IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 20, 2020, accepted October 12, 2020, date of publication October 15, 2020, date of current version October 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3031276

A Review of Sieve Algorithms in Solving the Shortest Lattice Vector Problem

ZEDONG SUN, CHUNXIANG GU, AND YONGHUI ZHENG
State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China
Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China

Corresponding author: Zedong Sun (iamsunzedong@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772584, and in part by the Foundation of Science and Technology on Information Insurance Laboratory under Grant KJ-17-001.

ABSTRACT As a category of algorithms to solve the shortest lattice vector problem, sieve algorithms have drawn more and more attention due to the prominent performance in recent years. Enumeration algorithms used to perform better in practice even though sieve algorithms are asymptotically faster. Combined with techniques like locality-sensitive hashing and rank reduction, sieve algorithms now are capable of competing with enumeration algorithms. In this work, we study sieve algorithms in solving the shortest vector problem on lattices by categorizing various sieve algorithms and elaborating on ideas and techniques used to improve sieve algorithms. In addition, we present several prospective directions worth future research.

INDEX TERMS Lattice theory, post-quantum cryptography, lattice-based cryptography, shortest vector problem, sieve algorithm.

I. INTRODUCTION

Originated in the 19th century, lattice theory has been thoroughly studied by famous mathematicians like Gauss, Minkowski and Hermite *et al.* At first, lattice theory was introduced to solve geometric problems such as sphere packing and sphere covering. For the first time the application of lattice theory in cryptography was as a cryptanalysis tool for that the security of many non-lattice-based cryptosystems can be reduced to the hardness of hard lattice problems. In 1980s, knapsack problem was reduced to the shortest lattice vector problem and LLL algorithm [53] was used to solve it [21], [50], [51]. Afterwards, Bleichenbacher and May [15] and Coppersmith [19], [20] utilized lattice algorithms to crack vulnerable RSA [69] cryptosystems. Moreover, lattice reduction algorithms were used to solve hidden number problem, which leads to attacks on DSA cryptosystems of special settings [37], [64].

Besides cryptanalysis, lattice theory plays a more important role in the design of cryptographic schemes. In 1996, Ajtai [1] proposed a method of constructing a random class of lattices, on which finding a short vector is as hard as several worst-case hard lattice problems. Furthermore, Ajtai and Dwork [3] presented the first worst-case to average-case reduction for hard lattice problems, which furnishes lattice-based cryptosystems with provable security. Based on these work, various cryptographic schemes including collision-resistant hash function [55], public-key encryption cryptosystem [3], [31], [32] and digital signature scheme [16], [24], [54] were proposed.

In addition, the birth and development of quantum computing have brought challenges to the security of conventional public key cryptosystems such as RSA [69], ElGamal [29] and ECC [42], [63] since that solving the hard mathematical problems they rely on become feasible with quantum algorithms [74], [75] and large-scale quantum computers. Under this circumstance, the issue of information security in the upcoming quantum era is attracting more and more attention and cryptographers seek to design ''post-quantum'' cryptosystems resistant to quantum attacks. Lattice-based cryptography, code-based cryptography, multivariate polynomial cryptography and hash-based signatures are the four main branches of post-quantum cryptography. Among these branches, lattice-based cryptosystems have been emerged as prime candidates for post-quantum cryptosystem for that they are relatively simple and own high efficiency and parallelizability. In July 2020, National Institute of Standards and Technology (NIST) announced the round 3 finalists of its

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

post-quantum cryptography standardization while 5 out of the 7 cryptographic schemes are based on hard lattice problems.

Either cryptanalysis using lattice algorithms or the design of lattice-based cryptosystem, hard lattice problems are of vital importance. Among all the hard lattice problems, the shortest vector problem (SVP), which aims to find a non-zero vector on the given lattice, is the most representative one. In 1998, Ajtai [2] proved that the shortest vector problem in L_2 norm is *NP*-hard for randomized reductions.

Till now, the shortest vector problem has been well studied and a quantity of algorithms to solve the shortest vector problem have been proposed. There are mainly two categories of algorithms to solve the shortest vector problem: enumeration algorithms and sieve algorithms.

Enumeration algorithms have been under constant study from the 1980s. In 1981, Pohst [66] proposed the first enumeration algorithm, which cost $2^{O(n^2)}$ time and poly(n) memory on a lattice of dimension n. Two years later, Kannan [40] proved that with appropriate preprocessing, the time complexity of enumeration algorithm can be reduced to $2^{O(n\log n)}$. In practice, for the sake of improving algorithm efficiency, enumerations are combined with pruning technique [28], [72], [73], [77].

Sieve algorithms started with Ajtai, Kumar and Sivakumar's work [4] in 2001, which is the first algorithm solving the shortest vector problem in single-exponential time. Years later, heuristic assumptions were used in the design of sieve algorithms, which makes NVSieve algorithm [65], GaussSieve algorithm [79] and their variants much more practical. Nevertheless, the exponential memory consumption still restircts the application of sieve algorithms. On this occasion, TupleSieve algorithms [9], [34], [35] were proposed to overcome this obstacle by means of a trade-off between time and memory. In addition, locality-sensitive hashing (LSH), a technique to solve the nearest neighbor search problem, was introduced into sieve algorithms [11], [13], [14], [44], [45], [47] to improve efficiency. In 2018, Ducas [23] and Laarhoven and Mariano [48] proposed two different rank reduction techniques respectively and applied them to sieve algorithms, which improve the efficiency greatly in practice. Inspired by SubSieve technique and progressive sieving, in 2019 Albrecht *et al.* [5] proposed the General Sieve Kernel, which is a stateful machine based on sieve algorithms and solved previously unsolved SVP Challenge instances. Moreover, quantum techniques [41], [49] and evolutionary techniques [46] have been used to improve the efficiency of sieve algorithms in recent years.

A. MOTIVATION

On one hand, the security of some non-lattice-based cryptosystems can be reduced to hard lattice problems. On the other hand, owing to the merit in security, efficiency and fuction, lattice-based cryptosystems have emerged as the most promising cryptographic construction in post-quantum era. As the most basic hard lattice problem, the shortest vector problem deserves in-depth study for that it offers insights for cryptanalysis and security parameter selection of certain cryptographic schemes. Enumeration algorithm used to be the most effective method to solve SVP in practice even though sieve algorithm has better asymptotic time complexity. Nevertheless, improvements of sieve algorithm in recent years enable it to compete with enumeration algorithm as well. Therefore, making such a review of sieve algorithms in solving SVP could provide a better comprehension.

B. CONTRIBUTION

In this work, we provide an overview of sieve algorithms in solving the shortest vector problem. For this purpose, we collect, categorize and elaborate on the advances of sieve algorithms and analyze the strengths and weaknesses of each algorithm as well. In the end, we discuss directions worth further research.

C. OUTLINE

The remainder of this article is organized as follows. In Section II, we introduce preliminaries on lattice and sieve algorithms. Section III provides an overview of sieve algorithms in solving SVP. Section IV concludes with a brief discussion about further research directions.

II. LATTICE AND SIEVE ALGORITHM

In this section, we introduce some theoretical backgroud about lattice-based cryptography. Though there are a series of hard lattice problems, in this work we mainly focus on the shortest vector problem. All vectors in this article are denoted by bold lower case letters and are to be read as column vectors while all matrices are denoted by bold capital letters.

A. LATTICE

Definition 1 (Lattice): Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. A lattice in $\mathbb{R}^m$ is the set

$$\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$$

of all integral combinations of n linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{R}^m (m \geq n)$.

The integers n and m are called the *rank* and *dimension* of the lattice respectively and the lattice is full-rank when $n = m$. Vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n$ are called a *lattice basis* and it is represented as a matrix $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_n] \in \mathbb{R}^{m \times n}$. Figure 1 illustrates an example lattice in $\mathbb{R}^2$ and the lattice basis consists of vector $\mathbf{b}_1$ and $\mathbf{b}_2$. Actually, the basis of a lattice is not unique, as the example lattice in Figure 1, $\mathbf{b}_1$ and $\mathbf{b}_1 - \mathbf{b}_2$ also make up a lattice basis.

If $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ is a full-rank matrix,the lattice $\mathcal{L}$ generated by the basis $\mathbf{B}$ is denoted by $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^{\mathbf{n}}\}$.

Definition 2 (Gram-Schmidt Orthogonalization): Let $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ be n linearly independent vectors, $\mathbf{B}^* = (\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*)$ is the Gram-Schmidt orthogonalization of $\mathbf{B}$, where $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$, $\mu_{i,j} = \frac{<\mathbf{b}_i, \mathbf{b}_j^*>}{<\mathbf{b}_j^*, \mathbf{b}_j^*>}$.

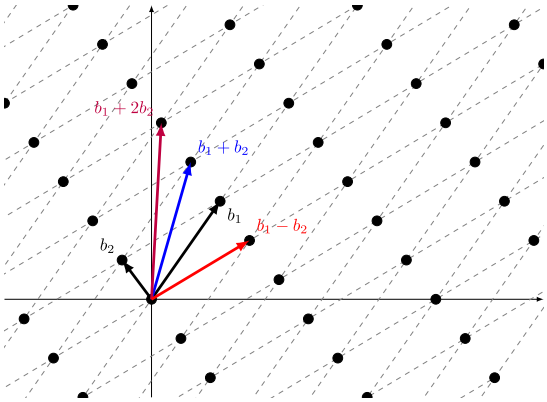


FIGURE 1. An example lattice in $\mathbb{R}^2$.

The volume of a lattice $\mathrm{Vol}(\mathcal{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\|$ where $\|\mathbf{b}_i^*\|$ is the Eculidean norm of vector $\mathbf{b}_i^*$.

Definition 3 (Successive Minima): Let $\mathcal{L}$ be a lattice of rank n, the i-th successive minimum is defined as

$$\lambda_i(\mathcal{L}) = \inf\{r|\dim(\mathrm{span}(\mathcal{L} \cap \bar{\mathcal{B}}(r)))\}, i = 1, \cdots, n,$$

where $\bar{\mathcal{B}}(r)$ is a closed ball of radius r with the origin as its center.

Definition 4 (Shortest Vector Problem): The exact shortest vector problem can be defined as: given a basis $\mathbf{B}$ of the lattice $\mathcal{L}$, find a non-zero vector $\mathbf{s} \in \mathcal{L}$ such that

$$\|\mathbf{s}\| = \min\{\|\mathbf{v}\| : \mathbf{v} \in \mathcal{L}(\mathbf{B}), \|\mathbf{v}\| \neq 0\}.$$

The norm of such a shortest non-zero vector of a lattic $\mathcal{L}$ is usually denoted by $\lambda_1(\mathcal{L})$.

Besides the exact SVP, there are several other versions.

Approximate Shortest Vector Problem: given a basis $\mathbf{B}$ of the lattice $\mathcal{L}$ and a real number $\gamma > 1$, find a non-zero vector $\mathbf{s} \in \mathcal{L}$ such that

$$\|\mathbf{s}\| \leq \gamma \|\mathbf{v}\|, \forall \mathbf{v} \in \mathcal{L}(\mathbf{B}), \|\mathbf{v}\| \neq 0.$$

Unique Shortest Vector Problem: given a a lattice $\mathcal{L}(\mathbf{B})$ satisfying $\lambda_2(\mathcal{L}) > \gamma\lambda_1(\mathcal{L})$ for $\gamma > 1$ and $\gamma \in \mathbb{R}$, find a non-zero vector $\mathbf{s} \in \mathcal{L}(\mathbf{B})$ such that

$$\|\mathbf{s}\| = \min\{\|\mathbf{v}\| : \mathbf{v} \in \mathcal{L}(\mathbf{B}), \|\mathbf{v}\| \neq 0\}.$$

Shortest Independent Vector Problem: given a lattice $\mathcal{L}(\mathbf{B})$ of rank n, find n linearly independent vectors $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$ such that

$$\|\mathbf{s}_i\| = \lambda_i(\mathcal{L}(\mathbf{B})), \quad i = 1, \cdots, n.$$

Definition 5 (Gaussian Heuristic): For a lattice $\mathcal{L}$, Gaussian Heurisitic $\mathrm{GH}(\mathcal{L})$ gives the expected first minimum. For a full rank lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathrm{GH}(\mathcal{L})$ is defined as:

$$\mathrm{GH}(\mathcal{L}) = \sqrt{n/2\pi e} \cdot \mathrm{Vol}(\mathcal{L})^{1/n}.$$

When solving the shortest vector problem, $\mathrm{GH}(\mathcal{L})$ is usually regraded as the expected norm of the shortest vector. For example, the target norm is $1.05\mathrm{GH}(\mathcal{L})$ in the SVP Challenge.[1]

B. SIEVE ALGORITHM

Since Ajtai *et al.* [4] introduced sieve algorithm in 2001, there have been a quantity of variants and they share the same framework as shown in Algorithm 1.

Algorithm 1 Sieve Algorithm ($\mathbf{B}$)

Require:
Lattice basis $\mathbf{B}$
Ensure:
A list of short lattice vectors
1: $L \leftarrow$ a set of N short vectors sampled from lattice $\mathcal{L}(\mathbf{B})$
2: **while** $\exists(\mathbf{v}, \mathbf{w}) \in L^2$ such that $\|\mathbf{v} - \mathbf{w}\| < \|\mathbf{v}\|$ **do**
3: $\quad \mathbf{v} \leftarrow \mathbf{v} - \mathbf{w}$
4: **end while**
5: **return** L

Typically, to imporve efficiency, before running sieve algorithm on a lattice, lattice basis needs to be preprocessed by lattice basis reduction algorithm (LLL [53] or BKZ [18], [72]). And as Algorithm 1 shows, a sieving procedure mainly consists of two parts: sampling short vectors and vector reduction. Through calling sieve algorithm iteratively, vectors in list L get shorter and shorter and as a result the shortest vector problem is solved.

Definition 6 (Gauss-reduced): For vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{L}(\mathbf{B})$, if $\max(\|\mathbf{v}_1\|, \|\mathbf{v}_2\|) \leq \|\mathbf{v}_1 \pm \mathbf{v}_2\|$, then $\mathbf{v}_1, \mathbf{v}_2$ are called Gauss-reduced.

Definition 7 (Pairwise-reduced): Let list L be a set of N vectors from lattice $\mathcal{L}(\mathbf{B})$, if for any two different vectors $\mathbf{v}_1, \mathbf{v}_2$ $(i, j = 1, \cdots, N$ and $i \neq j) \in L$, $\mathbf{v}_1, \mathbf{v}_2$ are Gauss-reduced, then list L is called pairwise-reduced.

III. A REVIEW OF SIEVE ALGORITHMS

In this section, we present a review of sieve algorithms in solving shortest lattice vector problem. In order to better describe the research progress of sieve algorithms, we categorize them according to when they were proposed and techniques they rely on. Except for special declarations, all the lattices we mention in the rest of the article are full-rank integer lattice of dimension n.

A. AKS SIEVE ALGORITHM

Based on Kumar and Sivakumar's work [43], Ajtai *et al.* proposed AKS Sieve algorithm [4] in 2001, which is the first algorithm solving the shortest vector problem in $2^{O(n)}$ time. Before AKS Sieve algorithm, the best algorithm to solve SVP is Kannan's enumeration algorithm [40], which solves the shortest vector problem in $2^{O(n\log n)}$ time.

For a lattice $\mathcal{L}(\mathbf{B})$, AKS Sieve algorithm uniformly samples a list of $2^{O(n)}$ pairs of vectors $L = \{(\mathbf{v}_i, \mathbf{y}_i), i \in I\} \in \mathcal{L}(\mathbf{B}) \times B_n(R)$, where $B_n(R)$ is a n-dimensional sphere with the origin

[1] http://www.latticechallenge.org/svp-challenge/

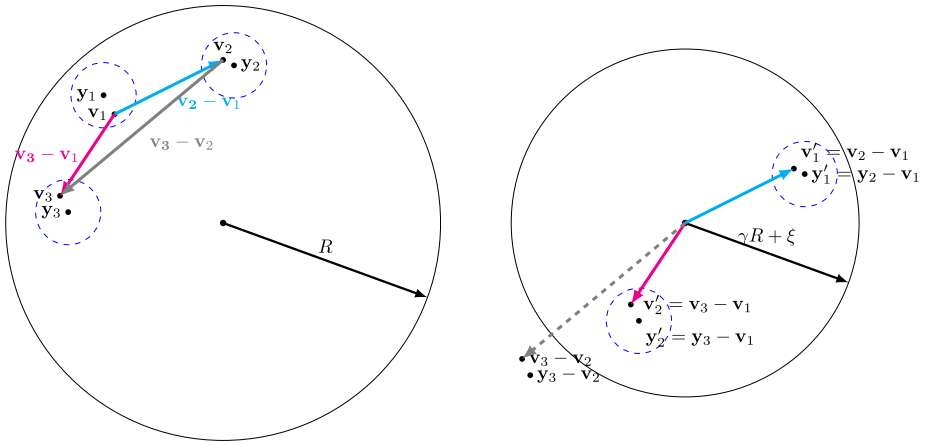


FIGURE 2. One round in AKS Sieve.

as its center and a radius not longer than R. For each pair $(\mathbf{v}_i, \mathbf{y}_i)$, $\|\mathbf{v}_i - \mathbf{y}_i\| \leq \xi$. AKS Sieve algorithm actually applies a "sieving" procedure on all the perturbation points $\mathbf{y}_i$:

$$\text{if } \|\mathbf{y}_i - \mathbf{y}_j\| \leq \gamma R, \text{ then } L' = L' \cup \{(\mathbf{v}_i - \mathbf{v}_j, \mathbf{y}_i - \mathbf{v}_j)\},$$

where $i, j \in I$, $\gamma < 1$.

After a round of sieving, list L will be replaced by L'. Since the distance between $\mathbf{v}_i$ and $\mathbf{y}_i$ is bounded by ξ, as the length of $\mathbf{y}_i$ shrinks, $\mathbf{v}_i$ becomes smaller as well. Actually, the new perturbation point $\mathbf{y}' = \mathbf{y}_i - \mathbf{v}_j$ satisfies

$$\|\mathbf{y}'\| = \|\mathbf{y}_i - \mathbf{v}_j\| \leq \|\mathbf{y}_i - \mathbf{y}_j\| + \|\mathbf{y}_j - \mathbf{v}_j\| \leq \gamma R + \xi$$

Figure 2 illustrates the change of list L after one round of AKS sieving. By iteratively applying AKS Sieve algorithm, a shortest vector will be found in list L.

AKS Sieve algorithm has two flaws: one is that it needs several parameters and it is hard to find the optimal choice; the other is that the complexity constant is too large. Schnorr [71] firstly concluded that the constant in the time complexity $O(n)$ is not smaller than 30 while Regev [68] claimed that it is not larger than 16. In 2008, Nguyen and Vidick [65] proved that the upper bound of the constant in the time complexity $O(n)$ is 5.9.

Though AKS Sieve algorithm is not practical enough, it is the first sieve algorithm to solve the shortest vector problem and enlightens a research area.

B. NV SIEVE ALGORITHM AND ITS VARIANTS

In 2008, Nguyen and Vidick [65] made a thorough analysis of AKS Sieve algorithm and proposed NV Sieve algorithm. NV Sieve algorithm is a heuristic variant of AKS and made a giant progress in both time and space complexity.

Instead of sieving on perturbation points like AKS Sieve algorithm, NV Sieve algorithm performs sieving procedure on lattice vectors directly based on the heuristic assumption that vectors in a sphere shell are uniformly distributed. N sampled lattice vectors which are shorter than R are divided into two sets: centers set C and short lattice vectors list L. In each sieving iteration, if vector $\mathbf{v} \leq \gamma R$, $\mathbf{v}$ will be put into new list L' immediately; if vector $\mathbf{v} > \gamma R$ but $\exists \mathbf{c} \in C$, $\|\mathbf{v} - \mathbf{c}\| \leq \gamma R$, $\mathbf{v} - \mathbf{c}$ will be put into new list L'; otherwise, vector $\mathbf{v}$ will be put into centers set C. At the end of each iteration, list L will be replaced by L'. Figure 3 shows the change of list L after one round of NV Sieve algorithm.

The time and space complexity of NV Sieve algorithm are $2^{0.415n+o(n)}$ and $2^{0.2075n+o(n)}$ respectively. However, it is not efficient enough to compete with enumeration algorithm. In 2011, Wang *et al.* [80] introduced a new technique and designed a two-level sieve algorithm based on NV Sieve algorithm. Instead of using γ to ensure vectors become shorter and short, shrink factors γ_1 and γ_2 are used in two-level sieve algorithm. Compared with NV Sieve algorithm, centers set C are divided into different groups and vectors in list L only try to conduct reduction with vectors in corresponding groups. In this way, two-level sieve algorithm achieves some balance between space and time, and its time and space complexity are $2^{0.3836n+o(n)}$ and $2^{0.2557n+o(n)}$ respectively. Two years later, Zhang *et al.* [82] proposed a three-level sieve algorithm based on the same idea, of which the time complexity is $2^{0.3778n+o(n)}$ and the space complexity is $2^{0.2833n+o(n)}$.

C. LIST SIEVE ALGORITHM, GAUSS SIEVE ALGORITHM AND THEIR IMPROVEMENTS

In 2009, Voulgaris and Micciancio [79] proposed List Sieve algorithm, which inherits the idea of AKS Sieve algorithm: sieving on perturbation points. List Sieve algorithm samples N pairs of vectors $(\mathbf{p}_i, \mathbf{e}_i)$ at the beginning, and in each sieving iteration $\mathbf{p}_i$ is reduced by all the vectors in list L. After that, let $\mathbf{v}_i = \mathbf{p}_i - \mathbf{e}_i$ and reduce $\mathbf{v}_i$ with all the vectors in list L (see Figure 4). Finally, $\mathbf{v}_i$ will be added to list L. During the whole process, vectors in list L are shrinking until the target norm is reached.

List Sieve algorithm solves the shortest vector problem in a lattice of dimension n in $2^{3.199n+o(n)}$ time and $2^{1.325n+o(n)}$ space. Even though it surpasses AKS Sieve algorithm, List Sieve algorithm is not efficient enough

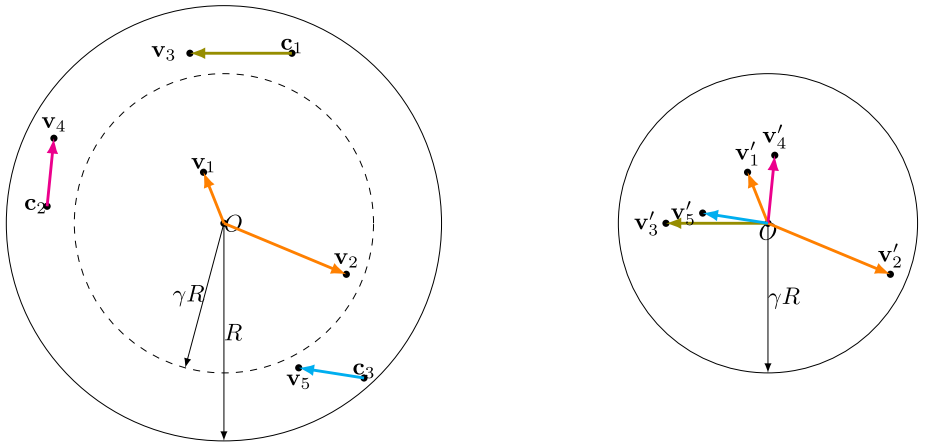


FIGURE 3. One round in NV Sieve.

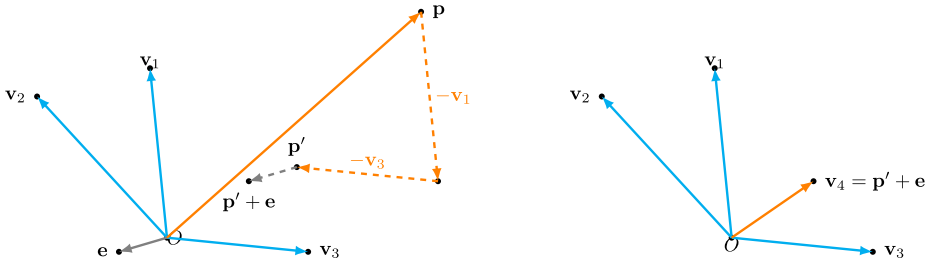


FIGURE 4. One round in List Sieve.

in practice. To improve List Sieve algorithm, Pujol and Stehlé [67] proposed List Sieve-Birthday algorithm. Compared with List Sieve algorithm, List Sieve-Birthday algorithm introduces a list U and adds vector pairs reduced by list L to U until the subtraction of the distinct closest vector pair $(\mathbf{u}_1, \mathbf{u}_2) \in U$ meets the target norm. Time and space complexity of List Sieve-Birthday algorithm are improved to $2^{2.465n+o(n)}$ and $2^{1.233n+o(n)}$.

Besides List Sieve-Birthday algorithm, Voulgaris and Micciancio [79] presented Gauss Sieve algorithm, a heuristic variant of List Sieve algorithm. Same as NV Sieve algorithm, Gauss Sieve algorithm performs sieving procedure directly on lattice vectors. It follows the approach of List Sieve algorithm to build a list of lattice vectors which become shorter and shorter after iterations. The difference is that when a new vector $\mathbf{v}_i$ is added to list L, not only $\mathbf{v}_i$ is reduced by all the vectors in list L, but also all the vectors in list L are reduced by $\mathbf{v}_i$ (see Figure 5). As a consequence, list L will be pairwise-reduced, which means for any vectors $\mathbf{u}, \mathbf{v} \in L$, $\max(\|\mathbf{u}\|, \|\mathbf{v}\|) \leq \min(\|\mathbf{u} \pm \mathbf{v}\|)$. The space complexity of Gauss Sieve algorithm is provably bounded by $2^{0.41n+o(n)}$ (seems to be at most $2^{0.21n+o(n)}$ in practice) while no upper bound on the running time is given. Experimental results show that the time complexity and space complexity seem to be $2^{0.48n+o(n)}$ and $2^{0.18n+o(n)}$.

Although Gauss Sieve algorithm does not own an attractive theoretical bound, it is one the most efficient sieve algorithm in practice and there are a series of improvements on Gauss Sieve algorithm afterwards. Fitzpatrick *et al.* [26] put forward several heuristic improvements including optimized Gaussian sampler, multiple randomized bases and angle approximation to Gauss Sieve algorithm, which achieves a good acceleration. Ducas [23] also modifies sampling method and utilizes hash tables to prevent invalid vector reductions to improve the efficiency of Gauss Sieve algorithm. In addition, there are various parallel implementations [39], [59], [62], [81], and [83] of Gauss Sieve algorithm.

D. SIEVE ALGORITHMS WITH NEAREST NEIGHBOR SEARCH TECHNIQUES

For a list of N vectors $L = \{\mathbf{v}_1, \cdots, \mathbf{v}_N\} \in \mathbb{R}^n$, the nearest neighbor search (NNS) [38] aims to find a vector $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{t}\| = \min\{\|\mathbf{v}_i - \mathbf{t}\| | \mathbf{v}_i \in L\}$, where target vector $\mathbf{t} \notin L$. Relying on a family of locality-sensitive hash functions, locality-sensitive hashing (LSH) [7], [8], [17], [38], [78] is a technique to solve the nearest neighbor search problem.

In 2015, Laarhoven [44] introduced LSH technique into sieve algorithm for the first time. By means of replacing the brute-force list search during reduction with method combining with angular LSH, sieve algorithm obtain a speedup in solving SVP. When combining NV Sieve algorithm with angular LSH, the time complexity decreases to $2^{0.3366n+o(n)}$ while the space complexity remains unchanged.

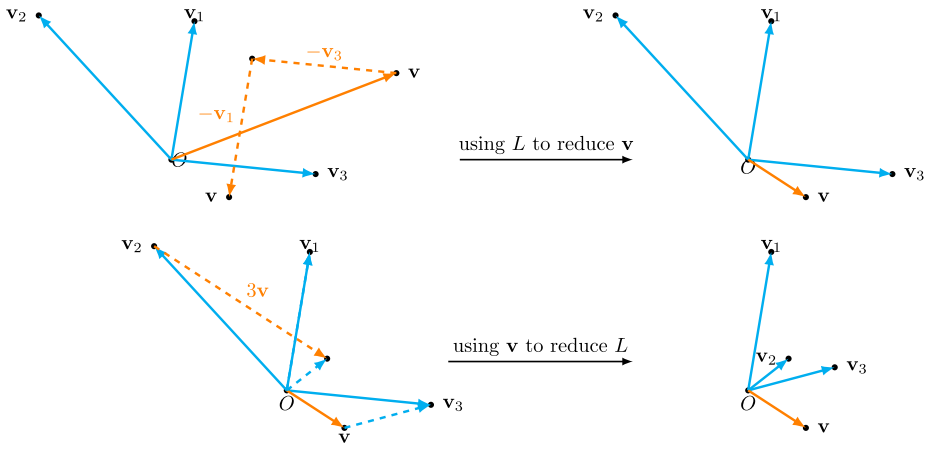


FIGURE 5. One round in Gauss Sieve.

Combining Gauss Sieve algorithm with angular LSH, he proposed Hash Sieve algorithm, whose time complexity and space complexity can be bounded by $2^{0.3366n+o(n)}$.

Inspired by angular LSH, other LSH methods attracted researchers to study whether they are able to accelerate sieve algorithm. Laarhoven and de Weger [47] combined NV Sieve algorithm with spherical LSH and proposed Sphere Sieve algorithm. The time complexity of Sphere Sieve algorithm is $2^{0.298n+o(n)}$ as the space complexity remains $2^{0.2075n+o(n)}$.

Besides LSH, other NNS techniques are used to speed up sieve algorithm as well. Becker *et al.* [13] adapted May and Ozerov's technique [60] which is used to solve NNS of binary code words, and applied it to searching vector pairs suitable for reduction during sieving. By combining this sub-quadratic NNS technique with NV Sieve algorithm, the time complexity drops to $2^{0.3112n+o(n)}$ and the space complexity is still $2^{0.2075n+o(n)}$.

Sphere Sieve algorithm surpassed all the sieve algorithms in terms of time complexity at that time. However, the spherical LSH that Sphere Sieve algorithm relies on seems incompatible with Gauss Sieve algorithm, which greatly limits the practicability of Sphere Sieve algorithm. In 2016, for the sake of obtaining a more practical speedup in sieve algorithm, Becker and Laarhoven [14] introduced cross-polytope LSH [8], [78] into sieve algorithm. Although cross-polytope LSH has the same acceleration effect as spherical LSH when working with NV Sieve algorithm, it can be combined with modified Gauss Sieve algorithm to generate CP Sieve algorithm, whose efficiency exceeds Gauss Sieve algorithm and Hash Sieve algorithm in practice.

Based on LSH, Becker *et al.* [11] proposed locality-sensitive filter (LSF), which adopts a new approach to pick up nearby vectors. By applying spherical LSF to sieve algorithm, LD Sieve algorithm [11] is capable to solve the shortest vector problem in $2^{0.292n+o(n)}$ time, which outperforms the previous best time complexity achieved by Sphere Sieve algorithm and CP Sieve algorithm. Until today, LD Sieve algorithm still owns the best asymptotic time complexity for solving the shortest vector problem and is wildly used to evaluate the security parameters of certain lattice-based cryptosystems.

During the sieving procedure, the reduction step cost the most time. Through quickly judging whether two vectors are nearby, sieve algorithms with NNS techniques avoid a large amount of invalid reductions and as a result the shortest vector can be found in less time. Figure 6 illustrates the time and space complexity of sieve algorithms with NNS techniques. However, these methods cannot change the space complexity of sieve algorithms and to some extent it is difficult to construct an effective LSH or LSF.

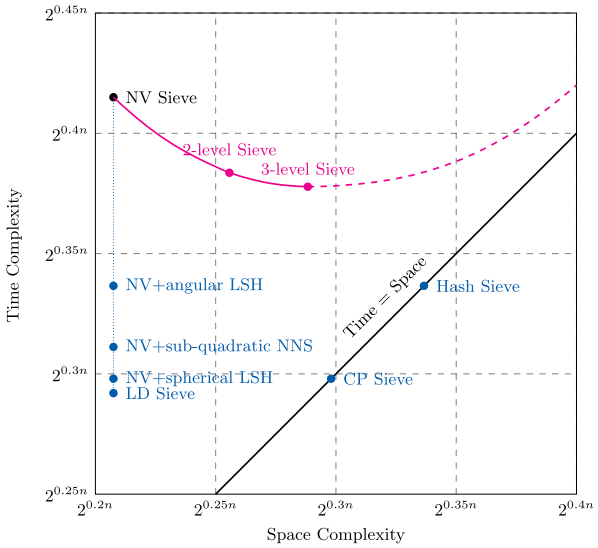


FIGURE 6. Space and time complexity of sieve algorithms with NNS techniques, reprinted from [11].

E. TUPLE SIEVE ALGORITHM AND ITS IMPROVEMENTS

Even though the time complexity of sieve algorithms is single-exponential, the exponential memory consumption has always limited the practicality of sieve algorithms while enumeration algorithms just cost negiliable polynomial space.

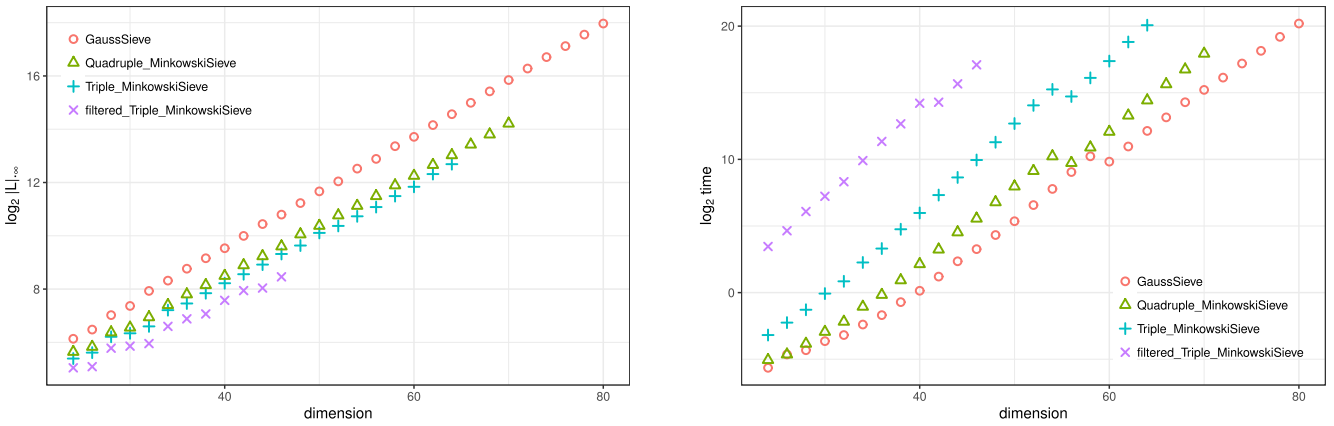


FIGURE 7. Average maximum list size ($\log_2 |L|_\infty$) and average running time ($\log_2$ time) of Gauss Sieve and different Tuple Sieve algorithms, reprinted from [9].

Aiming to overcome this obstacle, Bai *et al.* [9] proposed Tuple Sieve algorithm in 2016. Instead of reducing vector pairs $(\mathbf{v}_1 \pm \mathbf{v}_2)$ to decrease their norms, Tuple Sieve algorithm takes advantage of tuples of vectors $(\mathbf{v}_1 \pm \mathbf{v}_2 \pm \cdots \pm \mathbf{v}_k)$. In this way, Tuple Sieve algorithm actually achieves a comprise between time and space: less lattice vectors are needed while more time is spent during reduction to generate more short vectors. When combining this technique with NV Sieve algorithm, for tuple size $k = 3$, the space complexity decreases from $2^{0.2075n+o(n)}$ to $2^{0.1887n+o(n)}$ while the time complexity increases from $2^{0.415n+o(n)}$ to $2^{0.5662n+o(n)}$. To further improve efficiency, a filter was designed to avoid invalid vector reductions and with this filter the time complexity of 3-Tuple Sieve algorithm drops to $2^{0.4812n+o(n)}$. Moreover, by applying tuple sieve technique to Gauss Sieve algorithm, they proposed Tuple Minkowski Sieve algorithm, a more practical sieve algorithm. Figure 7 illustrates the experimental space and time costs of different Tuple Sieve algorithms.

In 2017, Herold and Kirshanova [34] proposed an improved algorithm for the approximate k-list problem in Euclidean norm which can be applied to accelerating Tuple Sieve algorithm. When tuple size $k = 3$, the time complexity is brought down from $2^{0.4812n+o(n)}$ to $2^{0.3962n+o(n)}$ with the space complexity remains $2^{0.1887n+o(n)}$. Furthermore, combining with LSH techniques, the time complexity drops to $2^{0.3717n+o(n)}$ as the space complexity remains unchanged.

At the same year, Laarhoven [45] generalized the spherical LSF from LD Sieve algorithm and applied it to Tuple Sieve algorithm. Using this method, Herold and Kirshanova's algorithm can be improved to solve SVP in $2^{0.3588n+o(n)}$ time and $2^{0.1887n+o(n)}$ memory.

In 2018, Herold *et al.* [35] extended their previous work together. With configuration framework and spherical LSF, they offered tunable time-memory trade-offs for sieve algorithms with arbitrary tuple sizes. For triple sieving, the time and space complexity are $2^{0.3588n+o(n)}$ and $2^{0.1887n+o(n)}$ respectively, which can be adjusted to $2^{0.3300n+o(n)}$ and $2^{0.2075n+o(n)}$ if more memory is taken up.

The core of various tuple sieve algorithms is a trade-off between time and space: the larger the tuple size, the less the memory, and the longer the time. And owing to that the time complexity of sieve algorithm is exponential, tuple sieve algorithms with tuple size larger than 4 is not practical.

F. SIEVE ALGORITHMS WITH RANK REDUCTION TECHNIQUES

Rank reduction is a principle extensively used in solving hard lattice problems: when solving a problem on a lattice of dimension n, one first solves problems on its sublattice of dimension $n - k$ and later uses the outcome to solve the original problem.

In 2018, Laarhoven and Mariano [48] proposed progressive sieving technique, which introduces a progressive approach to sieve algorithm. It starts from sieving on a low-rank sublattice of the original lattice and then gradually adds new basis vectors and launches sieve algorithm on sublattice of higher rank until all the basis vectors are added and seeks to find the shortest vector. Progressive sieving actually is a new mode of sieve algorithms and almost all the sieve algorithms can be modified to combine with it. It has faster convergence speed than conventional sieve algorithms that sometimes it even finds target vector on a sublattice. Moreover, algorithms with progressive sieving consume less memory since it mainly works on sublattices.

At the same time as Laarhoven and Mariano's work, Ducas [23] proposed SubSieve technique, which is another form of rank reduction. SubSieve technique relies on the fact that the output list of sieve algorithm contains the vast majority of short vectors of the lattice. For a lattice $\mathcal{L}$, SubSieve technique runs sieve algorithm on its projected sublattice $\mathcal{L}_d$ of rank $n - d$ and outputs a list L. For each vector $\mathbf{v}_d \in L$, the preimage $\mathbf{v} \in \mathcal{L}$ is calculated and the shortest one is outputted to judge whether it is a shortest vector. SubSieve technique is capable of bringing a sub-exponential speedup to sieve algorithm and it can be applied to most of sieve algorithms by design.

In 2019, based on the ideas of progressive sieving and Sub-Sieve, Albrecht *et al.* [5] proposed G6K, which is an abstract stateful machine solving various hard lattice problems based on sieve algorithms. Using G6K, they solved many previously unsolved lattice challenge instances and for previously solved instances, G6K solved much faster.

Rank reduction actually is a heuristic strategy which has been used in lattice reduction algorithms and enumeration algorithms for a long period. By converting SVP on a n-dimensional lattice into its sublattices, sieve algorithms with rank reduction techniques will cost less time and memory.

G. PARALLELIZATION

Though the time complexity of sieve algorithm is asymptotically faster than enumeration algorithm, sieve algorithms cannot compete with enumeration algorithms at the beginning. Under this circumstance, the parallelization of sieve algorithm is a good alternative.

In 2011, based on a ring structure, Milde and Schneider [62] proposed the first parallel implementation of Gauss Sieve algorithm. Independent Gauss Sieve instances are connected like a ring, and to keep all the instances active, a queue of vector are added to each instances. Vectors pass one instance will be handed to the next one. Only vectors pass all the all the instances will be added to the current list L, which has a negative influence on the convergence and scalability of the parallel algorithm. Experimental results show that efficiency does not improve linearly for more than four threads.

The bottleneck of Milde and Schneider's parallel algorithm is that vectors in lists of different instances are not Gauss-reduced and some short vectors are omitted during the reduction process, which greatly limits the efficiency of the algorithm. To tackle this problem, Ishiguro *et al.* [39] designed another parallel Gauss Sieve algorithm in 2014. By adding an additional list to each Gauss Sieve instance, vectors in each list are guaranteed to be Gauss-reduced. As a result, this algorithm has a excellent scalability and is able to solve SVP on lattice of dimension more than 100.

In 2014, Mariano *et al.* [57] proposed a parallel variant of List Sieve algorithm and made a comprehensive empirical comparison between it and Gauss Sieve algorithm. Then based on lock-free technique, Mariano *et al.* [59] presented a parallel variant of Gauss Sieve algorithm using shared-memory. This implementation is claimed to support a linear speedup for up to 64 threads. At the same year, Bos *et al.* [83] proposed a different version of parallel Gauss Sieve algorithm suitable for distributed-memory architectures.

In 2017, Yang *et al.* [81] implemented Ishiguro *et al.*'s parallel Gauss Sieve algorithm on a single GPU and Bos *et al.*'s on multiple GPUs. Both implementation behaved efficiently in solving the shortest vector problem.

In 2015 and 2017, Mariano *et al.* [56], [58] proposed the parallel implementation of Hash Sieve algorithm and LD Sieve algorithm respectively. Both of the implementations scale well on CPU cores and are able to achieve a linear speedup for up to 64 threads.

In addition, Ducas, Stevens and Woerden solved the 170-dimensional SVP Challenge this year based on efficient parallelization of G6K with GPU-implementation,[2] which is the highest dimension for SVP Challenge till now.

H. SIEVE ALGORITHMS WITH QUANTUM SPEEDUP

The development of quantum algorithms and quantum computer has brought a huge impact on the security of public key cryptosystems. Since lattice-based cryptosystems have become one of the most promising candidate in post-quantum era, research on how quantum algorithms and quantum computer can be applied to accelerating algorihtms solving related problems is of great significance.

In 2015, Laarhoven *et al.* [49] studied how to use Grover's quantum search algorithm [33] to improve sieve algorithms. By combining with Grover's quantum search algorithm, sieve algorithms obtain an asymptotic improvement both in time and space complexity.

In 2019, Kirshanova *et al.* [41] studied how to use quantum techniques to speed up variants of Tuple Sieve algorithm. They firstly applied Grover's quantum search [33] algorithm to speed up Tuple Sieve algorithm and its variants. Secondly, quantum k-clique finding techniques [27] were used to accelerate Tuple Sieve algorithms. Finally, they adapted parallel quantum search technique [10] and combined it with NV Sieve algorithm, which leads to a significant improvement in complexity.

In 2019, Albrecht *et al.* [6] designed quantum circuits for NNS algorithms that are deeply used in sieve algorithms and analyzed their costs.

Quantum sieve algorithms might be kind of unrealistic on account of that quantum computer is not practical at now. However, research on it would provide a reference for evaluating the security of lattice-based cryptosystems.

I. SIEVE ALGORITHMS ON IDEAL LATTICE

Ideal lattice is a special sort of lattice which is usually used to construct efficient lattice-based cryptographic schemes [30], [36], [52], [61], [76]. It was supposed that solving the shortest vector problem in ideal lattices is as hard as in general lattices until Voulgaris and Micciancio [79] presented the idea that the special structure of ideal lattice can be used to improve efficiency, especially for sieve algorithms.

In 2011, Schneider [70] proposed Ideal List Sieve algorithm and Ideal Gauss Sieve algorithm, which are variants of List Sieve algorithm and Gauss Sieve algorithm using the special structure of ideal lattices. Based on the rotation structure of cyclic lattices and anti-cyclic lattices, both algorithms are shown to be dozens of times faster than original algorithms in solving SVP on ideal lattices.

In 2014, Ishiguro *et al.* [39] and Bos *et al.* [83] proposed variants of parallel GaussSieve algorithm respectively. And

[2]http://www.latticechallenge.org/svp-challenge/halloffame.php

TABLE 1. A summary of sieve algorithms.

Algorithm Name[References]	$\log_2$(time)	$\log_2$(space)	Techniques	Type	Roadmap
AKS Sieve [4]	$5.9n$	$2.95n$	—-	Provable	III-A
NV Sieve [65]	$0.415n$	$0.2075n$	—-	Heurisic	III-B
2-level NV Sieve [80]	$0.3836n$	$0.2557n$	—-	Heurisic	III-B
3-level NV Sieve [82]	$0.3778n$	$0.2833n$	—-	Heurisic	III-B
List Sieve [79]	$3.199n$	$2.1325n$	—-	Provable	III-C
List Sieve-Birthday [67]	$2.465n$	$1.233n$	birthday method	Provable	III-C
Gauss Sieve [79]	$0.415n$	$0.2075n$	—-	Heurisic	III-C
NV+angular LSH [44]	$0.3366n$	$0.2075n$	angular LSH	Heurisic	III-D
Hash Sieve [44]	$0.3366n$	$0.3366n$	angular LSH	Heurisic	III-D
Sphere Sieve [47]	$0.298n$	$0.2075n$	spherical LSH	Heurisic	III-D
NV+sub-quadratic NNS [13]	$0.3112n$	$0.2075n$	sub-quadratic NNS	Heurisic	III-D
CP Sieve [14]	$0.298n$	$0.298n$	cross-polytope LSH	Heurisic	III-D
LD Sieve [11]	$0.292n$	$0.2075n$	spherical LSF	Heurisic	III-D
Overlattice Sieve [12]	$0.3774n$	$0.2925n$	overlattice	Heurisic	III-J

their algorithms effectively solved the SVP challenge on two different 128-dimensional ideal lattices.

In 2015, Becker and Laarhoven [14] presented Ideal CP Sieve algorithm based on CP Sieve algorithm, which has a linear speedup in time and a linear improvement in memory.

J. OTHER SIEVE ALGORITHMS

Besides these sieve algorithms mentioned above, there are several other sieve algorithms.

In 2014, Becker *et al.* [12] proposed a heuristic sieve algorithm based on overlattices. For both SVP and CVP, it finds a solution in $2^{0.3774n+o(n)}$ time and $2^{0.2925n+o(n)}$ memory. Moreover, by adjusting parameters, this algorithm achieves a trade-off between time and space.

In 2019, Laarhoven [46] compared sieve algorithms and evolutionary algorithms and explored the possibility to accelerate sieve algorithms with evolutionary techniques.

In 2020, Doulgerakis *et al.* [22] designed a hybrid algorithm which uses sieve algorithm as a subroutine. Combined with SubSieve [23] and CVPP [25] techniques, this algorithm obtains more free dimensions and solves SVP on high-dimensional lattices much faster.

IV. CONCLUSION

Sieve algorithm is regarded as one of the most promising algorithm to solve SVP for its single-exponential asymptotic time complexity. However, sieve algorithms did not perform well in time at first as its exponential space complexity is also an obstacle. To improve the performance of sieve algorithms both in time and memory, a spectrum of ideas and techniques were proposed to combine with sieve algorithms to gain a better result. Table 1 summarizes sieve algorithms reviewed in this article.

Based on our review of sieve algorithms, we consider that there are still several directions worth in-depth research:

- Precise complexity bound, especially for Gauss Sieve algorithm. As one of the most practical sieve algorithm, there is no tight upper bounds on the running time and memory consumption for Gauss Sieve algorithm while Gauss Sieve algorithm behaves much better than its present time and space complexity in practice. A precise complexity bound will provide us with a better comprehension about Gauss Sieve algorithm and its variants.
- Sieve algorithm with less space requirement. Though sieve algorithms have made great progress in recent years, its exponential memory consumption still restricts its application, especially compared with the negligible polynomial space requirement of enumeration algorithm. Tuple Sieve algorithm and its variants aim to tackle this bottleneck but do not obtain a desirable outcome. A sieve algorithm with sub-exponential or polynomial space requirement will be much more competitive in solving SVP of high dimension.
- Combination with enumeration algorithms. To some extent, using projected sublattices in SubSieve [23] technique is inspired by enumeration algorithms. Both sieve algorithms and enumeration algorithms have their own advantages. A hybrid algorithm in which sieve algorithms and enumeration algorithms act as subroutines maybe solves SVP more efficiently.
- Working as a subroutine of lattice reduction algorithms. For conventional lattice basis reduction algorithms like BKZ and its variants, enumeration algorithm is used as its subroutine, which makes it inefficient on lattice of high dimension. It is worthwhile to explore whether sieve algorithm can be used as the subroutine of BKZ algorithm and its variants since it is becoming more and

more efficient. As for this aspect, G6K has achieved some good results. Besides, the output list of sieve algorithm contains a vast majority of short vectors in the lattice, which caters to the need of random sampling reduction algorithm: sampling short vectors. By designing appropriate strategies, sieve algorithm is capable of benefiting lattice reduction algorithms greatly.

- Parallel implementation with GPU. Compared with multiple CPU cores, the parallelization of sieve algorithms with GPU supports a larger prallel scale, which will guarantee we find the shortest vector faster. However, the parallelization with GPU may be unable to participate in the entire procedure of sieving for that the short vector list L is dynamically changing. Under this circumstance, sampling short vectors and some reductions seem to be compatible with implementation with GPU. Actually, Ducas *et al.* has utilized GPU in solving SVP Challenge in practice.[3]

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and helping us improve this article.

REFERENCES

[1] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, Philadelphia, PA, USA, 1996, pp. 99–108.

[2] M. Ajtai, "The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract)," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, Dallas, TX, USA, 1998, pp. 10–19.

[3] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th Annu. ACM Symp. Theory Comput. (STOC)*, El Paso, TX, USA, 1997, pp. 284–293.

[4] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *Proc. 33rd Annu. ACM Symp. Theory Comput. (STOC)*, Heraklion, Greece, 2001, pp. 601–610.

[5] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The general sieve kernel and new records in lattice reduction," in *Advances in Cryptology—EUROCRYPT 2019* (Lecture Notes in Computer Science), vol. 11477. Darmstadt, Germany: Springer, 2019, pp. 717–746.

[6] M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, "Quantum speedups for lattice sieves are tenuous at best," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1161, Oct. 2019.

[7] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*. Berkeley, CA, USA: IEEE Computer Society, Oct. 2006, pp. 459–468.

[8] A. Andoni, P. Indyk, T. Laarhoven, I. P. Razenshteyn, and L. Schmidt, "Practical and optimal LSH for angular distance," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 1225–1233.

[9] S. Bai, T. Laarhoven, and D. Stehlé, "Tuple lattice sieving," *LMS J. Comput. Math.*, vol. 19, no. A, pp. 146–162, 2016.

[10] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, "Efficient distributed quantum computing," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 469, no. 2153, 2013, Art. no. 20120686.

[11] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New directions in nearest neighbor searching with applications to lattice sieving," in *Proc. 27th Annu. ACM-SIAM Symp. Discrete Algorithms*. Arlington, VA, USA: SIAM, Jan. 2016, pp. 10–24.

[12] A. Becker, N. Gama, and A. Joux, "A sieve algorithm based on overlattices," *LMS J. Comput. Math.*, vol. 17, no. A, pp. 49–70, 2014.

[13] A. Becker, N. Gama, and A. Joux, "Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search," *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 522, 2015.

[14] A. Becker and T. Laarhoven, "Efficient (ideal) lattice sieving using cross-polytope LSH," in *Progress in Cryptology—AFRICACRYPT 2016* (Lecture Notes in Computer Science), vol. 9646. Fes, Morocco: Springer, 2016, pp. 3–23.

[15] D. Bleichenbacher and A. May, "New attacks on RSA with small secret crt-exponents," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 3958. Berlin, Germany: Springer, 2006, pp. 1–13.

[16] J. A. Buchmann, E. Dahmen, and A. Hülsing, "XMSS—A practical forward secure signature scheme based on minimal security assumptions," in *Post-Quantum Cryptography* (Lecture Notes in Computer Science), vol. 7071. Taipei, Taiwan: Springer, 2011, pp. 117–129.

[17] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. 34th Annu. ACM Symp. Theory Comput. (STOC)*, Montréal, QC, Canada, 2002, pp. 380–388.

[18] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *Advances in Cryptology—ASIACRYPT 2011* (Lecture Notes in Computer Science), vol. 7073. Seoul, South Korea: Springer, 2011, pp. 1–20.

[19] D. Coppersmith, "Finding a small root of a bivariate integer equation; factoring with high bits known," in *Advances in Cryptology—EUROCRYPT '96* (Lecture Notes in Computer Science), vol. 1070. Berlin, Germany: Springer, 1996, pp. 178–189.

[20] D. Coppersmith, "Finding a small root of a univariate modular equation," in *Advances in Cryptology—EUROCRYPT'96* (Lecture Notes in Computer Science), vol. 1070. Berlin, Germany: Springer, 1996, pp. 155–165.

[21] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *Comput. Complex.*, vol. 2, no. 2, pp. 111–128, Jun. 1992.

[22] E. Doulgerakis, T. Laarhoven, and B. de Weger, "Sieve, enumerate, slice, and lift: Hybrid lattice algorithms for SVP via CVPP," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 487, 2020.

[23] L. Ducas, "Shortest vector from lattice sieving: A few dimensions for free," in *Advances in Cryptology—EUROCRYPT 2018* (Lecture Notes in Computer Science), vol. 10820. Tel Aviv, Israel: Springer, 2018, pp. 125–145.

[24] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal Gaussians," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 8042. Berlin, Germany: Springer, 2013, pp. 40–56.

[25] L. Ducas, T. Laarhoven, and W. P. J. van Woerden, "The randomized slicer for CVPP: Sharper, faster, smaller, batchier," in *Public Key Cryptography (2)* (Lecture Notes in Computer Science), vol. 12111. Berlin, Germany: Springer, 2020, pp. 3–36.

[26] R. Fitzpatrick, C. H. Bischof, J. A. Buchmann, O. Dagdelen, F. Göpfert, A. Mariano, and B. Yang, "Tuning Gausssieve for speed," in *Progress in Cryptology—LATINCRYPT 2014* (Lecture Notes in Computer Science), vol. 8895. Florianópolis, Brazil: Springer, 2014, pp. 288–305.

[27] F. Le Gall and S. Nakajima, "Quantum algorithm for triangle finding in sparse graphs," *Algorithmica*, vol. 79, no. 3, pp. 941–959, Nov. 2017.

[28] N. Gama, P. Q. Nguyen, and O. Regev, "Lattice enumeration using extreme pruning," in *Advances in Cryptology—EUROCRYPT 2010* (Lecture Notes in Computer Science), vol. 6110. Monaco, French Riviera: Springer, 2010, pp. 257–278.

[29] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 196. Santa Barbara, CA, USA: Springer, 1984, pp. 10–18.

[30] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, 2009, pp. 169–178.

[31] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. 40th Annu. ACM Symp. Theory Comput. (STOC)*, Victoria, BC, Canada, 2008, pp. 197–206.

[32] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Advances in Cryptology—CRYPTO'97* (Lecture Notes in Computer Science), vol. 1294. Berlin, Germany: Springer, 1997, pp. 112–131.

[33] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, Philadelphia, PA, USA, 1996, pp. 212–219.

[3] http://www.latticechallenge.org/svp-challenge/halloffame.php

[34] G. Herold and E. Kirshanova, "Improved algorithms for the approximate k-list problem in Euclidean norm," in *Public-Key Cryptography* (Lecture Notes in Computer Science), vol. 10174. Amsterdam, The Netherlands: Springer, 2017, pp. 16–40.

[35] G. Herold, E. Kirshanova, and T. Laarhoven, "Speed-ups and time-memory trade-offs for tuple lattice sieving," in *Public-Key Cryptography* (Lecture Notes in Computer Science), vol. 10769. Rio de Janeiro, Brazil: Springer, 2018, pp. 407–436.

[36] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory—ANTS* (Lecture Notes in Computer Science), vol. 1423. Berlin, Germany: Springer, 1998, pp. 267–288, doi: 10.1007/BFb0054868.

[37] N. A. Howgrave-Graham and P. Nigel Smart, "Lattice attacks on digital signature schemes," *Des., Codes Cryptogr.*, vol. 23, no. 3, pp. 283–290, 2001.

[38] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, Dallas, TX, USA, 1998, pp. 604–613.

[39] T. Ishiguro, S. Kiyomoto, Y. Miyake, and T. Takagi, "Parallel Gauss Sieve algorithm: Solving the SVP challenge over a 128-dimensional ideal lattice," in *Public-Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 8383. Buenos Aires, Argentina: Springer, 2014, pp. 411–428.

[40] R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proc. 15th Annu. ACM Symp. Theory Comput. (STOC)*, Boston, MA, USA, 1983, pp. 193–206.

[41] E. Kirshanova, E. Mårtensson, E. W. Postlethwaite, and S. R. Moulik, "Quantum algorithms for the approximate K-list problem and their application to lattice sieving," in *Advances in Cryptology—ASIACRYPT 2019* (Lecture Notes in Computer Science), vol. 11921. Kobe, Japan: Springer, 2019, pp. 521–551.

[42] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.

[43] R. Kumar and D. Sivakumar, "On polynomial approximation to the shortest lattice vector length," in *Proc. 12th Annu. Symp. Discrete Algorithms*. Washington, DC, USA: ACM/SIAM, 2001, pp. 126–127.

[44] T. Laarhoven, "Sieving for shortest vectors in lattices using angular locality-sensitive hashing," in *Advances in Cryptology—CRYPTO 2015* (Lecture Notes in Computer Science), vol. 9215. Santa Barbara, CA, USA: Springer, 2015, pp. 3–22.

[45] T. Laarhoven, "Faster tuple lattice sieving using spherical locality-sensitive filters," *CoRR*, vol. abs/1705.02828, 2017.

[46] T. Laarhoven, "Evolutionary techniques in lattice sieving algorithms," in *Proc. 11th Int. Joint Conf. Comput. Intell.*, Vienna, Austria: SciTePress, 2019, pp. 31–39.

[47] T. Laarhoven and B. de Weger, "Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing," in *Progress in Cryptology—LATINCRYPT 2015* (Lecture Notes in Computer Science), vol. 9230. Guadalajara, Mexico: Springer, 2015, pp. 101–118.

[48] T. Laarhoven and A. Mariano, "Progressive lattice sieving," in *Post-Quantum Cryptography* (Lecture Notes in Computer Science), vol. 10786. Fort Lauderdale, FL, USA: Springer, 2018, pp. 292–311.

[49] T. Laarhoven, M. Mosca, and J. van de Pol, "Finding shortest lattice vectors faster using quantum search," *Designs, Codes Cryptogr.*, vol. 77, nos. 2–3, pp. 375–400, Dec. 2015.

[50] J. C. Lagarias, "Knapsack public key cryptosystems and diophantine approximation," in *Advances in Cryptology*. New York, NY, USA: Plenum, 1983, pp. 3–23.

[51] J. C. Lagarias and A. M. Odlyzko, "Solving low-density subset sum problems," *J. ACM*, vol. 32, no. 1, pp. 229–246, Jan. 1985.

[52] A. Langlois, D. Stehlé, and R. Steinfeld, "GGHLite: More efficient multilinear maps from ideal lattices," in *Advances in Cryptology—EUROCRYPT 2014* (Lecture Notes in Computer Science), vol. 8441. Berlin, Germany: Springer, 2014, pp. 239–256.

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

[54] V. Lyubashevsky and D. Micciancio, "Asymptotically efficient lattice-based digital signatures," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 4948. Berlin, Germany: Springer, 2008, pp. 37–54.

[55] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen, "SWIFFT: A modest proposal for FFT hashing," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 5086. Berlin, Germany: Springer, 2008, pp. 54–72.

[56] A. Mariano, C. Bischof, and T. Laarhoven, "Parallel (Probable) lock-free hash sieve: A practical sieving algorithm for the SVP," in *Proc. 44th Int. Conf. Parallel Process.*, Beijing, China: IEEE Computer Society, Sep. 2015, pp. 590–599.

[57] A. Mariano, O. Dagdelen, and C. H. Bischof, "A comprehensive empirical comparison of parallel listsieve and Gausssieve," in *Euro-Par 2014: Parallel Processing Workshops* (Lecture Notes in Computer Science), vol. 8805. Porto, Portugal: Springer, 2014, pp. 48–59.

[58] A. Mariano, T. Laarhoven, and C. Bischof, "A parallel variant of LDSieve for the SVP on lattices," in *Proc. 25th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*. St. Petersburg, Russia: IEEE Computer Society, 2017, pp. 23–30.

[59] A. Mariano, S. Timnat, and C. Bischof, "Lock-free GaussSieve for linear speedups in parallel high performance SVP calculation," in *Proc. IEEE 26th Int. Symp. Comput. Archit. High Perform. Comput.* Paris, France: IEEE Computer Society, Oct. 2014, pp. 278–285.

[60] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Advances in Cryptology—EUROCRYPT 2015* (Lecture Notes in Computer Science), vol. 9056. Sofia, Bulgaria: Springer, 2015, pp. 203–228.

[61] D. Micciancio, "Generalized compact knapsacks, cyclic lattices, and efficient one-way functions," *Comput. Complex.*, vol. 16, no. 4, pp. 365–411, Dec. 2007.

[62] B. Milde and M. Schneider, "A parallel implementation of Gausssieve for the shortest vector problem in lattices," in *Parallel Computing Technologies* (Lecture Notes in Computer Science), vol. 6873. Kazan, Russia: Springer, 2011, pp. 452–458.

[63] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology—CRYPTO'85* (Lecture Notes in Computer Science), vol. 218. Santa Barbara, CA, USA: Springer, 1985, pp. 417–426.

[64] P. Q. Nguyen, "The dark side of the hidden number problem: Lattice attacks on DSA," in *Cryptography and Computational Number Theory*. Berlin, Germany: Springer, 2001, pp. 321–330.

[65] P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *J. Math. Cryptol.*, vol. 2, no. 2, pp. 181–207, Jan. 2008.

[66] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, Feb. 1981.

[67] X. Pujol and D. Stehlé, "Solving the shortest lattice vector problem in time $2^{2.465n}$," *IACR Cryptol. ePrint Arch.*, vol. 2009, p. 605, 2009.

[68] O. Regev, "Lecture notes of lattices in computer science," Lecture Notes, 2004.

[69] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[70] M. Schneider, "Sieving for shortest vectors in ideal lattices," *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 458, 2011.

[71] C. Schnorr, "Lattice reduction by random sampling and birthday methods," in *STACS 2003* (Lecture Notes in Computer Science), vol. 2607. Berlin, Germany: Springer, 2003, pp. 145–156.

[72] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, nos. 1–3, pp. 181–199, Aug. 1994.

[73] C. Schnorr and H. H. Hörner, "Attacking the Chor-Rivest cryptosystem by improved lattice reduction," in *Advances in Cryptology—EUROCRYPT'95* (Lecture Notes in Computer Science), vol. 921. Saint-Malo, France: Springer, 1995, pp. 1–12.

[74] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.* Santa Fe, NM, USA: IEEE Computer Society, 1994, pp. 124–134.

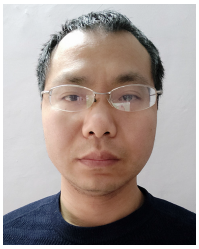
[75] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, Jan. 1999.

[76] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," in *Advances in Cryptology—ASIACRYPT 2009* (Lecture Notes in Computer Science), vol. 5912. Berlin, Germany: Springer, 2009, pp. 617–635.

[77] D. Stehlé and M. Watkins, "On the extremality of an 80-dimensional lattice," in *Algorithmic Number Theory* (Lecture Notes in Computer Science), vol. 6197. Nancy, France: Springer, 2010, pp. 340–356.
[78] K. Terasawa and Y. Tanaka, "Spherical LSH for approximate nearest neighbor search on unit hypersphere," in *Algorithms and Data Structures* (Lecture Notes in Computer Science), vol. 4619, F. K. H. A. Dehne, J. Sack, and N. Zeh, Eds. Halifax, Canada: Springer, 2007, pp. 27–38, doi: 10.1007/978-3-540-73951-7_4.
[79] P. Voulgaris and D. Micciancio, "Faster exponential time algorithms for the shortest vector problem," *Electron. Colloq. Comput. Complex.*, vol. 16, no. 65, p. 58, 2009.
[80] X. Wang, M. Liu, C. Tian, and J. Bi, "Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, Hong Kong, 2011, pp. 1–9.
[81] S. Yang, P. Kuo, B. Yang, and C. Cheng, "Gauss sieve algorithm on GPUS," in *Topics in Cryptology—CT-RSA 2017* (Lecture Notes in Computer Science), vol. 10159. Berlin, Germany: Springer, 2017, pp. 39–57.
[82] F. Zhang, Y. Pan, and G. Hu, "A three-level sieve algorithm for the shortest vector problem," in *Selected Areas in Cryptography—SAC 2013* (Lecture Notes in Computer Science), vol. 8282. Burnaby, BC, Canada: Springer, 2013, pp. 29–47.
[83] J. W. Bos, M. Naehrig, and J. van de Pol, "Sieving for shortest vectors in ideal lattices: A practical perspective," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 880, 2014.

ZEDONG SUN is currently pursuing the Ph.D. degree in cyberspace security with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. His research interest includes algorithms for solving hard lattice problems.

CHUNXIANG GU is currently a Professor with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. His research interests include cryptography and network security.



YONGHUI ZHENG is currently an Associate Professor with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. His research interests include cryptography and information security.