Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.2, Apr. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.2.223

Gauss Sieve 반복 동작에서의 비효율성 개선*

천 병 호,[1†] 이 창 원,[1] 전 찬 호,[1] 홍 석 희,[2] 김 수 리[3‡]
[1,2]고려대학교 (대학원생, 교수), [3]성신여자대학교 (교수)

Improvement in Inefficient Repetition of Gauss Sieve*

Byeongho Cheon,[1†] Changwon Lee,[1] Chanho Jeon,[1] Seokhie Hong,[2] Suhri Kim[3‡]
[1,2]Korea University (Graduate student, Professor),
[3]Sungshin Women's University (Professor)

요 약

Gauss Sieve는 격자 기반 문제 중 하나인 SVP를 풀기 위한 알고리즘으로 지수 시간 및 공간 복잡도를 필요로 한다. 알고리즘의 종료 조건은 공간 복잡도와 관련이 있는 리스트의 크기 및 충돌 횟수에 의해 결정된다. 여기서 충돌이란 샘플링 된 벡터에 대한 축소 연산 뒤 이미 리스트에 존재하는 벡터와 동일한 벡터가 되는 상황을 의미하며 일정 횟수 이상의 충돌이 발생할 경우 알고리즘은 종료된다. 기존 알고리즘으로부터 제시된 공간 복잡도를 기준으로 실제 실행 결과를 확인하였을 때, 가장 짧은 벡터를 발견한 이후에도 불필요한 연산이 지속되는 것을 확인하였다. 이는 기존의 종료 조건이 필요 이상으로 크게 설정되었음을 의미한다. 따라서 본 논문에서는 불필요한 연산이 반복되는 지점을 파악한 뒤 기존에 필요로 하는 연산의 횟수에 대한 최적화를 진행한다. 종료 조건이 되는 충돌의 임계값과 샘플 벡터가 생성되는 분포를 조정하는 방식으로 실험을 진행하였으며 실험 결과 가장 큰 비중을 차지하는 축소 연산은 62.6% 감소하였으며 이에 따른 공간 및 시간 복잡도는 각각 4.3%, 1.6% 감소하였다.

ABSTRACT

Gauss Sieve is an algorithm for solving SVP and requires exponential time and space complexity. The termination condition of the Sieve is determined by the size of the constructed list and the number of collisions related to space complexity. The term 'collision' refers to the state in which the sampled vector is reduced to the vector that is already in the list. if collisions occur more than a certain number of times, the algorithm terminates.

When executing previous algorithms, we noticed that unnecessary operations continued even after the shortest vector was found. This means that the existing termination condition is set larger than necessary. In this paper, after identifying the point where unnecessary operations are repeated, optimization is performed on the number of operations required. The tests are conducted by adjusting the threshold of the collision that becomes the termination condition and the distribution in which the sample vector is generated. According to the experiments, the operation that occupies the largest proportion decreased by 62.6%. The space and time complexity also decreased by 4.3 and 1.6%, respectively.

Keywords: Gauss Sieve, Lattice, SVP, Vector Reduce, Sample Vector

Received(02. 20. 2023), Modified(03. 23. 2023), Accepted(03. 23. 2023)
* 이 논문은 2021년 정부(방위사업청)의 재원으로 국방과학연구소의 지원을 받아 수행된 연구임(UD210027XD).
† 주저자, cheonbh0819@gmail.com
‡ 교신저자, suhrikim@gmail.com(Corresponding author)

I. 서 론

1994년 Shor에 의해 제안된 양자 알고리즘에 의해 기존의 인수분해 문제의 어려움에 기반한 RSA, 이산대수 문제에 기반한 타원곡선 암호 등의 공개키 암호들이 다항 시간 내에 해결될 수 있다는 것이 증명되었다. 이에 NIST에서는 양자 컴퓨터 환경에서 안전한 새로운 공개키 암호 시스템에 대한 표준화 작업을 위해 2017년부터 공모를 시작하게 되었다. 여기에는 크게 다섯 가지 분야의 암호가 있는데, 본 논문에서는 그 중 격자 기반암호에 관한 내용을 다룬다. 격자 기반암호에서 다루는 문제는 대표적으로 SVP와 CVP가 존재한다. 격자 상에서 가장 짧은 벡터를 찾는 문제를 SVP (Shortest Vector Problem) 라 하고 격자 밖의 임의의 점에서 가장 가까운 격자 상의 벡터를 찾는 문제를 CVP (Closest Vector Problem) 라 한다. 본 논문에서는 SVP를 풀기 위한 알고리즘의 연구를 진행한다. 현재까지 다항 시간 내에 SVP를 풀 수 있는 알고리즘은 알려지지 않았으며 위 문제들은 다항 시간 안에 풀 수 없는 NP-Hard이다[1,2].

SVP를 풀기 위한 대표적인 알고리즘은 Sieve[3]와 Enumeration[4]이 있으며 본 논문에서는 Sieve 알고리즘 중 하나인 Gauss Sieve[5]에 대한 분석을 진행한다. 최초의 Sieve 알고리즘인 AKS Sieve[3]는 필요한 벡터를 한 번에 샘플링하기 때문에 높은 공간 복잡도를 요구하게 된다. List Sieve[5]에서는 기존 Sieve에서의 공간 복잡도 문제를 개선하기 위해 한 번에 하나의 벡터를 샘플링한다. 이를 통해 한 번에 많은 수의 벡터를 샘플링 해야 하는 부담을 줄여줄 수 있다. 각 단계에서 샘플링된 벡터들은 리스트(L) 에 존재하는 벡터를 이용해 크기를 줄이면서 리스트에 추가하는 방식으로 격자 내의 짧은 벡터를 찾는 방식으로 동작한다. 이러한 List Sieve의 휴리스틱한 변형 알고리즘이 Gauss Sieve[5]이며 본 논문의 이후의 장들에서는 Gauss Sieve의 동작 원리에 대해 살펴보고 Gauss Sieve의 결과 벡터 반환 시점에 대한 분석을 진행하며 이를 통해 기존의 알고리즘에 대한 개선된 형태의 알고리즘을 제안하고 이를 통해 해당 알고리즘이 개선될 수 있는 정도에 대해 살펴본다.

II. 배경 지식

본 장에서는 격자 기반 암호에서 기반 문제로 다루는 대표적 난제인 SVP, CVP에 대해 설명하고 본 논문에서 분석하고자 하는 SVP를 풀기 위한 Gauss Sieve 알고리즘이 갖는 복잡도에 대해 간단히 설명한다.

2.1 격자 암호 기반 문제

격자 기반 암호는 격자 상에서의 문제에 대해 해당 문제를 풀기 어려운 정도에 기반하여 암호 알고리즘의 안전성을 확보할 수 있게 된다. 여기에 사용되는 난제들은 대표적으로 격자 상의 가장 짧은 벡터를 찾는 문제인 SVP (Shortest Vector Problem)와 격자 위가 아닌 곳에 존재하는 임의의 점에서 가장 가까운 격자 위의 벡터를 찾는 문제인 CVP (Closest Vector Problem) 가 있다[1, 2].

2.2 최단 벡터 길이 범위

격자를 형성하는 기저 행렬은 m차원의 선형 독립인 n개의 벡터로 구성된다. n개의 벡터로부터 생성되는 기저 행렬을 B라하고 일반적으로 $m=n$인 기저 행렬을 다루며 이를 full-rank 혹은 full-dimensional 행렬이라고 한다.

B에 의해 생성되는 격자를 $\mathcal{L}(B)$라고 나타내며 해당 격자 상의 가장 짧은 벡터의 크기는 $\lambda_1(B)$ 또는 $\lambda_1(\mathcal{L}(B))$라고 정의한다. SVP를 풀기 위한 알고리즘의 목표는 입력받은 격자 상에 존재하는 가장 짧은 길이의 벡터를 찾는 것이다. 이때 우리는 가우시안 휴리스틱을 이용해 주어진 격자 $\mathcal{L}$에서의 λ_1의 상한을 결정할 수 있다.

정의 1. [가우시안 휴리스틱 (Gaussian Heuristic)][6]

임의의 격자 $\mathcal{L}$에 가우시안 휴리스틱 $GH(\mathcal{L})$은 가장 짧은 벡터의 크기에 대한 예상되는 값을 추정하며 이는 다음의 식과 같이 계산한다.

$$GH(\mathcal{L}) = \sqrt{\frac{n}{2\pi e}} \cdot vol(\mathcal{L})^{1/n} \quad (1)$$

SVP를 풀 때 $GH(\mathcal{L})$는 일반적으로 격자에서의

가장 짧은 벡터의 예상되는 크기를 의미하기 때문에 문제를 해결하기 위한 중요한 단서로 이용된다. 즉, 가우시안 휴리스틱으로부터 λ_1에 대한 근삿값을 얻을 수 있으며 이를 이용하는 가장 대표적인 예로, SVP-Challenge는 λ_1의 목표 크기를 1.05 $GH(\mathcal{L})$로 설정한다[7,8].

2.3 구면 접촉 수(kissing number)

구면 접촉수란 정해진 차원에서 중심구와 서로 겹쳐지지 않고 접할 수 있는 동일한 크기의 단위 구(unit sphere)의 최대 개수를 의미하며 동일한 차원에 대해서도 단위 구의 배치 방법에 따라 그 수는 조금씩 달라질 수 있다.

n차원에서의 구면 접촉 수는 τ_n이라 하며 이를 통해 우리는 주어진 n차원에서 필요로 하는 벡터의 수에 대한 상한을 예상할 수 있다. Sieve 알고리즘에서는 τ_n에 대한 상한으로부터 특정 조건에서의 알고리즘에서 필요로 하는 벡터의 개수를 알 수 있고 이는 곧 알고리즘의 공간 복잡도와 직접적인 연관이 있다. 여기에서 Kabaitiansky와 Levinshtein에 의해 주어진 공간에서 존재하는 벡터의 수의 상한은 다음과 같이 정의된다[9].

정리 1. [Kabatiansky and Levenshtein][9]

$A(n,\phi)$를 $\mathbb{R}^n$위에서의 점들로 구성된 어떤 집합 C의 최대 크기라 하며 집합 C의 임의의 두 벡터 v_i, v_j $(v_i \neq v_j)$가 이루는 각도를 ϕ (여기서는 ϕ_{v_i,v_j}라고 한다)라 한다. 이때 $0 < \phi < 63^\circ$을 만족하는 경우 아래의 부등식을 만족하는 c에 대해 $A(n,\phi) = 2^{cn}$를 만족한다.

$$c \leq -\frac{1}{2}log(1-\cos(\phi)) - 0.099 \quad (2)$$

(증명) 식 (1)에서 c의 값은 $\frac{1}{n}log_2 A(n,\phi)$와 같기 때문에[9] 다음과 같이 식을 정리할 수 있다.

$$\begin{aligned} c &= \frac{1}{n}log_2 A(n,\phi) \\ cn &= \log_2 A(n,\phi) \\ 2^{cn} &= A(n,\phi) \end{aligned} \quad (3)$$

위와 같이 n차원에서의 집합 C 크기의 최댓값을 계산할 수 있다. □

위 정리에 의해 Gauss Sieve의 n차원에서의 리스트 L의 크기의 상한을 계산할 수 있다.

결과적으로 Gauss Sieve는 n차원 격자에 대한 SVP를 푸는 과정에서 리스트 내에 대략 2^{cn}개의 벡터가 존재할 때, 리스트에는 가장 짧은 벡터가 존재하고 있음을 의미한다[5]. 시간 복잡도는 임의의 기저와 격자 상의 벡터를 샘플링하는 확률적 샘플 알고리즘으로 인해 이론적 증명은 불가능하다. 때문에 시간 복잡도는 리스트 내의 임의의 두 벡터 v_1, v_2에 대해 각 벡터 쌍이 서로가 줄일 수 있는지를 확인하기 위해 적어도 한 번은 비교를 수행하는 상황을 기준으로 공간복잡도의 제곱에 비례한다. 이렇게 계산된 시간 및 공간 복잡도는 각각 $2^{0.21n}$, $2^{0.48n}$이다[5].

III. 알고리즘 분석

Micciancio와 Voulgaris에 의해 제안된 Gauss Sieve는 반복적으로 격자 벡터를 리스트에 추가하며 점점 더 긴 리스트를 구성하게 되며 리스트에 가장 짧은 벡터가 포함될 때까지 진행된다. 샘플 벡터의 경우 샘플링 알고리즘을 사용하여 격자 상의 이산 가우시안 분포로부터 샘플링되거나 스택에 벡터가 존재하는 경우 스택 내의 벡터를 꺼내 사용한다. 만약 리스트 벡터들이 수정되거나 새로운 샘플 벡터가 축소된다면 해당 벡터들은 스택으로 보내지게 된다. 이런 식으로 크기는 점진적으로는 증가한다.

Gauss Sieve에서 리스트 내의 임의의 벡터는 다음을 만족하도록 진행된다.

Reduce v_1 with v_2 : if $\| v_1 \pm v_2 \| < \| v_1 \|$ then $v_1 \leftarrow v_1 \pm v_2$

알고리즘의 실행을 통해 리스트는 항상 pairwise-reduce를 만족하는 상태가 된다.

$$\| v_1 \pm v_2 \| \geq \max\{\| v_1 \|, \| v_2 \|\} \quad \text{for all } v_1, v_2 \in L$$

위 식으로부터 리스트 내의 임의의 두 벡터 v_1, v_2는 적어도 60°이상의 사잇각을 갖게 된다는 것을

내포하며[9] 그렇지 않은 경우, 두 벡터 중 하나는 나머지 한 벡터로 축소되어 리스트에 추가된다. 즉, 임의의 두 벡터 간의 각도가 60° 이상인 $\mathbb{R}^n$에서의 벡터의 최대 개수는 구면 접촉 수에 의해 계산될 수 있으며 이를 통해 n차원에서 Gauss Sieve에서 가장 짧은 벡터가 리스트에 포함되는 경우의 리스트의 크기는 $2^{0.21n+o(n)}$을 초과하지 않음을 의미한다[9]. 이와 관련한 내용은 2.3에 언급되어있다.

3.1 내부 연산

Gauss Sieve의 전체 과정은 Fig. 1.를 통해 확인할 수 있으며 알고리즘의 내부에서 동작하는 주요 연산은 크게 두 부분으로 나누어서 볼 수 있다.

샘플링. 짧은 벡터를 만들기 위해 샘플링 알고리즘을 이용해 새로운 샘플 벡터를 생성하거나 기존에 가진 스택 저장공간으로부터 벡터를 불러오는 연산.

벡터 축소. 샘플링 과정에서 생성된 샘플 벡터들에 대해 기존에 리스트에 저장된 벡터를 이용해 그 크기를 줄여나가며 보다 짧은 벡터를 만들기 위한 연산 샘플링 과정으로부터 얻게 되는 샘플 벡터들은 리스트 벡터와의 축소 과정을 통해 기존의 값과 달라지거나 혹은 달라지지 않고 유지될 수 있다. Gauss Sieve에서는 축소 이후의 결과 벡터가 리스트에 이미 존재하는 벡터와 동일한 벡터가 되는 경우를 충돌이라고 정의한다. 충돌이 발생하면 해당 시점에서 반복을 종료하고 다음 반복을 진행하기 위해 새로운 벡터를 샘플링하는 단계로 돌아간다. 충돌이 발생하지 않았지만 벡터의 값이 기존과 달라진 경우 Pairwise-reduce를 만족하도록 리스트 벡터 축소를 위해 사용되며 이를 위해 임시 저장공간으로 보내진다. 축소 과정 이후에 값이 달라지지 않을 때는 새로운 리스트 벡터로 추가되며 이러한 일련의 흐름을 한 번의 반복이라고 한다.

Algorithm 1. Gauss Sieve

Require: $B = \{ b_1, b_2, \ldots, b_n \}, \alpha, \beta > 0 \in \mathbb{R}$

1. $L \leftarrow \{ \}, S \leftarrow \{ \}, K \leftarrow 0$
2. while $K < \alpha|L| + \beta$ do
3. if $S \neq \{ RIGHT$ then
4. Pop from Stack S to v
5. else
6. Generate a new vector v from sampler
7. $(v', L, S) \leftarrow$ Gauss_Reduce(v, L, S)
8. if $\| v' \| = 0$ then
9. $K \leftarrow K + 1$
10. else
11. $L \leftarrow L \cup \{ v' \}$
12. Return a shortest vector in L

Fig. 1. Gauss Sieve

Gauss Sieve에서 공간 복잡도는 리스트의 최대 크기에 의해 결정되며 리스트를 생성하는 과정에서 발생하는 벡터 축소 연산은 각 반복에서의 연산량 대부분을 차지한다. 따라서 본 알고리즘의 벡터 축소 연산에 대한 최적화와 종료에 필요한 조건인 충돌수를 기존보다 줄이는 방식으로 최적화를 진행함으로써 알고리즘 동작에서의 필요 비용을 줄이는 것을 목적으로 한다. 본 논문에서는 전체 반복에서 사용되는 축소 연산의 수를 줄이는 방향으로 연구를 진행하며 실험을 진행한다. 이 과정에서 필요 이상으로 지속해서 수행되는 연산에 대한 문제를 파악하고 이를 통해 발생하는 불필요한 연산 비용을 해소하는 방향으로 기존에 알고리즘이 동작하는 방식보다 효율적인 방법을 찾는다. 이로 인해 기존과 동일한 수준의 결과를 보다 적은 비용을 통해 얻는 것을 목적으로 한다. 이를 위해 격자 상에서의 알고리즘을 수행하였을 때 가장 짧은 벡터를 발견하는 시점이 언제인지를 확인할 필요가 있다. 이를 확인한 뒤 해당 벡터가 발견된 이후의 연산들은 모두 불필요한 연산으로 분류하며, 해당 시점을 미리 파악하여 이후의 연산을 수행하지 않도록 알고리즘을 수정한다. 이로 인해 불필요한 시간 및 공간을 사용하지 않게 됨으로써 알고리즘 수행에 필요한 전반적인 비용을 줄일 수 있게 된다.

위 가정을 바탕으로 임의의 랜덤하게 선택된 5개의 기저에 대해 각 차원에 대한 실험을 진행하였다. 실험에서 가장 짧은 벡터를 발견하는 지점은 기존 알고리즘의 종료되어 결과 벡터를 반환하는 시점을 100%라고 했을 때를 기준으로 가장 짧은 벡터가 발견된 지점을 의미한다(Table 1 참조).

실험은 40~70차원을 대상으로 진행하였으며 해당 차원들에서 가장 짧은 벡터가 발견되는 시점은 Table 1의 결과를 통해 확인할 수 있다. 실험에 사용된 기저에 대한 추가적인 설명은 3.3에서 확인할 수 있으며 본 실험 결과로부터 최소 10%에서 최대 40% 가까이 불필요한 연산이 반복되고 있음을 알 수 있다. 또한 평균 82% 진행되었을 때 이미 가장

Table 1. SV(Shortest Vector) Discovery Point on Dimension from 40 to 70

Dimension	SV Discovery Point(%)	Dimension	SV Discovery Point(%)
40	69	**56**	80
41	85	**57**	82
42	62	**58**	85.5
43	71.5	**59**	89.5
44	57	**60**	83.25
45	82	**61**	84
46	74	**62**	83.4
47	77	**63**	88.67
48	78.25	**64**	83.6
49	85.25	**65**	86
50	87.67	**66**	91.75
51	90.33	**67**	89.75
52	84	**68**	86.2
53	87.67	**69**	90.8
54	83.67	**70**	87.2
55	88	Average	82.35

짧은 벡터를 발견함을 확인할 수 있으며 이후의 진행될 실험(4장)에서는 해당 평균 벡터 발견 시점 비율을 이용한다.

3.2 종료 조건

Gauss Sieve는 일정 수 이상의 충돌이 발생할 때 알고리즘이 종료되는 방식을 이용한다. 종료되는 조건의 범위는 해당 반복 시점에서 동적으로 변하는 리스트의 크기에 따라 결정되며 발생하는 충돌 횟수의 누적값에 따라 해당 값이 아래의 식과 같이 설정한 임곗값을 초과하면 종료되는 방식으로 구성되어 있다.

$$K < \alpha |L| + \beta \tag{4}$$

K는 알고리즘 실행 과정에서 발생하는 충돌 횟수의 누적값을 의미하며, α와 β는 각각 0.1, 200으로 설정되어 사용된다[5]. 이를 통해 각 단계에서 충돌 누적값이 리스트 크기에 의해 설정되는 종료 조건의 임곗값을 초과하는 경우 알고리즘이 종료되며 해당 시점에서 반환되는 결과 벡터를 가장 짧은 벡터 λ_1로 판단하게 된다.

3.3 격자 기저 및 차원 선정 방법

본 논문에서 사용하는 랜덤한 격자의 기저는 NTL 라이브러리 내에 존재하는 의사 난수 생성기를 이용해 생성되는 격자의 기저를 의미한다[12]. Goldstein과 Mayer는 임의로 선택된 충분히 큰 소수 p와 $\{ 0, ..., p-1 \}$의 수 중 독립적이고 균일하게 무작위로 선택한 x_i에 의해 격자 기저가 생성될 수 있음을 보이며 이와 같은 기저를 본 논문에서는 랜덤한 격자라고 한다[14]. 이렇게 생성된 기저의 형태는 다음과 같이 나타낼 수 있다.

$$\begin{pmatrix} p & & \\ x_1 & 1 & \\ \vdots & & \ddots \\ x_{n-1} & & & 1 \end{pmatrix}$$

위와 같이 생성된 격자의 가장 짧은 벡터의 크기는 식 (1) 에서의 식을 이용해 근삿값을 추정할 수 있고, 해당 기저는 추가로 블록 크기가 20으로 설정된 격자 축소 기저 알고리즘인 BKZ 알고리즘[4]을 이용해 격자 기저 축소 후 Sieve 과정을 적용한다.

실험 대상 차원은 40~70차원으로, Gauss Sieve 알고리즘이 제안된 논문에서 진행된 실험과 최대한 동일한 차원에서 실험을 진행하기 위함이다. 실제로 [5]에서는 30차원부터 62차원까지 진행하였으며 우리는 [5]의 실험 결과를 기준으로 비교를 진행한다. 다만 40차원부터 실험을 진행하는 이유는 SVP-Challenge의 정보를 이용해 기존의 알고리즘과 수정된 형태의 알고리즘으로 발견한 가장 짧은 벡터에 대한 검증을 진행하기 때문이다. SVP-Challenge는 가장 짧은 벡터에 대한 실험 결과를 40차원부터 제공하기 때문에 기존의 실험들에서 사용되었던 30~39차원에 대한 실험은 진행하지 않고 40차원에서부터 실험을 진행한다. 대신 기존의 알고리즘에서 대략 30개의 차원 범위에서의 실험을 통한 추이를 확인하였기 때문에 60~70차원에 대한 실험을 추가함으로써 총 40~70차원을 실험의 대상으로 설정하였다.

3.4 Gauss Sieve 공간 복잡도

Gauss Sieve의 복잡도를 실험을 통해 직접 구한 값은 2.3에서 언급된, 즉 [5]의 이론적인 수치와는 상이한 값을 보인다. 본 논문에서는 Gauss Sieve에서의 복잡도를 리스트 크기가 가장 큰 경우에 대해 계산하였으며 이는 다음과 같이 계산된다.

$$\log_2(\max(|L|)) \tag{5}$$

리스트의 크기는 줄어들거나 늘어나는 과정을 반복하기 때문에, 전체 과정 중 가장 많은 벡터를 저장하고 있는 순간의 리스트의 크기를 기준으로 복잡도를 계산하였으며 이때의 공간 복잡도는 $2^{0.24n}$이다. 따라서 이후에 계산되는 복잡도에 대한 증감률은 해당 복잡도를 이용한다.

IV. 제안 기법 및 실험

본 장에서는 실제 알고리즘에 대해 종료 임곗값을 조정하는 방식과 샘플 벡터를 얻게 되는 샘플링 분포를 조정하는 방식을 결합하여 기존 알고리즘을 수정하고 해당 방식으로부터 기존 알고리즘에서의 연산 대비 감소율을 통해 실제 개선 효과가 있는지를 확인한다.

4.1 종료 임계 값 조정

3장에서의 실험을 통해 실험 대상 차원들에 대해 평균 82% 지점에서 이미 가장 짧은 벡터가 발견되는 것을 확인하였다. 여기에서는 종료 임곗값과 가장 짧은 벡터가 발견되는 평균 지점을 이용하여 새롭게 종료 임곗값의 범위를 설정한다. 이 과정에서 총 두 가지 조건을 통해 우리는 기존보다 더 이른 시점에 종료를 할 수 있게 만들어준다.

첫 번째로, 기존에 종료 조건으로 사용되었던 임곗값인 식 (4)에 우리가 구한 비율을 곱하여, 종료 임곗값에 대한 비율을 조정하는 것으로 다음과 같다.

$$K < (\alpha|L|+\beta)*(rate) \tag{6}$$

위 방법을 이용하여 기본적으로 본 알고리즘에서 종료하는 데 필요로 하는 충돌수를 줄여줄 수 있고, 이 과정에서 기존보다 종료 시점을 앞당기게 됨에 따라 리스트의 크기를 해당 비율에 비례하여 줄어들 것을 예상할 수 있고 다음과 같이 나타낼 수 있다.

$$\log_2(|L|)*(rate)) \approx 0.235 \tag{7}$$

이는 줄어든 충돌 범위에 비례하여 리스트의 크기가 줄어든 상황을 가정했을 때의 복잡도를 의미한다. Gauss Sieve의 경우 알고리즘이 동작하면서 가장 짧은 벡터를 발견한 시점 이후에는 더는 벡터의 크기가 변경되지 않은 상태로 임곗값에 도달할 때까지 충돌을 만들어낸다. 이 경우는 알고리즘 동작 과정에서 가장 짧은 벡터의 크기가 변경되는 횟수를 이용해 종료될지를 결정하기 위한 새로운 조건을 추가해준다. 해당 조건을 통해 벡터의 크기가 변경된 횟수가 많아질수록 해당 시점에서 필요한 임곗값을 더 줄여나가게 된다. 이에 알고리즘은 기존에 동작하던 것보다 더 적은 반복 횟수와 적은 연산만으로 기존과 같은 길이를 갖는 결과 벡터를 반환하며 종료될 수 있게 된다.

이를 알고리즘으로 나타내면 Fig. 2.와 같으며, 이때 알고리즘에서의 space upperbound는 2.3에서 언급된 Gauss Sieve가 가질 수 있는 공간 복잡

Algorithm 2. Modified Gauss Sieve

Require: $B = \{ b_1, b_2, \dots, b_n \}, \alpha, \beta > 0 \in \mathbb{R}$

```
1.  L ← { }, S ← { }, K ← 0, norm_change_count ← 0
2.  while K < (α|L|+β)*(rate) do
3.    if S ≠ { RIGHT then
4.      Pop from Stack S to v
5.    else
6.      Generate a new vector v from sampler
7.    (v', L, S) ← Gauss_Reduce(v, L, S)
8.    if ‖v'‖ = 0 then
9.      K ← K + 1
10.   else
11.     L ← L ∪ { v' }, norm_change_count++
12.   if norm_change_count ≥ 1 and |K| ≥ (space upperbound*(rate)) / (norm_change_count + 1)
13.     break
14. Return a shortest vector in L
```

Fig. 2. Modified Gauss Sieve

Table 2. Reduce Rate of Reduce Operations by Adjusting Termination Threshold Parameters on Each Dimension and Average

Dimension	Before	After	Reduce Rate(%)
40	55153	19716	64.25
45	129115	46310	64.13
50	287526	106250	63.05
55	657913	248971	62.16
60	1549568	582028	62.44
65	3372301	1276727	62.14
70	6295474	2297588	62.92
Average	-	-	62.60

도의 상한인 2^{cn}을 의미한다. 이를 통해 알고리즘 2가 우리가 목표로 설정한 공간에 대한 상한을 초과하지 않도록 하면서 위 방식을 통해 실험을 진행한 결과는 Table 2에서 확인할 수 있다. 본 실험을 통해 기존 알고리즘 대비 축소 연산이 62.6% 감소한 것을 확인할 수 있다.

4.2 샘플 벡터 분포 범위 조정

Gauss Sieve는 격자 상의 벡터를 무작위로 샘플링하는 과정에서 Klein 샘플 알고리즘을 이용한다[10].

$$x \xleftarrow{\$} [c-s \cdot t, c+s \cdot t] \tag{8}$$

벡터가 생성되는 샘플링 분포를 조정함으로써 우리는 기존보다 작은 크기를 갖는 벡터를 샘플링 할 수 있게 됨을 기대할 수 있다. 그렇게 된다면 더 작은 벡터를 이용해 우리가 목표로 하는 짧은 벡터를 기존보다 더 적은 시간을 이용해 발견하게 될 수 있다.

Table 3에서는 샘플링 매개변수를 변경하여 분포를 조정하고 이에 따른 전후 값의 변화를 확인할 수 있다. 변경되는 매개변수의 값은 [11]의 결과를 기반으로 평균 샘플 벡터가 평균적으로 가장 작게 형성되는 값으로 조정되었으며 구체적으로는 위 x가 결정되는 분포상의 t의 값을 $t/70$로 조정하게 된다. 즉, 기존의 분포보다 더 좁은 범위에서 벡터를 샘플링하게 되며 이러한 방식으로 샘플링되는 벡터의 분포가 좁아지게 되는 경우 평균 샘플 벡터의 크기는 43.22% 감소하는 것을 확인할 수 있다.

Table 3. Vector Norm Reduce Rate by Adjusting Sampling Distribution Parameters on Each Dimension and Average

Dimension	Before	After	Reduce Rate(%)
40	14983	9959	50.46
45	19231	12887	47.89
50	23701	16150	46.76
55	31105	21169	46.94
60	38846	27629	40.60
65	48950	36928	32.55
70	64932	49387	31.48
Average	-	-	43.22

위 결과로부터 샘플링 과정에서 생성되는 벡터의 평균 크기가 작아지는 것을 확인하였으며 작은 벡터를 뽑는 빈도가 높아짐에 따라 우리는 기존보다 더 적은 자원을 이용해 최단 벡터를 찾을 수 있음을 예상할 수 있다.

4.3 실험 및 결과

4.1과 4.2에서 언급된 종료 임곗값과 샘플링 분포를 조절하는 방식을 이용한 수정된 알고리즘과 기존 알고리즘의 연산량 및 복잡도에 대한 실험 결과는 다음과 같다(Table 4 참조). Table 4에 따르면 전체 반복 횟수는 11.43%, 벡터 축소 연산은 62.6% 감소하였고, 감소한 반복 횟수와 벡터 축소 연산에 따른 공간 복잡도는 4.3%, 시간 복잡도는 1.6% 감소하였다. 추가로, 우리는 1장에서 언급되었던 가우시안 휴리스틱을 이용해 주어진 기저에 대한 대략적인 최단 길이 벡터의 크기를 추정할 수 있다. 이를 바탕으로 Table 5 에서는 가우시안 휴리스틱으로 추정한 최단 길이 벡터의 크기와 SVP-Challenge에서의 결과, 그리고 마지막으로 본 장에서 수정된 형태의 알고리즘을 이용해 진행한 실험 결과로부터 얻은 벡터의 크기를 비교하여 그 차이를 확인할 수 있다.

Table 4. Iterations, Reduce Operations, Space Complexity, Time Complexity Reduce Rate on Each Dimension and Average

Dimension	Iterations Reduce Rate(%)	Reduce Operations Reduce Rate(%)	Space Reduce Rate(%)	Time Reduce Rate(%)
40	14.03	64.25	13.74	3.07
45	14.83	64.13	9.66	2.55
50	11.59	63.05	5.11	1.6
55	10.92	62.16	6.4	1.13
60	9.93	62.44	6.4	0.45
65	10.74	62.14	15	0.86
70	10.55	62.92	15.63	0.42
Average	11.43	62.60	9.42	1.59

Table 5. Comparing Vector Norm on Each Dimension

Dimension	Gaussian Heuristic	SVP Challenge	Modified Gauss Sieve
40	1556.93	1634.78	1679.65
45	1645.84	1728.14	1732.25
50	1740.62	1827.65	1827.65
55	1825.52	1916.8	1920.08
60	1913.48	2009.15	1964.98
65	1983.13	2082.29	2099.84
70	2064.2	2167.41	2149.71

V. 결 론

본 논문에서는 Gauss Sieve에 대해 실제 동작 과정에서 발생하는 비효율적인 부분을 확인하였다. 주어진 기저들에 대해 가장 짧은 벡터를 발견되는 평균 지점을 확인하였고, 이를 기준으로 종료 조건에 대한 임곗값 범위를 조정하고 동작 과정 내의 추가 종료 조건을 설정하였다. 또한 샘플링 과정에서 생성되는 샘플 벡터의 평균 크기를 줄이기 위해 샘플링 분포 조정을 통해 기존보다 이른 종료를 목표로 실험을 진행하였다. 결과적으로 이른 종료를 하게 됨으로써 기존에 사용되던 연산량의 감소와 더불어 전체적인 반복 횟수의 감소 및 사용되는 리스트 크기의 감소를 통해 전반적인 공간 복잡도의 감소를 기대해 볼 수 있다. 실험 결과 전체 반복 횟수는 11.43%, 벡터 축소 연산은 62.6% 감소한 것을 확인할 수 있었다. 감소한 반복 횟수와 벡터 축소 연산에 의해 기존 알고리즘 복잡도 대비 공간 복잡도는 4.3%, 시간 복잡도는 1.6% 감소한 것을 Table 5를 통해 확인할 수 있다.

결과적으로 수정된 알고리즘의 결과 벡터는 기존 알고리즘을 통해 얻게 되는 벡터와 동일한 벡터이기 때문에 실제로 동일한 결과를 출력하는 과정에서 중간 과정에 필요한 연산량이 감소한 것과 더불어 복잡도의 감소로 이어질 수 있었기 때문에 유의미한 결과로 볼 수 있다. 추가로, 감소한 축소 연산량과 비교해 실제 감소한 복잡도의 비율은 높지 않은데, 이는 수정된 알고리즘을 통해 감소하는 연산 대부분은 리스트에 새로운 벡터를 추가할 때가 아닌 충돌이 발생하는 벡터에 대한 것으로 추측된다. 따라서 실제 감소한 벡터 축소 연산의 경우는 복잡도에 직접적으로 미치는 영향은 크지 않다고 볼 수 있다. 또한 본 논문에서의 실험에 사용되지 않은 차원에서는 본 논문에서 보인 감소 수치와는 다소 차이가 존재할 수 있다. 이러한 이유로는, 입력되는 기저와 샘플링되는 벡터에 따라 알고리즘이 갖는 성능은 실제 복잡도와는 달라질 수 있다. 다만 본 논문에서의 요지는 기존의 알고리즘 종료 조건은 필요한 공간 이상을 사용하도록 크게 설정되어있는 것을 확인하는 것이었으며 이를 개선하기 위한 비율을 추정하기 위해 3.1에서의 실험을 진행하였다. 추가로 샘플링 과정에서 샘플러의 분포와 관련된 매개변수를 조정하여 샘플 벡터에 대한 최초 크기를 줄이는 과정 또한 함께 수행하였다. 이를 통해 불필요한 벡터 축소 과정을 해소함과 동시에 벡터가 저장되는 리스트의 크기를 줄임으로써 기존에 동작하던 알고리즘 대비 적은 비용으로 격자 상의 가장 짧은 벡터를 찾는 것이다. Table 6의 결과로부터 수정된 알고리즘에서는 $1.052GH$를 만족하는 크기를 갖는 벡터를 찾았으며 이는 SVP-Challenge에서의 목표치인 $1.05GH$에 매우

Table 6. Comparing Space and Time Complexity of Original Gauss Sieve and Modified Gauss Sieve Algorithm on Each Dimension ($\log_2 | L |$,$\log_2 | Time_{/\mu s} |$)

Dimension	Space before	Space after	Time before	Time after
40	0.251	0.246	0.431	0.422
45	0.246	0.241	0.438	0.430
50	0.241	0.236	0.441	0.436
55	0.238	0.236	0.445	0.442
60	0.235	0.230	0.452	0.449
65	0.233	0.228	0.468	0.465
70	0.231	0.226	0.478	0.476
Average	0.24	0.234	0.45	0.445

근사한 수치라는 점에서 충분히 유의미한 결과라고 판단하였다.

본 논문의 목적은 격자의 특성을 고려하지 않은 상황에서 연산 과정에 불필요하게 발생하는 병목 현상을 개선하는 것이다. 그러므로 알고리즘의 복잡도를 보다 개선하기 위해서는 축소 연산 수의 감소를 통한 방식보다는 축소 연산의 형태 자체에 대한 개선과 더불어 주어진 격자의 특성에 따라 적용될 알고리즘이 변경될 필요가 있다. 따라서 이후의 연구에서는 앞서 언급된 본 논문에서의 한계인 복잡도 도출에 대한 이론적인 접근과 이를 해결하기 위한 연구를 진행하고자 한다. 구체적으로는 단위 축소 연산 자체의 개선을 통해 복잡도를 줄이는 방안과 더불어 아이디얼 격자, 순환 격자와 같은 특수한 형태의 격자[13]에 대해, 그리고 더 확장된 추가적인 차원들에서도 적용할 수 있는 일반화된 특성을 찾고 이에 대한 이론적인 분석과 증명을 통한 후속 연구를 진행할 예정이다.

References

[1] M. Ajtai, "The shortest vector problem in L_2 is NP-Hard for randomized reduction," Proceedings of the 30th annual ACM symposium on Theory of computing, pp. 10-19, May. 1998.

[2] D. Micciancio and S. Goldwasser, Complexity of lattice problems:A cryptographic perspective, 1st Ed., pp. 45-90, Mar. 2002.

[3] M. Ajtai, R. Kumar and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," Proceedings of the 33rd annual ACM symposium on Theory of computing, pp. 601-610, Jul. 2001.

[4] C.P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," Mathematical programming 66, pp. 181-199, Aug. 1994.

[5] D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," ACM-SIAM symposium on Discrete Algorithms, pp. 1468-1480, Jan. 2010.

[6] P.Q. Nguyen, "Hermitian constant and lattice reductions," The LLL algorithm, Springer, pp. 19-70, Jan. 2009.

[7] SVP CHALLENGE, "svp challenge," https://www.latticechanllenge.org/svp-challenge, 2023.

[8] S. Zedong, G. Chunxiang and Z. Yonghui, "A review of sieve algorithms in solving the shortest lattice vector problem," IEEE Access, vol. 8, Oct. 2020.

[9] J.H. Conway and N.J.A. Sloane, "Sphere Packings, Lattices and Groups," Springer, pp. 21-30, Sep. 1999.

[10] P. Klein, "Finding the closest lattice vector when it's unusually close," Proceeding of the 11st annual ACM-SIAM symposium on Discrete algorithm, pp. 937-941, Feb. 2000.

[11] T. Ishiguro, S. Kiyomoto, Y. Miyake and T. Takagi, "Parallel Gauss Sieve Algorithm: Solving the SVP Challenge over a 128-Dimensional Ideal Lattice," Public-Key Cryptography-PKC 2014:

17th International Conference on Practice and Theory in Public-Key Cryptography, pp. 411-428, Mar. 2014.

[12] NTL: A library for doing number theory, "ntl library," https://libntl.org, 2023.

[13] M. Schneider, "Sieving for Shortest Vectors in Ideal Lattices," Progress in Cryptology-AFRICACRYPT 2013: 6th International Conference on Cryptology, Proceedings 6, pp. 375-391, Jun. 2013.

[14] J. van de Pol, N.P. Smart, "Estimating key sizes for high dimensional lattice-based systems," Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Proceedings 14, pp. 290-303, Dec. 2013.

〈저 자 소 개〉

천 병 호 (Byeongho Cheon) 학생회원
2021년 8월: 호서대학교 컴퓨터정보공학부 졸업
2021년 9월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 공개키 암호, 후양자암호

이 창 원 (Changwon Lee) 학생회원
2021년 2월: 서울시립대학교 수학과 졸업
2023년 2월: 고려대학교 정보보호대학원 공학석사
〈관심분야〉 후양자암호

전 찬 호 (Chanho Jeon) 학생회원
2019년 2월: 고려대학교 수학과 졸업
2019년 3월~현재: 고려대학교 정보보호대학원 석박사 통합과정
〈관심분야〉 후양자암호

홍 석 희 (Seokhie Hong) 종신 회원
1995년: 고려대학교 수학과 학사
1997년: 고려대학교 수학과 석사
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주)시큐리티 테크놀로지 선임연구원
2003년 3월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U. Leuven ESAT/SCD-COSIC 박사후연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키 및 공개키 암호 알고리즘, 부채널 공격 및 대응기법, 디지털포렌식

김 수 리 (Suhri Kim) 정회원
2014년 2월: 고려대학교 수학과 이학사
2016년 8월: 고려대학교 정보보호학과 공학석사
2020년 2월: 고려대학교 정보보호대학원 공학박사
2020년 3월~2021년 2월: 고려대학교 정보보호대학원 박사후연구원
2020년 3월~2021년 2월: K.U. Leuven ESAT/SCD-COSIC 박사후연구원
2022년 3월~현재: 성신여자대학교 수리통계데이터사이언스학부 조교수
〈관심분야〉 공개키 암호시스템, 후양자암호