$, we have the upper bound $n^2 > \log_2(|\mathrm{Stab}(\mathbf{G})|)$, then we can use LAGP oracle $\mathcal{O}$ and the BPP algorithm in Theorem 2.4 to obtain an error version SaLAP oracle $\mathcal{P}$, which takes the Gram matrix $\mathbf{G}$ as input and outputs each transition matrix $\mathbf{U} \in \mathrm{Stab}(\mathbf{G})$ with probability $\frac{1}{|\mathrm{Stab}(\mathbf{G})|} \cdot (1 \pm 2^{-n^2})$. This error version SaLAP oracle is sufficient to run the reduction in Proposition 3.1. Specifically, for an SLIP instance $(\mathcal{L} \cong \mathcal{L}')$ (written in the lattice form as in Proposition 3.1), let $\mathcal{N} = \mathcal{L} \oplus \mathcal{L}'$, then for $\phi \xleftarrow{\$} \mathcal{P}(\mathcal{N})$, we have $\Pr_{\phi \xleftarrow{\$} \mathcal{P}(\mathcal{N})}[\phi(\mathcal{L}_1) \in \mathcal{L}'] \geq \frac{1}{2}(1 - 2^{-n^2}) > \frac{1}{3}$ for $n > 2$, where $\mathcal{L}_1$ is the first irreducible component of $\mathcal{L}$. This property is adequate for completing the reduction. Both the reduction process and the proof follow the same steps as outlined in Proposition 3.1. □

Next, we prove that computing the size of a lattice automorphism group can be reduced to the search lattice isomorphism problem, which corresponds to the reduction of computing the size of a graph automorphism group to the graph isomorphism problem.

Corollary 4.2. *There is a randomized Turing reduction from* #LAP *to* SLIP.

Proof. Given a #LAP instance $\mathbf{G}$, we can call the SLIP oracle to compute a set of generators $\{g_1, \ldots, g_k\}$ of $\mathrm{Stab}(\mathbf{G})$ by Theorem 4.1. Then, we use the BPP algorithm in Theorem 2.3 and $\{g_1, \ldots, g_k\}$ to compute $|\mathrm{Stab}(\mathbf{G})|$ and output it. □

Corollary 4.3. *If* #LAP *is* #P*-complete, then* NP $\subseteq$ coAM.

Proof. Since #SAT is in #P and SAT $\leq$ #SAT, if #LAP is #P-complete, then there is a randomized Turing reduction from SAT to SLIP by Corollary 4.2, thus we have NP $\subseteq$ coAM by Theorem 3.1. □

Next, we give a reduction from DLIP to #LAP, which exploits the automorphism group of the direct sum of irreducible lattices.

Theorem 4.2. *There is a Turing reduction from* DLIP *to* #LAP.

Proof. Given a DLIP instance $(\mathbf{G}_0, \mathbf{G}_1)$. The reduction algorithm is as follows: first, we use the #LAP oracle $\mathcal{O}$ to calculate $n_{01} = |\mathrm{Stab}(\mathbf{G}_0 \oplus \mathbf{G}_1)|$, $n_{00} = |\mathrm{Stab}(\mathbf{G}_0 \oplus \mathbf{G}_0)|$ and $n_{11} = |\mathrm{Stab}(\mathbf{G}_1 \oplus \mathbf{G}_1)|$. Then we determine whether n_{01} is equal to n_{00} and n_{11}. If both equalities hold, output 1, i.e., declare that $\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1) \neq \emptyset$; otherwise output 0.

Next we prove the correctness. If $\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1) \neq \emptyset$, then $\mathbf{G}_0 \oplus \mathbf{G}_1$, $\mathbf{G}_0 \oplus \mathbf{G}_0$, and $\mathbf{G}_1 \oplus \mathbf{G}_1$ are isomorphic Gram matrices, and hence $n_{00} = n_{01} = n_{11}$.

If $\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1) = \emptyset$, group the irreducible components of $\mathbf{G}_0$ and $\mathbf{G}_1$ by isomorphism type. After reindexing the types, write

$$\mathbf{G}_0 \cong \bigoplus_{i=1}^{a_1} \mathbf{N}_1 \oplus \bigoplus_{i=1}^{a_2} \mathbf{N}_2 \oplus \cdots \oplus \bigoplus_{i=1}^{a_s} \mathbf{N}_s,$$
$$\mathbf{G}_1 \cong \bigoplus_{i=1}^{b_1} \mathbf{M}_1 \oplus \bigoplus_{i=1}^{b_2} \mathbf{M}_2 \oplus \cdots \oplus \bigoplus_{i=1}^{b_t} \mathbf{M}_t,$$

where $s, t \geq 1, a_i, b_i > 0$, and $\mathbf{N}_i, \mathbf{M}_i$ are irreducible Gram matrices and the $\mathbf{N}_i$ are pairwise non-isomorphic, the $\mathbf{M}_j$ are pairwise non-isomorphic, and all multiplicities are positive. After reindexing, assume that the common isomorphism types are represented by

$$\mathbf{N}_1 \cong \mathbf{M}_1, \ldots, \mathbf{N}_\ell \cong \mathbf{M}_\ell,$$

and that no other $\mathbf{N}_i$ is isomorphic to any $\mathbf{M}_j$.

Due to Lemma 3.1, Corollary 3.1, we have $|\mathrm{Stab}(\mathbf{G}_0)| = \prod_{i=1}^{s}(a_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{a_i}$ and $|\mathrm{Stab}(\mathbf{G}_1)| = \prod_{i=1}^{t}(b_i)!|\mathrm{Stab}(\mathbf{M}_i)|^{b_i}$. Similarly, $n_{00} = |\mathrm{Stab}(\mathbf{G}_0 \oplus \mathbf{G}_0)|$, $n_{11} = |\mathrm{Stab}(\mathbf{G}_1 \oplus \mathbf{G}_1)|$, $n_{01} = |\mathrm{Stab}(\mathbf{G}_0 \oplus \mathbf{G}_1)|$, then we have:

$$n_{00} = \prod_{i=1}^{s}(2a_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{2a_i}, \quad n_{11} = \prod_{i=1}^{t}(2b_i)!|\mathrm{Stab}(\mathbf{M}_i)|^{2b_i},$$
$$n_{01} = \prod_{i=1}^{\ell}(a_i+b_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{a_i+b_i} \cdot \prod_{i=\ell+1}^{s}(a_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{a_i} \cdot \prod_{i=\ell+1}^{t}(b_i)!|\mathrm{Stab}(\mathbf{M}_i)|^{b_i}.$$

We show $(n_{01})^2 < n_{00}n_{11}$. The stabilizer-size factors cancel ($|\mathrm{Stab}(\mathbf{N}_i)| = |\mathrm{Stab}(\mathbf{M}_i)|$ for $1 \le i \le \ell$), so it suffices to compare the factorial terms. For non-common types, $(a_i!)^2 < (2a_i)!$ and $(b_i!)^2 < (2b_i)!$. For common types, the log-convexity of factorials gives

$$((a_i+b_i)!)^2 \le (2a_i)!(2b_i)!,$$

with equality if and only if $a_i = b_i$. Therefore equality in $(n_{01})^2 \le n_{00}n_{11}$ can hold only if $\ell = s = t$ and $a_i = b_i$ for every i, which would imply $\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1) \neq \emptyset$, a contradiction. Hence $(n_{01})^2 < n_{00}n_{11}$, so in particular $n_{00} = n_{01} = n_{11}$ cannot hold. □

Corollary 4.4. *In the context of Turing reduction,* #LIP *is equivalent to* #LAP.

Proof. For #LIP $\le$ #LAP, given a #LIP instance $(\mathbf{G}_0, \mathbf{G}_1)$, first use Theorem 4.2 to decide whether $\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1)$ is empty. If it is empty, output 0. Otherwise, call the #LAP oracle on $\mathbf{G}_0$ and output $|\mathrm{Stab}(\mathbf{G}_0)|$, which equals $|\mathrm{Iso}(\mathbf{G}_0, \mathbf{G}_1)|$.

For #LAP $\le$ #LIP, given a #LAP instance $\mathbf{G}$, call the #LIP oracle on $(\mathbf{G}, \mathbf{G})$ and output $|\mathrm{Iso}(\mathbf{G}, \mathbf{G})| = |\mathrm{Stab}(\mathbf{G})|$. □

In addition, counting the size of the automorphism group of a lattice can be used to distinguish whether a lattice is irreducible.

Theorem 4.3. *There is a Turing reduction from* DILP *to* #LAP.

Proof. Given a DILP instance $\mathbf{G}$, query the #LAP oracle on $\mathbf{G}$ and $\mathbf{G} \oplus \mathbf{G}$ to obtain

$$n_1 = |\mathrm{Stab}(\mathbf{G})|, \qquad n_{11} = |\mathrm{Stab}(\mathbf{G} \oplus \mathbf{G})|.$$

Accept if and only if $n_{11}/n_1^2 = 2$.

For correctness, if $\mathbf{G}$ is irreducible, then $n_{11} = 2|\mathrm{Stab}(\mathbf{G})|^2 = 2n_1^2$ by Lemma 3.1 and Corollary 3.1, so the ratio equals 2. If $\mathbf{G}$ is reducible, write

$$\mathbf{G} = \bigoplus_{i=1}^{a_1}\mathbf{N}_1 \oplus \bigoplus_{i=1}^{a_2}\mathbf{N}_2 \oplus \cdots \oplus \bigoplus_{i=1}^{a_s}\mathbf{N}_s,$$

where the $\mathbf{N}_i$ are pairwise non-isomorphic irreducible Gram matrices and all $a_i > 0$. Then

$$|\mathrm{Stab}(\mathbf{G})| = \prod_{i=1}^{s}(a_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{a_i}, \qquad |\mathrm{Stab}(\mathbf{G} \oplus \mathbf{G})| = \prod_{i=1}^{s}(2a_i)!|\mathrm{Stab}(\mathbf{N}_i)|^{2a_i}.$$

Thus

$$\frac{n_{11}}{n_1^2} = \frac{|\mathrm{Stab}(\mathbf{G} \oplus \mathbf{G})|}{|\mathrm{Stab}(\mathbf{G})|^2} = \prod_{i=1}^{s}\frac{(2a_i)!}{(a_i)!(a_i)!} = \prod_{i=1}^{s}\binom{2a_i}{a_i} \geq 2,$$

with equality if and only if $s = 1$ and $a_1 = 1$, which is precisely the irreducible case. Hence every reducible $\mathbf{G}$ gives a ratio strictly larger than 2. □

5 Conclusion

In this paper, we demonstrate that SLIP cannot be NP-hard unless PH collapses, by leveraging the randomized self-reduction algorithm of LIP together with the properties of irreducible lattices. Furthermore, we reduce the task of computing the orthogonal decomposition of a lattice to that of calculating the KZ basis, thereby revealing a novel structural connection between KZ bases and irreducible decompositions. Additionally, we establish reductions among LAGP, SLIP, #LAP, #LIP, and DLIP, thereby deepening our understanding of the complexity landscape surrounding LIP.

As a direction for future research, it remains an intriguing open problem whether a reduction exists from SLIP to #LAP and from #LAP to DLIP. Moreover, investigating whether such reductions could be exploited to design a more efficient algorithm for SLIP represents another important avenue for exploration.

Acknowledgement

The authors would like to thank anonymous reviewers for their helpful comments and suggestions. K.J. is supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM114) and thanks ANT Group. Y.L. is supported by Beijing Natural Science Foundation (QY26408).

References

1. Ackermann, L., Roux-Langlois, A., Wallet, A.: Public-key encryption from the lattice isomorphism problem. In: WCC. pp. 1–11 (2024)
2. Aggarwal, D., Bennett, H., Golovnev, A., Stephens-Davidowitz, N.: Fine-grained hardness of CVP(P) - everything that we can prove (and nothing else). In: SODA. pp. 1816–1835 (2021)
3. Aggarwal, D., Dadush, D., Regev, O., Stephens-Davidowitz, N.: Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In: STOC. pp. 733–742 (2015)

4. Aggarwal, D., Stephens-Davidowitz, N.: Just take the average! an embarrassingly simple 2^n-time algorithm for SVP (and CVP). In: SOSA. pp. 12:1–12:19 (2018)
5. Aiello, W., Hastad, J.: Statistical zero-knowledge languages can be recognized in two rounds. J. Comput. Syst. Sci. **42**(3), 327–345 (1991)
6. Ajtai, M.: The shortest vector problem in l2 is np-hard for randomized reductions. In: STOC. pp. 10–19 (1998)
7. Akavia, A., Goldreich, O., Goldwasser, S., Moshkovitz, D.: On basing one-way functions on np-hardness. In: STOC. pp. 701–710 (2006)
8. Allombert, B., Pellet-Mary, A., van Woerden, W.: Cryptanalysis of rank-2 module-LIP: a single real embedding is all it takes. In: EUROCRYPT. pp. 184–212 (2025)
9. Babai, L.: Local expansion of vertex-transitive graphs and random generation in finite groups. In: STOC. pp. 164–174 (1991)
10. Babai, L.: Graph isomorphism in quasipolynomial time [extended abstract]. In: STOC. pp. 684–697 (2016)
11. Babai, L., Moran, S.: Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. J. Comput. Syst. Sci. **36**(2), 254–276 (1988)
12. Beals, R., Babai, L.: Las vegas algorithms for matrix groups. In: FOCS. pp. 427–436 (1993)
13. Bennett, H.: The complexity of the shortest vector problem. Electron. Colloquium Comput. Complex. **TR22-170** (2022)
14. Bennett, H., Dadush, D., Stephens-Davidowitz, N.: On the lattice distortion problem. In: ESA. pp. 9:1–9:17 (2016)
15. Bennett, H., Ganju, A., Peetathawatchai, P., Stephens-Davidowitz, N.: Just how hard are rotations of Z^n? algorithms and cryptography with the simplest lattice. IACR ePrint 2021/1548 (2021)
16. Bennett, H., Golovnev, A., Stephens-Davidowitz, N.: On the quantitative hardness of CVP. In: FOCS. pp. 13–24 (2017)
17. Biasse, J.F., Micheli, G.: A search-to-decision reduction for the permutation code equivalence problem. In: ISIT. pp. 602–607 (2023)
18. Blanks, T.L., Miller, S.D.: Generating cryptographically-strong random lattice bases and recognizing rotations of Z^n. In: PQCrypto. pp. 319–338 (2021)
19. Bogdanov, A., Brzuska, C.: On basing size-verifiable one-way functions on np-hardness. In: TCC. pp. 1–6 (2015)
20. Boppana, R.B., Hastad, J., Zachos, S.: Does co-np have short interactive proofs? Inf. Process. Lett. **25**(2), 127–132 (1987)
21. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: STOC. pp. 575–584 (2013)
22. Branco, P., Malavolta, G., Maradni, Z.: Fully-homomorphic encryption from lattice isomorphism. In: TCC. pp. 220–252 (2025)
23. Bratus, S., Pak, I.: On sampling generating sets of finite groups. Manuscript (1999)
24. Ducas, L.: Provable lattice reduction of Z^n with blocksize $n/2$. Des. Codes Cryptogr. **92**(4), 909–916 (2024)
25. Ducas, L., Postlethwaite, E.W., Pulles, L.N., van Woerden, W.P.J.: Hawk: Module LIP makes lattice signatures fast, compact and simple. In: ASIACRYPT. pp. 65–94 (2022)
26. Ducas, L., van Woerden, W.P.J.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: EUROCRYPT. pp. 643–673 (2022)
27. Dutour Sikirić, M., Haensch, A., Voight, J., van Woerden, W.P.: A canonical form for positive definite matrices. Open Book Series **4**(1), 179–195 (2020)

28. Eichler, M.: Note zur theorie der kristallgitter. Math. Ann. **125**(1), 51–55 (1952)
29. Feit, W.: Orders of finite linear groups. Preprint (1995)
30. Fortnow, L.: The complexity of perfect zero-knowledge. In: STOC. pp. 204–209 (1987)
31. Geißler, K., Smart, N.P.: Computing the $M = U\,U^t$ integer matrix decomposition. In: IMA Cryptography and Coding. pp. 223–233 (2003)
32. van Gent, D., van Woerden, W.: A search to distinguish reduction for the isomorphism problem on direct sum lattices. IACR ePrint 2025/1204 (2025)
33. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. pp. 197–206 (2008)
34. Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: EUROCRYPT. pp. 299–320 (2002)
35. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. J. ACM **38**(3), 690–728 (1991)
36. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. In: STOC. pp. 59–68 (1986)
37. Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: STOC. pp. 469–477 (2007)
38. Haviv, I., Regev, O.: On the lattice isomorphism problem. In: SODA. pp. 391–404 (2014)
39. Hemkemeier, B., Vallentin, F.: On the decomposition of lattices. In: ECCC (1998)
40. Hunkenschröder, C.: Deciding whether a lattice has an orthonormal basis is in co-NP. Math. Program. **208**(1), 763–775 (2024)
41. Jiang, K., Wang, A., Luo, H., Liu, G., Tang, G., Pan, Y., Wang, X.: Re-randomize and extract: A novel commitment construction framework based on group actions. In: EUROCRYPT. pp. 124–153 (2025)
42. Jiang, K., Wang, A., Luo, H., Liu, G., Yu, Y., Wang, X.: Exploiting the symmetry of Z^n: Randomization and the automorphism problem. In: ASIACRYPT. pp. 167–200 (2023)
43. Jr., H.W.L., Silverberg, A.: Revisiting the gentry-szydlo algorithm. In: CRYPTO. pp. 280–296 (2014)
44. Jr., H.W.L., Silverberg, A.: Lattices with symmetry. J. Cryptol. **30**(3), 760–804 (2017)
45. Kneser, M.: Zur theorie der kristallgitter. Math. Ann. **127**(1), 105–106 (1954)
46. Lagarias, J.C., Lenstra Jr, H.W., Schnorr, C.P.: Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. Combinatorica **10**(4), 333–348 (1990)
47. Lenstra Jr, H.W., Silverberg, A.: Testing isomorphism of lattices over cm-orders. SIAM J. Comput. **48**(4), 1300–1334 (2019)
48. Leporati, A., Rovida, L., van Woerden, W.: Beyond LWE: a lattice framework for homomorphic encryption. IACR ePrint 2025/1171 (2025)
49. Liu, T.: On basing search sivp on np-hardness. In: TCC. pp. 98–119 (2018)
50. Luo, H., Jiang, K., Pan, Y., Wang, A.: Cryptanalysis of rank-2 module-LIP with symplectic automorphisms. In: ASIACRYPT. pp. 359–385 (2024)
51. Mathon, R.: A note on the graph isomorphism counting problem. Inf. Process. Lett. **8**(3), 131–136 (1979)
52. Micciancio, D., Goldwasser, S.: Complexity of lattice problems - a cryptographic perspective. Springer (2002)

53. Mureau, G., Pellet-Mary, A., Pliatsok, G., Wallet, A.: Cryptanalysis of rank-2 module-LIP in totally real number fields. In: EUROCRYPT. pp. 226–255 (2024)
54. O'Meara, O.: On indecomposable quadratic forms. J. Reine Angew. Math. **317**, 120–156 (1980)
55. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Trans. Inf. Theory **43**(5), 1602–1604 (2002)
56. Plesken, W., Pohst, M.: Constructing integral lattices with prescribed minimum. i. Math. Comput. **45**(171), 209–221 (1985)
57. Plesken, W., Souvignier, B.: Computing isometries of lattices. J. Symb. Comput. **24**(3-4), 327–334 (1997)
58. Sikiric, M.D., Schürmann, A., Vallentin, F.: Complexity and algorithms for computing voronoi cells of lattices. Math. Comput. **78**(267), 1713–1731 (2009)
59. Szydlo, M.: Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In: EUROCRYPT. pp. 433–448 (2003)
60. Valiant, L.G., Vazirani, V.V.: Np is as easy as detecting unique solutions. In: STOC. pp. 458–463 (1985)
61. van Woerden, W.P.J.: Dense and smooth lattices in any genus. In: ASIACRYPT. pp. 386–417 (2024)