entropy

MDPI

Article

Solving HNP with One Bit Leakage: An Asymmetric Lattice Sieving Algorithm

Wenhao Shi [1,2,*], Haodong Jiang [1,2] and Zhi Ma [1,2]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China
[2] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China
* Correspondence: swh3006@163.com

Abstract: The Hidden Number Problem (HNP) was introduced by Boneh and Venkastesan to analyze the bit-security of the Diffie–Hellman key exchange scheme. It is often used to mount a side-channel attack on (EC)DSA. The hardness of HNP is mainly determined by the number of nonce leakage bits and the size of the modulus. With the development of lattice reduction algorithms and lattice sieving, the range of practically vulnerable parameters are extended further. However, 1-bit leakage is still believed to be challenging for lattice attacks. In this paper, we proposed an asymmetric lattice sieving algorithm that can solve HNP with 1-bit leakage. The algorithm is composed of a BKZ pre-processing and a sieving step. The novel part of our lattice sieving algorithm is that the lattice used in these two steps have different dimensions. In particular, in the BKZ step we use more samples to derive a better lattice basis, while we just use truncated lattice basis for the lattice sieving step. To verify our algorithm, we use it to solve HNP with 1-bit leakage and 116-bit modulus.

Keywords: HNP; BKZ reduction; sieving; side-channel attack; ECDSA

Citation: Shi, W.; Jiang, H.; Ma, Z. Solving HNP with One Bit Leakage: An Asymmetric Lattice Sieving Algorithm. *Entropy* **2023**, *25*, 49. https://doi.org/10.3390/e25010049

Academic Editor: Ivan B. Djordjevic

Received: 1 October 2022
Revised: 18 December 2022
Accepted: 22 December 2022
Published: 27 December 2022

1. Introduction

A lattice is a discrete subgroup of $\mathbb{R}^m$, and is usually presented by a basis. There are infinitely many basis for a non-trivial lattice and we are usually interested in a basis with a short norm and that is orthogonal to other basis, which we call a good basis. Lattice reduction algorithms are designed to find high quality lattice basis, such as LLL reduction and BKZ reduction. The LLL reduction algorithm can be performed in polynomial time and outputs a LLL-reduced basis, which will be shorter and more orthogonal than the original basis. If you want a better lattice basis, then a stronger lattice reduction should be performed, which is what the BKZ reduction algorithm does. The BKZ algorithm is a generalization of the LLL algorithm with a higher block size that can output a much better lattice basis than LLL, and with costs that are exponential with time. With a good basis, we can solve hard problems in lattice with less effort, such as finding the short(est) vector in a lattice or the closest vector to a given target, called SVP and CVP, which are two hard problems in lattice. SVP asks to find the non-zero shortest vector in a given lattice, while CVP asks to find the closest lattice vector to a given target, in a given lattice. To find the short(est) vector in a lattice, there are currently four main methods we can use: enumeration [1–3], sieving [4–8], Voronoi cell [9], and Gaussian sampling [10]. Enumeration costs are exponential (of the dimension) in time but polynomial in memory. The best enumeration costs $2^{0.029d^2+o(d)}$ in time. Sieving costs are exponential in both time and memory but the asymptotic time complexity is $2^{0.292d+o(d)}$ for the best sieve algorithm, which is much lower than enumeration in a high dimension. In short, sieving is faster than enumeration when the dimension is larger than 80, approximately. The closest vector problem can be solved via Kannan's embedding technique, which changes the closest vector problem into a shortest vector problem with 1 more dimension.

Breaking (EC)DSA and Diffie–Hellman with side-channel attacks usually results in a Hidden Number Problem (HNP), which can be converted into a shortest vector problem

and solved by SVP algorithm. The Hidden Number Problem is proposed by D.Boneh and R.Venkatesan in 1996 [11] to analyze the bit-security of the private key in some key exchange schemes, such as the Diffie–Hellman key exchange scheme. Later, P. Q. Nguyen and I. E. Shparlinski analyzed the security of Digital Signature Algorithm (DSA) with partially bit-leakage in the private key by HNP. There are two main methods used to solve HNP: the original approach is due to Bleichenbacher and relies on Fourier analysis technique [12]. Another method is a lattice attack, which was discovered by Boenh and Venkatesan in Ref. [11], in which they convert the HNP into a CVP and use the LLL algorithm together with Babai's nearest plane algorithm [13] to solve it. The time and memory consumption of Bleichenbacher's method are higher compared with the lattice attack, since Bleichenbacher's method needs exponential many samples while it only needs polynomial many samples for a lattice attack; however, the Bleichenbacher's method can solve HNP with less known bits, such as HNP with only 1-bit leakage. However, as for the lattice attack, it is believed that with only 1-bit leakage, a lattice attack has difficulty succeeding [12,14], which is mainly because the lattice constructed by the adversary increases quickly to an unacceptable dimension with the decrease of the known bits.

In the general case of the Hidden Number Problem, the adversary knows some of the most significant bits of the hidden number multiples and some randomly sampled integers modulo for a given integer, which can be translated into modular equations for the hidden number. The hardness of HNP is mainly determined by the size of the modulus and the number of bits known to the adversary. When a lattice attack is applied to HNP, the number of samples affects the distance between the target vector and the lattice, and the dimension of the lattice is nearly the same as the number of samples used. A BDD solver is believed to succeed when the norm of the target vector is less than the shortest vector in the lattice, i.e., $||v|| \leq GH$. When sieving is applied, the constraint on samples used can be relaxed by a scalar factor $\sqrt{4/3}$, that is $||v|| \leq \sqrt{4/3} \cdot GH$. Since sieving is "more than SVP", it outputs all the short vectors with a norm below a bound, thus providing more information. With the development of lattice reduction algorithms, Liu and Nguyen solve 160-bit with 2-bit leakage with BKZ2.0 [15] in 2013. Albrecht and Heninger propose the idea of "predicate" [16] and utilize it with General Sieve Kernel (G6K) [17] to break the records. In 2022, Ref. [18] use bits guessing to solve HNP, for each guess, which is a closest vector problem in the same lattice with a different target. As mentioned above, the less bits are known to the adversary, the harder the HNP instance becomes, since with less bits leakage the adversary needs to construct a lattice with a higher dimension.

Contributions. We propose an asymmetric lattice sieving algorithm to solve HNP. We use more samples for BKZ pre-processing step to derive a better lattice basis while using truncated lattice basis for sieving.

Compared with previous lattice sieving methods that do not use a BKZ pre-processing step or just yse the same number of samples for both steps, we use more samples for pre-processing and get a better lattice basis, which can benefit the sieving step. How to improve the lattice attack with more samples is a question, and the first solution to it is introduced by Ref. [18], using more samples to find a special HNP instance, such as an instance with small multipliers. We take advantage of "more samples" by applying them to the pre-processing step after the lattice reduction, as the constraint introduced by each sample will propagate to other rows of the basis, and result in a better lattice basis, which is more orthogonal to each other.

We compare our algorithm with "sieve-pred", the state-of-the-art algorithm mentioned in Ref. [16]. We estimate the cost of our algorithms in various parameters and list it in Tables 1 and 2. Our algorithm can solve the problem using less time, and the comparison between our algorithm and the state-of-the art algorithm is listed in Table 3. We also experimentally verified the quality of lattice basis obtained by our BKZ pre-processing step, and compare it with previous methods. It turns out that after our BKZ pre-processing step, the lattice basis is more orthogonal compared with previous methods. We illustrate this result in Section 6. To verify our algorithm, we apply it to HNP with only 1-bit leakage,

and successfully solved them with a modulus up to 116-bit. We also successfully solved all the parameters reported in Refs. [16,18], and found that our algorithm needs less time. There are still some parameters that we cannot solve at this moment, mainly because the dimension of the lattice is too large.

Table 1. Resources required to solve HNP with 1-bit leakage.

	80-bit	**90-bit**	**100-bit**	**112-bit**	**116-bit**	**128-bit**	**160-bit**
Samples for uSVP	112	126	139	155	161	177	220
Samples for sieving	87	98	108	121	125	137	171
Δm	25	28	31	34	36	40	49
Sieving dimension	88	99	109	122	126	138	172
Sieving cost	24.6 s	179.1 s	1131.9 s	15,847.5 s	46,598.8 s	-	-

Table 2. Resources required to solve HNP.

160-bit		
Leakage	2-bit	3-bit
Sieving dimension	85	57
Sieving cost	47.3 s	<1 s
192-bit		
Leakage	2-bit	3-bit
Sieving dimension	100	67
Sieving cost	606.9 s	3.2 s
224-bit		
Leakage	2-bit	3-bit
Sieving dimension	117	77
Sieving cost	14,850.8 s	8.2 s
256-bit		
Leakage	2-bit	3-bit
Sieving dimension	134	88
Sieving cost	-	59.8 s

Table 3. Comparison with "sieve-pred".We compare the algorithms for HNP with 1-bit and 2-bit leakage with various modulus. The "Time" stands for the average time.

		1-bit Leakage				**2-bit Leakage**		
	$\log_2(q)$	**80-bit**	**90-bit**	**100-bit**	**112-bit**	**160-bit**	**192-bit**	**224-bit**
Ours	Time	24.6 s	179.1 s	1131.9 s	15,847.5 s	47.3 s	606.9 s	14,850.8 s
	Samples	87	98	108	121	84	99	116
sieve-pred	Time	25.1 s	228.7 s	3868.4 s	18,206.8 s	51.7 s	743.2 s	29,616.1 s
	Samples	87	98	108	121	87	98	116

2. Preliminaries

We use $||\cdot||$ to denote the Euclidean norm and $||\cdot||_\infty$ for infinity norm. We use $v_{[i]}$ to denote the ith entry of a vector and $A_{i,j}$ for the entry in the ith row and jth column of the matrix A. Index starts from 1 in this work.

2.1. Lattices

A lattice Λ in $\mathbb{R}^m$ is a discrete subgroup. Such a lattice is generated by a basis $B = (b_0, b_1, \cdots, b_{d-1}) \subset \mathbb{Z}^m$ of linearly independent integer vectors, as $\Lambda = \mathcal{L}(\mathrm{B}) = \mathrm{B} \cdot \mathbb{Z}^m = \{B \cdot x : x \in \mathbb{Z}\}$. We define the volume of a lattice Λ as $Vol(\Lambda) = \sqrt{det(B \cdot B^T}$,

where B is an arbitrary basis of Λ, volume is a lattice invariant since it is independent of the lattice basis used. We use $\pi_i : \mathbb{R}^d \mapsto span(b_0, b_1, \cdots, b_{i-1})^{\perp}, i = 0, 1, \cdots, d-1$ to present the orthogonal projections. Particularly, $\pi_0(\cdot)$ means the identity. We use $B^* = (b_0^*, b_1^*, \cdots, b_{d-1}^*)$ to present the Gram–Schmidt orthogonalization (GSO) of B, where the Gram–Schmidt vector $b_i^* = \pi_i(b_i)$. Let $\mu_{i,j} = \frac{\langle b_i, b_i^* \rangle}{\langle b_j^*, b_j^* \rangle}$. We use $\lambda_i(\Lambda)$ to denote the ith successive minimum, which means the smallest r such that Λ has i linearly independent vectors of the norm at most r. $\lambda_1(\Lambda)$ is the norm of a shortest vector in Λ.

Let $\mathcal{S} \subset \mathbb{R}^d$ be a measurable subset with finite volume, then we can use the Gaussian Heuristic to predicate the number of lattice points in $\mathcal{S}$:

$$\{\mathcal{S} \cap \Lambda\} \approx \frac{Vol(\mathcal{S})}{Vol(\Lambda)} \tag{1}$$

when $\mathcal{S}$ is a closed hyber ball of dimension d, which leads to the predication of the length of a non-zero shortest vector in Λ. We use $GH(\Lambda)$ to denote the expected length of a non-zero shortest vector in Λ, then $GH(\Lambda)$ is given by:

$$GH(\Lambda) = \frac{\Gamma(1 + d/2)^{1/d}}{\sqrt{\pi}} \cdot (\Lambda)^{1/d} \approx \sqrt{\frac{d}{2\pi e}} \cdot (\Lambda)^{1/d} \tag{2}$$

which is the non-zero shortest vector in a lattice usually estimated by $\lambda_1 = \sqrt{\frac{d}{2\pi e}} \cdot (\Lambda)^{1/d}$.

2.2. Hard Lattice Problems

The Shortest Vector Problem (SVP) and Closest Vector Problem (CVP) are in a center position of lattice problems. Many problems can be transformed into hard problems in lattice, which can thus be solved via lattice algorithms.

Definition 1. ***(Shortest Vector Problem (SVP)).*** *Given a lattice basis B, we need to find a non-zero shortest vector in $\Lambda(B)$, i.e., find a vector $v \in \Lambda(B)$ with $||v|| = \lambda_1(\Lambda)$*

Definition 2. ***(Closest Vector Problem (CVP)).*** *Given a lattice basis B and a target vector $t \in \mathbb{R}^d$, we need to find a lattice vector closest to the target t. There is a reduction from CVP to SVP due to Kannan [1], which we refer to as Kannan's embedding technique. For a closest vector problem with a lattice basis B and a target vector t, it constructs*

$$\mathcal{L} = \begin{pmatrix} B & 0 \\ t & \mu \end{pmatrix} \tag{3}$$

where μ is the Kannan's embedding factor. A recommended value for it is $\mathbb{E}(\frac{||t-v||}{\sqrt{d}})$. For the vector v, which is closest to t, the corresponding vector $(v - t, -\mu)$ is small.

Definition 3. ***(α-Bounded Distance Decoding (BDD_α)).*** *Given a lattice basis B and a target vector $t \in \mathbb{R}^d$ which satisfies $dist(\mathcal{L}(B), t) < \alpha \cdot \lambda_1(\mathcal{L}(B))$, it asks to find the lattice vector $v \in \mathcal{L}(B)$ which is closest to the target t.*

In this paper, we will transform CVP into SVP by Kannan's embedding technique, since it is thought to be more efficient.

2.3. Lattice Algorithms

Sieving [4–8] takes a list of points as input, denoted as $L \in \Lambda$, and searches for linear combinations of the points that are short. If the initial list is large enough, then it is believed that SVP can be solved by this process recursively. Each point in the list is sampled in polynomial time in d.

Assuming that the distribution of the angles of the lattice points in L is the same as the distribution of angles sampled randomly from the unit sphere, Phong Q. Nguyen and Thomas Vidick proposed a heuristic sieving algorithm with time complexity of $2^{0.415d+o(d)}$ and memory complexity of $2^{0.2075d+o(d)}$ [7]. Later, Thijs Larrhoven and Benne de Weger sped it up it with Locality Sensitive Hashing, achieving a time complexity of $2^{0.3366d+o(d)}$ and memory complexity of $2^{0.415d+o(d)}$ [8]. The asymptotically fastest sieve achieves a time complexity of $2^{0.292d+o(d)}$ and a memory complexity of $2^{0.415d+o(d)}$, which is sped up by using the Locality Sensitive Filter [5].

If the linear combination takes k points at the same time, it is called k-sieve. For example, 2-sieve searches for integer combinations of lattice vectors $u, v \in L$ for $u \neq \pm v$. In high dimensions, we may use the 3-sieve since it requires less memory compared with 2-sieve, but more time consumption.

The LLL Algorithm was developed by A. K. Lenstra, H. W. Lenstra, Jr and L. Lovasz in 1982, which can solve the approximate SVP by achieving an approximation factor of $(\frac{2}{\sqrt{3}})^n$. Given a parameter $\frac{1}{4} < \delta \leq 1$, a lattice basis $B = (b_0, b_1, \cdots, b_{n-1})$ is LLL reduced if the Gram–Schmidt orthogonalization of B satisfies $\mu_{i,j} \leq \frac{1}{2}$ for $i > j$, and $(\delta - \mu_{i+1,i}^2) \cdot ||b_i^{*2}|| \leq ||b_{i+1}^{*2}||$ (Lovasz conditions). Let $\alpha = 1/(\delta - 1/4)$, then the first vector of a LLL reduced basis satisfies $||b_0|| \leq \alpha^{(n-1)/2} \cdot \lambda_1(\Lambda)$. For $\frac{1}{4} < \delta < 1$, the LLL algorithm can be computed in polynomial time in the dimension.

The BKZ Algorithm was proposed by Schnorr in 1987 [19,20] and can be seen as a generalization variant of the LLL algorithm. It obtains higher quality of the output lattice basis, however, with a running time in exponential in the dimension d. The BKZ algorithm uses an oracle that solves SVP in the β dimension "block", and inserts the short vector to the lattice basis recursively. It first finds the shortest vector in the first block $\pi_1(b_1)$ and the shortest in $\pi_1(b_1)$ will be inserted to the basis. It then proceeds to the next "block" until it reaches the last "block" $\pi_{d-2}(b_{d-1})$, which is called a BKZ-tour. After a BKZ-tour, the algorithm will go to the first block and continue this process until the lattice basis remains unchanged. A small, constant number of BKZ-tour is enough for many applications.

The SVP oracle can be instantiated by enumeration or sieving. When it is instantiated by enumeration, it achieves a running time of $1.02^{\beta^2+O(\beta)}$ and a polynomial memory cost in β. As for sieving, the asymptotic time complexity becomes $2^{0.292\beta+o(\beta)}$ and the memory complexity is $2^{0.2075\beta+o(\beta)}$.

2.4. The Hidden Number Problem

In order to study the bit-security of private keys in the Diffie–Hellman key exchange scheme, the Hidden Number Problem (HNP) was first proposed by D. Boneh and R. Venkatesan in 1996 [11], who converted the HNP to CVP, using the LLL algorithm to solve it.

In the Hidden Number Problem, q, l is the fixed number known to the public and α is the secret. Given many random $t \in \mathbb{Z}$, there is an oracle $O_\alpha(t)$ that on inputs t outputs a tuple $(t, a$ such that $|\alpha \cdot t - a|_q < 2^l$, where $|x|_q$ means the unique number $0 \leq z < q$ such that $z \equiv x \bmod q$. Suppose we have queried the oracle m times and have m tuples $(t_i, a_i), i = 1, 2, \ldots, m$, then the problem asks to recover the secret α from these tuples. We will write it as $\alpha \cdot t \equiv a_i + k_i \bmod q, 0 \leq k_i < 2^l$.

The hardness of HNP is mainly determined by the number of leakage and the modulus size; more precisely, it is determined by $\frac{log_2(q)}{leakage}$. The larger the value is, the harder the HNP is.

An important application of HNP is to mount the side channel attack on (EC)DSA. We will introduce DSA and ECDSA, and then take DSA as an example to explain how to transform it into HNP.

2.5. Digital Signature Algorithm (DSA)

In DSA, p is the modulus and $g \in \mathbb{Z}_{||}^*$ is an element of order q, with $q|p-1$. Here is a hash function H which maps an arbitrary-length input into $\mathbb{Z}_{||}$. The private key is $\alpha \in \mathbb{Z}_{||}^*$ and the public key is $y \equiv g^{\alpha} \bmod p$.

A DSA signature is composed of two integers r and s, generated as follows:

$$r \equiv (g^k \bmod p) \bmod q \tag{4}$$

$$s \equiv k^{-1}(H(m) + \alpha r) \bmod q \tag{5}$$

where k is a random number in $\mathbb{Z}_{||}^*$ and is unique for each signature.

In order to verify a signature on given a pair (r, s), one needs to compute

$$g^{H(m)\cdot s^{-1}} y^{rs^{-1}} \bmod q \tag{6}$$

and check whether it equals to r.

2.6. The Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is an elliptic curve variant of DSA, and is one of the most used signature schemes. In ECDSA, the private key is a randomly generated large number x and the public key is computed by $[x]G$, where G is the base point and the multiplication is the scalar multiplication on an elliptic curve. An ECDSA signature is composed of two integers r and s, which are computed as follows:

$$r \quad is \quad the \quad x-coordinate \quad of \quad [k]G \tag{7}$$

$$s \equiv k^{-1}(H(m) + xr) \bmod p \tag{8}$$

where p is the modulus and k is a random number that is unique for each signature. We call it nonce, and $H(m)$ is the hash of the message.

2.7. (EC)DSA as HNP

In the general case of a side-channel attack against (EC)DSA, some of the most significant bits of the signature nonce k will be reveled to the adversary. Without loss of generality, we assume that these bits are zero. We will use DSA as an example to explain how to mount a lattice attack on DSA.

Since $s \equiv k^{-1}(H(m) + \alpha r) \bmod q$, we rearrange it and then have $\alpha r \equiv ks - H(m) \bmod q$. Write $k = k_1 + k_2$, where k_1 denotes the known part of the nonce k, and without loss of generality we assume that $k_1 = 0$, $k_2 < 2^l$ is the unknown part. We then have:

$$\alpha r \equiv k_2 s - H(m) \bmod q \tag{9}$$

$$\alpha \cdot rs^{-1} \equiv k_2 - s^{-1}H(m) \bmod q, k_2 < 2^l \tag{10}$$

Let $t = rs^{-1}$ and $a = -s^{-1}H(m)$, then we have a HNP equation:

$$\alpha t \equiv a + k \bmod q \tag{11}$$

2.8. Solving the HNP with Lattices

Recall that we have m tuples $(t_i, a_i), i = 1, 2, \cdots, m$, satisfying $|\alpha t_i - a_i|_q < 2^l$. Boneh and Venkatesan construct the following lattice basis and solve it via a BDD oracle:

$$\mathbb{B} = \begin{pmatrix} q & 0 & 0 & \cdots & 0 & 0 \\ 0 & q & 0 & \cdots & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ t_1 & t_2 & t_3 & \cdots & t_m & \frac{1}{q} \end{pmatrix} \tag{12}$$

Lattice $\mathcal{L}(B)$ is generated by the rows of B. The target vector is $t = (a_1, a_2, \cdots, a_m, 0)$ and the lattice vector $v = (t_1\alpha \bmod q, t_2\alpha \bmod q, \cdots, t_m\alpha \bmod q, \frac{\alpha}{q}) \in \mathcal{L}(B)$ is close to t, with $||t - v|| \leq \sqrt{m+1} \cdot 2^l$. We will call v as the hidden vector since it contains the information about the hidden number α. This method can only solve HNP with large leakage. For small leakages such as 2-bit or 1-bit leakage it will not succeed.

We can solve this BDD problem via CVP methods or use Kannan' embedding technique to change it into a shortest vector problem.

Martin R.Albrecht and Nadia Heninger use two techniques to improve the attack [16]: the recentering technique and the elimination method. These two techniques play an important role in pushing the boundaries of the unique shortest vector scenario.

The recentering technique is first described in Ref. [21] and provides a significant improvement in practice. It works as follows: since $0 \leq k_i < 2^l, i = 1, 2, \cdots, m$, we can reduce the size of k_i by 1 bit via letting $a_i^{'} = a_i + 2^{l-1}, i = 1, 2, \cdots, m$, thus $k_i^{'} = k_i - 2^{l-1}, i = 1, 2, \cdots, m$. Now we have reduced k_i by 1 bit because $-2^{l-1} \leq k_i^{'} < 2^{l-1}$.

The elimination method is described in Ref. [16]. It works as follows: since we have m equations $a_i + k_i \equiv \alpha t_i m \bmod q, i = 1, 2, \cdots, m$, we rearrange these equations and then we have:

$$\alpha \equiv t_1^{-1}(a_1 + k_1) \equiv t_2^{-1}(a_2 + k_2) \equiv \cdots \equiv t_m^{-1}(a_m + k_m) \bmod q \tag{13}$$

for each equation $t_1^{-1}(a_1 + k_1) \equiv t_i^{-1}(a_i + k_i) \bmod n, i = 2, 3, \cdots, m$, we rearrange it to get

$$a_i - t_i \cdot t_1^{-1} \cdot a_1 + k_i \equiv t_i \cdot t_1^{-1} \cdot k_1 \bmod q \tag{14}$$

thus we have a new HNP instance with $a_i^{'} = a_i - t_i \cdot t_1^{-1} \cdot a_1$ and $t_i^{'} = t_i \cdot t_1^{-1}$, now the secret is k_1 and we have $m - 1$ relations about it.

There are two advantages of this method: . it can reduce the dimension of the lattice by 1, also making the secret and the unknown parts k_i equal sized.

Let $w = 2^{l-1}$, with these two techniques, Martin R.Albrecht and Nadia Heninger construct a new lattice Λ generated by:

$$\mathbb{B} = \begin{pmatrix} q & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & \cdots & 0 & 0 & 0 \\ & \vdots & & & \vdots & & \\ 0 & 0 & 0 & \cdots & q & 0 & 0 \\ t_2^{'} & t_3^{'} & t_4^{'} & \cdots & t_m^{'} & 1 & 0 \\ a_2^{'} & a_3^{'} & a_4^{'} & \cdots & a_m^{'} & 0 & w \end{pmatrix} \tag{15}$$

There is a short vector $v = (w - k_2, w - k_3, \cdots, w - k_1, w)$ in Λ with norm $||v|| \leq \sqrt{m+1} \cdot w$. The parameter w is actually the Kannan's embedding factor and a recommended value for it is $\mathbb{E}(\frac{||t-v||}{\sqrt{d}})$. Furthermore, w is also the upper bound of the known bits after using recentering technique, since $-2^{l-1} \leq w < 2^{l-1}$.

3. Algorithm

We propose a two-step algorithm to solve the HNP. The algorithm is composed of a pre-processing algorithm and a sieving algorithm. The pre-processing algorithm takes m samples as input and outputs a BKZ-β reduced basis with $m^{'} + 1$ dimension, which is smaller than the original dimension $m + 1$.

Compared to only using BKZ reduction, we use sieving to reduce the dimension of the lattice, which is because the success condition of BKZ is different from it than sieving, mainly due to the fact that sieving can produce exponentially many short vectors while BKZ reduction cannot. The difference of dimension between BKZ and sieving is listed in Table 1, and it can be seen that for the parameters considered in this paper, the difference is large, for example, 36 for 1-bit leakage and the 116-bit modulus.

Compared with only using sieving, we add a pre-processing step; the cost is negligible when compared with the sieving step, but it produces a better lattice basis. We experimentally verified the effect of the pre-processing step and find that the basis obtained by our BKZ pre-processing step is more orthogonal than the other BKZ algorithm.

The sieving algorithm will output a list of all the short vectors with a norm smaller than $\sqrt{4/3}GH$ and we will check the list for the desired hidden number α.

3.1. Baseline

Assume that we have m tuples of (t_i, a_i). We will use the recentering technique and elimination technique mentioned above to pre-process the HNP instances, and construct a lattice in the same way as Ref. [16]. We then choose a submatrix for it and apply lattice sieving.

3.2. Pre-Processing Algorithm

We use more samples to construct lattice basis B because it can take advantage of more information about the secret and results in a better basis for solving HNP. In this way, after the BKZ-β reduction, more information about the secret will propagate to every row of the basis, and more constraints are used to the lattice basis.

After the BKZ-β reduction we will choose a $(m^{'}+1)\times(m^{'}+1)$ submatrix of B. We choose the last two columns because they contain the information about the secret, since the expected hidden vector is $v = (w-k_2, w-k_3, \cdots, w-k_m, w-k_1, w)$. We choose $m^{'}-1$ columns in the rest randomly, so the result in the hidden vector comes to $v = (w-k_{i_1}, w-k_{i_2}, \cdots, w-k_{i_{m^{'}-1}}, w-k_1, w)$.

In this way, we have a matrix $B^{'}$ that has m rows and $m^{'}+1$ columns, $m > m^{'}+1$. It is clear that $B^{'}$ has linear dependence in rows. We will therefore apply the LLL algorithm to it. There are two benefits we can get from the LLL algorithm: first, it can eliminate the linear dependence in rows conveniently, and furthermore, by using it we can get a more orthogonal basis (see Algorithm 1).

Algorithm 1 Pre-processing algorithm.

Input: m tuples of (t_i, a_i), parameter $m^{'}$ and block size β
Output: A BKZ$-\beta$ reduced basis $C_{(m^{'}+1)\times(m^{'}+1)}$

1: Construct a lattice basis with $(t_i, a_i), i = 1, 2, \cdots, m$, denoted as B
2: Perform BKZ$-\beta$ reduction on B, which results in a BKZ$-\beta$ reduced basis $B^{'}_{(m+1)\times(m+1)}$
3: Sample $m^{'}-1$ columns randomly from the first $m-2$ columns of B, create a matrix $C_{m\times(m^{'}+1)}$ with the sampled $m^{'}-1$ columns together with the last 2 columns of $B^{'}$, and all the rows of $B^{'}$
4: Perform LLL algorithm on $C_{m\times(m^{'}+1)}$ to eliminate linear dependence in rows
5: Delete the first $m^{'}+1-m$ rows, which are all-zero, to obtain a $(m^{'}+1)\times(m^{'}+1)$ matrix C
6: **return** C

3.3. Sieving

We apply the lattice sieving algorithm to the $d = m^{'}+1$ dimension lattice $\mathcal{L}(C)$. The sieving algorithm will output all the short vectors with a norm less than $\sqrt{4/3}GH$ in $\mathcal{L}(C)$, and we check the list for candidates.

We will explain how to choose $m^{'}$ to ensure that this algorithm succeeds with a high probability in the next section. We point out that since the hidden vector in $\mathcal{L}(C)$ is $v = \pm(w-k_{i_1}, w-k_{i_2}, \cdots, w-k_{i_{m^{'}-1}}, w-k_1, w)$, we can therefore recover k_1 from the $m^{'}$th column of v, and thus calculate α as described in Algorithm 2, lines 4–5.

Algorithm 2 Sieving for HNP.

Input: A lattice basis C of $m^{'}+1$ dimension
Output: The hidden number α for the HNP

Perform sieving on the lattice $\mathcal{L}(C)$ to get a list L of all short vectors with norm less than $\sqrt{4/3}GH$
for all v in the list L **do**
  if $abs(||v||_\infty) \leq w$ and $abs(v_{[m'+1]}) = w$ **then**
    Compute $k_1 = (w - v_{[m']}) \cdot (v_{[m'+1]}/w)$
    Compute $\alpha \equiv t_1^{-1} \cdot (a_1 + k_1) \bmod q$
    if α satisfies all the tuples $(a_i, t_i), i = 1, 2, \cdots, m$ **then**
      break;
      return(α)
    else
      continue;
    end if
  end if
end for
return("Failed.")

4. Analysis

4.1. Time Complexity and Memory Complexity

We construct the lattice basis B in polynomial time and the BKZ-β reduction can be computed in a running time of $2^{0.292\beta+o(\beta)}$. Random sampling as well as the LLL algorithm will complete in a polynomial time. We will later perform lattice sieving; the cost is $2^{0.292d+o(d)}$ in time for the asymptotically fastest sieving or $0.658 \cdot d - 21.11 \cdot log(d) + 119.91$ for log of cost in the CPU cycles recommended in Ref. [16].

The result in time complexity of the algorithm is

$$T = 2^{0.292\beta+o(\beta)} + 2^{0.292d+o(d)} \tag{16}$$

A recommended value of β is $\beta = d - 20$.

For memory complexity, we should hold a list of vectors output by sieving in the last step and the database for sieving during lattice sieving. The number of vectors output by sieving can be estimated by Gaussian heuristic, namely $\sqrt{4/3}^d$, and the size of the database for sieving is $O(\sqrt{4/3}^d)$. Thus, the asymptotic memory complexity is

$$M = 2^{0.2075d+o(d)} \tag{17}$$

4.2. Number of Samples

Now we analyze the number of samples needed in the problem. Let us analyze the number of samples for sieving, namely $m^{'}$, and the corresponding sieving dimension is $m^{'}+1$. It is well known that sieving can output a list of all short vectors with a norm less than $\sqrt{4/3}GH$, so we expect that the hidden vector v should be contained in the list if $||v|| \leq \sqrt{4/3}GH$. If we use the BKZ algorithm to solve the problem, since it can only provide d short vectors, the corresponding condition will become $||v|| \leq GH$. This difference will result in a large gap in the dimension of the lattice, so the BKZ algorithm should construct a much larger lattice than sieve. We list the gap in Table 1.

GH can be computed as follows: for the lattice generated by C, it is exactly the same lattice generated by randomly choosing $m^{'}$ samples and constructing a lattice basis in the same way as with Equation (15). So, the volume of $\mathcal{L}(C)$ is $Vol(\mathcal{L}(C)) = q^{m^{'}-1} \cdot w$, thus the GH is

$$GH(\mathcal{L}(C)) = \frac{\Gamma(1+(m^{'}+1)2)^{1/(m^{'}+1)}}{\sqrt{\pi}} \cdot Vol(\mathcal{L}(C))^{1/(m^{'}+1)} = \sqrt{\frac{m^{'}+1}{2\pi e}} \cdot (q^{m^{'}-1} \cdot w)^{1/(m^{'}+1)} \quad (18)$$

As for the norm of the hidden vector v, it can be bounded by $\sqrt{m^{'}+1} \cdot w$ since each entry of v is bounded by w. However, it can be estimated more precisely since it is assumed that t_i is distributed uniformly in $\mathbb{Z}_q$, and α is a random number in $\mathbb{Z}_q^*$, so we can assume that k_i is distributed uniformly and randomly in $\mathbb{Z}_q$. Thus. we can compute the expectation of $||v||$, which is the same as [16].

$$\mathbb{E}(||v||) = \sqrt{\mathbb{E}(\sum_{i=1}^{m^{'}}(w-k_i)^2 + w^2} = \sqrt{m^{'} \cdot (w-k_i)^2 + w^2} = \sqrt{\frac{m^{'} \cdot w^2}{3} + \frac{m^{'}}{6} + w^2} \quad (19)$$

Thus, we use the constraint $\mathbb{E}(v) \leq \sqrt{4/3}GH$ to find $m^{'}$. That is, we use the minimum $m^{'}$ such that

$$\sqrt{\frac{m^{'} \cdot w^2}{3} + \frac{m^{'}}{6} + w^2} \leqslant \sqrt{4/3} \cdot \sqrt{\frac{m^{'}+1}{2\pi e}} \cdot (q^{m^{'}-1} \cdot w)^{1/(m^{'}+1)} \quad (20)$$

holds.

5. Experiment on HNP

We apply the two-step algorithm on several HNP instances with only 1-bit leakage. All the experiments are performed by SageMath and G6K [17] on Intel Xeon Platinum 8280 @ 224x 4GHz. We use "uSVP" to present the method of solving HNP via uSVP, such as using the BKZ algorithm to solve HNP, and the corresponding number of samples needed is estimated by $\mathbb{E}(||v||) \leq GH$. When sieving is applied, we use $\mathbb{E}(||v||) \leq \sqrt{4/3} \cdot GH$ to estimate the number of samples needed, since "sieving is more than SVP". Δm stands for the difference of samples needed for uSVP and sieving. It can be seen that sieving can use a much smaller lattice.

We list the number of samples needed for 1-bit leakage. "uSVP" stands for solving HNP by BKZ algorithm and "Sieving" stands for using Algorithm 2. We have solved 1-bit leakage with modulus up to 116-bit, and list the expected requirements for a larger modulus. We point out that the main constraint for larger parameters is the memory consumption, for example, when solving HNP with 1-bit leakage and a 116-bit modulus, the peak memory reached is 960 GB, which is unacceptable for larger parameters.

We solved all the instances listed in Table 2, except for 2-bit leakage with a 256-bit modulus. We need to construct a lattice of dimension 132. The memory cost becomes to the main obstacle with the dimension going up.

Let us take 116-1 HNP as an example. We show how the number of samples m' affects $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$ in Figure 1 . The x-axis stands for the number of samples $m^{'}$ used for sieving. The y-axis stands for the value of $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$, since they are functions of $m^{'}$. The red line is $\mathbb{E}(||v||)$ and the black line is $\sqrt{4/3} \cdot GH$. The crossing point is the value of m' we choose to solve HNP. When $\mathbb{E}(||v||) \leq \sqrt{4/3} \cdot GH$, which means that the red line it lower than the black line, the HNP is believed to be solvable, and the corresponding minimum $m^{'}$ is the samples we use for sieving.

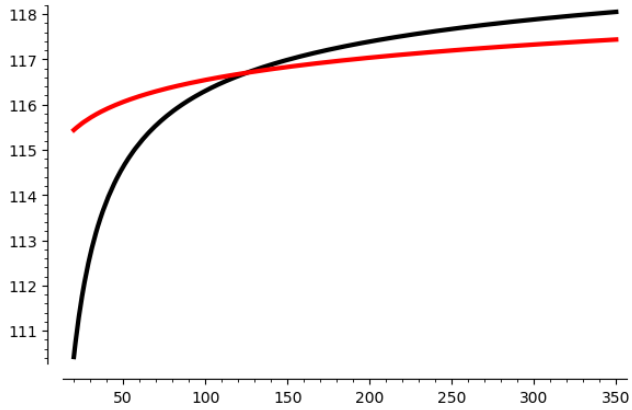


Figure 1. The curve of $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$ with respect to the number of samples m'.

We take 116-bit modulus with 1-bit leakage as an example and illustrate it in Figure 2. This figure shows how the number of samples affect the gap between $\lambda_1(\Lambda)$ and the expectation of the hidden vector. With an increasing number of samples, the value of $\mathbb{E}(||v||)/\lambda_1(\Lambda)$ decreases, and it becomes solvable when $\mathbb{E}(||v||)/\lambda_1(\Lambda) \leq \sqrt{4/3}$

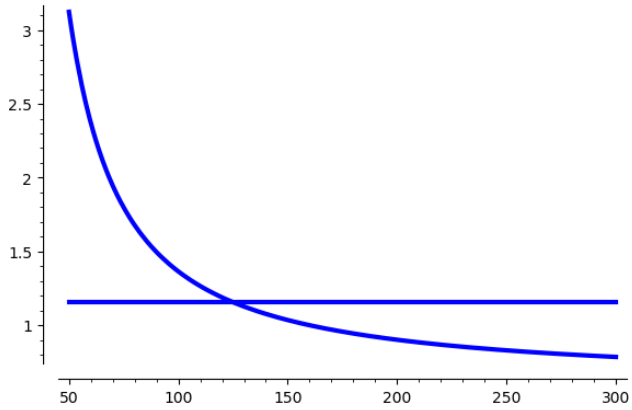


Figure 2. $\mathbb{E}(||v||)/\lambda_1(\Lambda)$ of 116-bit modulus with 1-bit leakage.

Regarding the number of samples required for Algorithm 2, the point of intersection is the number of samples that we use. However, we find that the limitation of $\mathbb{E}(||v||) \leq \sqrt{4/3} \cdot GH$ is not a necessary condition. For example, 2-bit leakage with a modulus of 160-bit is expected to be solvable with more samples than 84 but can be solved with only 77 samples with a success probability of nearly 1.

6. Comparison of BDD with Predicate

In this section, we compare our asymmetric lattice sieving algorithm with previous lattice methods. To the best of our knowledge, there are two algorithms that achieve the same result: the BDD with the predicate method [16] and the bit guessing method [18]. Experimentally, the BDD with the predicate method is faster and gives a thorough analysis of its algorithm in various parameters. So, we compare our algorithm with the algorithm mentioned in Ref. [16], which is the state-of-the-art algorithm for solving HNP. There are four algorithms mentioned in Ref. [16] for the different parameters: "BKZ-enum", "BKZ-sieve", "enum-pred", and "sieve-pred". For the parameters considered in this paper, we mainly use the "sieve-pred" algorithm to solve the problems, since "sieve-pred" is the fastest algorithm for these parameters. Therefore, we compare our algorithm with the "sieve-pred" algorithm in Ref. [16].

In this table, "Ours" stands for our asymmetric two-step sieving and "sieve-pred" stands for the "sieve with predicate" algorithm mentioned in Ref. [16]. The time is walltime

and all these experiments are performed on the same machine. In the same dimension, our algorithm obtains a better basis in the aspect of orthogonality, since we use more samples to restrict the reduction process. Let us take 2-bit leakage with a modulus of 224-bit as an example. The following figure shows that the basis obtained by our algorithm is more orthogonal. Note that the range of y-axis in "Ours" and "Previous methods" is different.

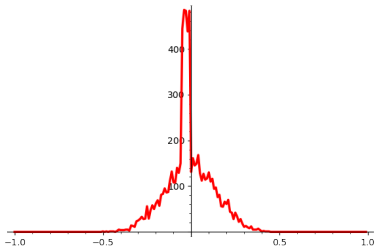
We experimentally verified the orthogonality of the lattice basis obtained by our BKZ pre-processing step, and find that it is more orthogonal to each other and thus we obtain a better lattice basis.

We demonstrate the conclusion by computing the cosine value between each basis. That is, we first generate two lattice basis: one is obtained by our BKZ-pre-processing step, denoted as $B_{ours} = (b_{ours,0}, \cdots, b_{ours,d-1})$, and the other is obtained by the previous method, denoted as $B_{previous} = (b_{previous,0}, \cdots, b_{previous,d-1})$. We then calculate the cosine values and compare them. The cosine values are calculated as follows:

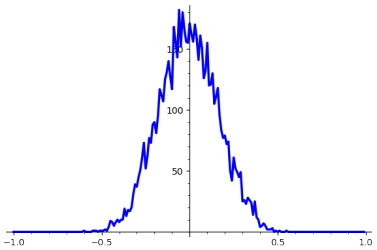
$$cos_{ours_{i,j}} = \frac{\langle b_{ours,i}, b_{ours,j} \rangle}{\|b_{ours,i}\| \cdot \|b_{ours,j}\|}, i, j = 0, 1, \cdots, d-1 \tag{21}$$

$$cos_{previous_{i,j}} = \frac{\langle b_{previous,i}, b_{previous,j} \rangle}{\|b_{previous,i}\| \cdot \|b_{previous,j}\|}, i, j = 0, 1, \cdots, d-1 \tag{22}$$

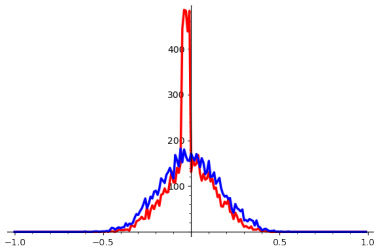
We can draw the results based on Figure 3. The x-axis stands for the cosine values and the y-axis stands for the number of the cosine values of the basis. So, these figures show the distribution of the cosine of lattice basis, It can be seen that the basis obtained by our algorithm is more orthogonal since its cosine value is more centered at zero, which means the angle is closer to $\frac{\pi}{2}$. We combine the figure "Ours" and "Previous methods" together to get the figure "Comparison". In "Comparison", the red curve stands for the cosine distribution of lattice basis obtained by our algorithm and the blue curve stands for previous methods. It can be seen that the cosine distribution is more centered at zero , which means that the basis is more orthogonal to each other.



(**a**) Ours



(**b**) Previous methods



(**c**) Comparison

Figure 3. Comparison of the cosine distribution. The x-axis stands for the cosine value and the y-axis stands for the number of angles with cosine equal to x-axis. "Ours" is our pre-processing step and "Previous methods" is the other BKZ methods.

However, there are still two problems unsolved: how to choose the pre-processing step parameter m and how the angle between the lattice basis affects the sieving step. As for the

first question, we usually choose $m = 2d$ for simplicity. That is because for the parameters considered in this paper, performing a BKZ-β reduction on a lattice of dimension $2d$ is acceptable. If we use a large m, the pre-processing step will be too expensive. As for the second question, a better basis can make it easier to find "good combinations", which will give a shorter vector in lattice. However, how the angle distribution affects the sieving step needs more rigorous analysis.

7. Conclusions

In this paper, we proposed an asymmetric lattice algorithm for HNP. We call it "asymmetric" since the algorithm uses a different number of samples for the two steps.

Compared with the BKZ algorithms, we use sieving to solve HNP since sieving can reduce the dimension of the lattice significantly, as can be seen in Table 1, parameter Δm. The main reason why sieving can reduce the dimension is that sieving can produce exponentially many short vectors while BKZ algorithms cannot. For the parameters considered in this paper, this reduction in dimension is usually over 20, which results in a significant speedup in time. Compared with sieving only, we apply a BKZ pre-processing step with more samples to make use of the information that each sample sufficiently gives. We thus expect to obtain a better lattice basis, namely, a basis that is more orthogonal.

We experimentally verified the efficiency of our algorithm, and applied it to solve HNP with 1-bit leakage with a modulus up to 116-bit. To verify the effect of the pre-processing step, we studied the "cosine distribution" of the lattice basis obtained by our algorithm and other methods, and conclude that the angle of our basis tends more to $\frac{\pi}{2}$, which means that it is more orthogonal. To verify the overall efficiency of our algorithm, we compared it with the state-of-the-art algorithm mentioned in Ref. [16]. We performed the algorithms on the same machine and compared the runtime. The results can be seen in Tables 1 and 2.

An analysis of the parameters used in this algorithm is also given. We take 1-bit leakage and a 116-bit modulus as an example, and we illustrate the effect of m' in Figures 1 and 2. For the other parameters, we list it in Appendix A. However, for the parameter m used for the pre-processing step, we simply choose $m = 2d$. This is because performing a BKZ reduction in a lattice of $2d$ dimension is acceptable and has a good result on the basis. For smaller m, the effect of pre-processing step will be reduced. More rigorous analysis and experimental verification will be done in future work.

Author Contributions: Formal analysis, W.S. and H.J.; Writing—original draft, W.S.; Writing—review & editing, W.S. and H.J.; Supervision, H.J. and Z.M.; Funding acquisition, Z.M. All authors have read and agreed to the published version of the manuscript.

Funding: Haodong Jiang was funded by the National Key R &D Program of China (No. 2021YFB3100100), and the National Natural Science Foundation of China (No. 62002385). Zhi Ma was funded by the National Natural Science Foundation of China (No. 61972413).

Institutional Review Board Statement: Not applicable.

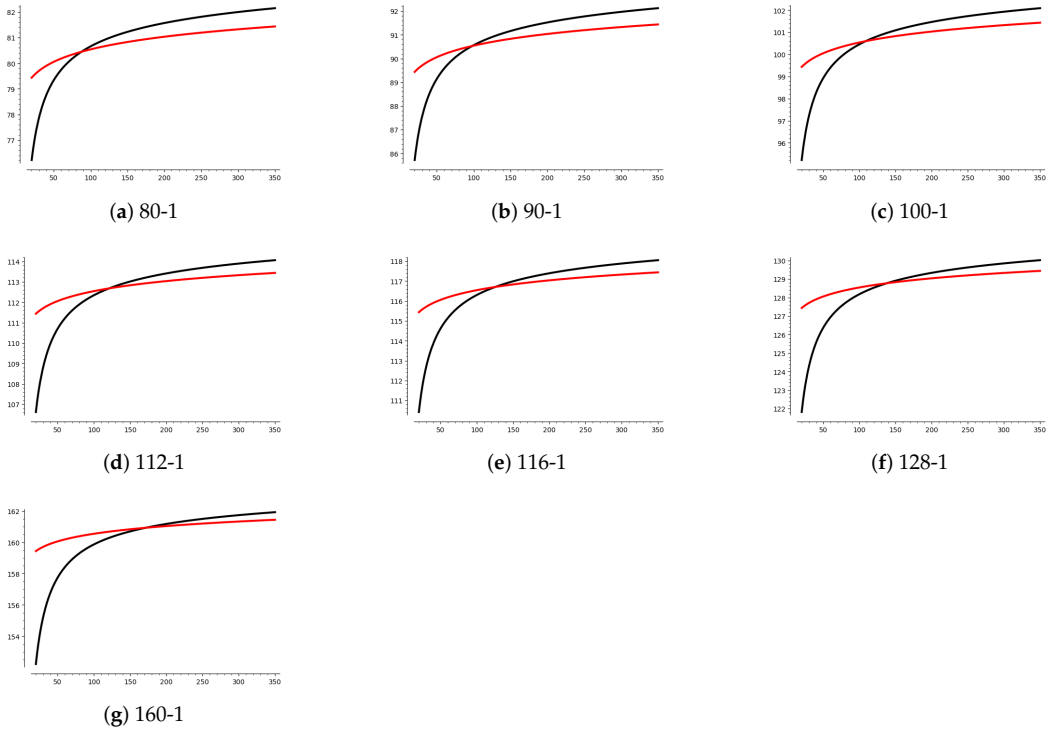
Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Here we show how does the number of samples for sieving affect the value of $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$. The same as in Figure 1, the x-axis stands for m' and the y-axis stands for the value of $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$. We use the red line to present $\mathbb{E}(||v||)$ and the black line to present $\sqrt{4/3} \cdot GH$. When the red line is lower than the black one, the corresponding HNP is thought to be solvable, which means $\mathbb{E}(||v||) \leq \sqrt{4/3} \cdot GH$.



(**a**) 80-1 (**b**) 90-1 (**c**) 100-1

(**d**) 112-1 (**e**) 116-1 (**f**) 128-1

(**g**) 160-1

Figure A1. The curve of $\mathbb{E}(||v||)$ and $\sqrt{4/3} \cdot GH$ with respect to the number of samples m'.

References

1. Albrecht, M.R.; Bai, S.; Fouque, P.A.; Kirchner, P.; Stehlé, D.; Wen, W. Faster Enumeration-based Lattice Reduction: Root Hermite Factor $k^{(1/(2k))}$ in Time$k^{(k/8+o(k))}$. Cryptology ePrint Archive, Paper 2020/707. 2020. Available online: https://eprint.iacr.org/2020/707 (accessed on 10 August 2020).
2. Micciancio, D.; Walter, M. Fast Lattice Point Enumeration with Minimal Overhead. In Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Diego, CA, USA, 4–6 January 2015; pp. 276–294. [CrossRef]
3. Gama, N.; Nguyen, P.Q.; Regev, O. Lattice Enumeration Using Extreme Pruning. In Proceedings of the Advances in Cryptology—EUROCRYPT 2010, Santa Barbara, CA, USA, 15–19 August 2010; Gilbert, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 257–278.
4. Ajtai, M.; Kumar, R.; Sivakumar, D. A Sieve Algorithm for the Shortest Lattice Vector Problem. In Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, Hersonissos, Greece, 6 July 2001.
5. Becker, A.; Ducas, L.; Gama, N.; Laarhoven, T. New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving. Cryptology ePrint Archive, Paper 2015/1128. 2015. Available online: https://eprint.iacr.org/2015/1128 (accessed on 1 January 2016).
6. Becker, A.; Gama, N.; Joux, A. Speeding-up Lattice Sieving without Increasing the Memory, Using Sub-Quadratic Nearest Neighbor Search. Cryptology ePrint Archive, Paper 2015/522. 2015. Available online: https://eprint.iacr.org/2015/522 (accessed on 24 August 2015).
7. Nguyen, P.Q.; Vidick, T. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.* **2008**, *2*, 181–207. [CrossRef]
8. Laarhoven, T. Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing. In Proceedings of the Advances in Cryptology—CRYPTO 2015, Santa Barbara, CA, USA, 16–20 August 2015; Gennaro, R.; Robshaw, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–22.
9. Micciancio, D.; Voulgaris, P. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations. *SIAM J. Comput.* **2013**, *42*, 1364–1391. [CrossRef]
10. Aggarwal, D.; Dadush, D.; Regev, O.; Stephens-Davidowitz, N. Solving the shortest vector problem in 2n time via discrete Gaussian sampling. In Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, Portland, OR, USA, 14–17 June 2015; pp. 733–742. [CrossRef]
11. Boneh, D.; Venkatesan, R. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In Proceedings of the Advances in Cryptology—CRYPTO'96, Santa Barbara, CA, USA, 18–22 August 1996; Koblitz, N., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 129–142.

12. De Mulder, E.; Hutter, M.; Marson, M.E.; Pearson, P. Using Bleichenbacher"s Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-Bit ECDSA. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2013, Santa Barbara, CA, USA, 20–23 August 2013; Bertoni, G., Coron, J.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 435–452.
13. Babai, L. On Lovász' lattice reduction and the nearest lattice point problem. In Proceedings of the STACS'85, Saarbrücken, Germany, 3–5 January 1985; Mehlhorn, K., Ed.; Springer: Berlin/Heidelberg, Germany, 1985; pp. 13–20.
14. Aranha, D.F.; Fouque, P.A.; Gérard, B.; Kammerer, J.G.; Tibouchi, M.; Zapalowicz, J.C. GLV/GLS Decomposition, Power Analysis, and Attacks on ECDSA Signatures with Single-Bit Nonce Bias. In Proceedings of the Advances in Cryptology—ASIACRYPT 2014, Kaoshiung, Taiwan, 7–11 December 2014; Sarkar, P., Iwata, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 262–281.
15. Chen, Y.; Nguyen, P.Q. BKZ 2.0: Better Lattice Security Estimates. In Proceedings of the Advances in Cryptology—ASIACRYPT 2011, Seoul, Republic of Korea, 4–8 December 2011; Lee, D.H., Wang, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 1–20.
16. Albrecht, M.R.; Heninger, N. On Bounded Distance Decoding with Predicate: Breaking the "Lattice Barrier" for the Hidden Number Problem. Cryptology ePrint Archive, Paper 2020/1540. 2020. Available online: https://eprint.iacr.org/2020/1540 (accessed on 16 June 2021).
17. Albrecht, M.R.; Ducas, L.; Herold, G.; Kirshanova, E.; Postlethwaite, E.W.; Stevens, M. The General Sieve Kernel and New Records in Lattice Reduction. Cryptology ePrint Archive, Paper 2019/089. 2019. Available online: https://eprint.iacr.org/2019/089 (accessed on 24 April 2019).
18. Sun, C.; Espitau, T.; Tibouchi, M.; Abe, M. Guessing Bits: Improved Lattice Attacks on (EC)DSA with Nonce Leakage. Cryptology ePrint Archive, Paper 2021/455. 2021. Available online: https://eprint.iacr.org/2021/455 (accessed on 14 October 2021).
19. Schnorr, C.P.; Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In Proceedings of the Fundamentals of Computation Theory, Gosen, Germany, 9–13 September 1991; Budach, L., Ed.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 68–85.
20. Schnorr, C. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.* **1987**, *53*, 201–224. [CrossRef]
21. Nguyen, P.Q.; Shparlinski, I.E. The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. *J. Cryptol.* **2000**, *15*, 151–176. [CrossRef]