IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2023, No. 2, pp. 568–586. DOI:10.46586/tches.v2023.i2.568-586

# Improved Attacks on (EC)DSA with Nonce Leakage by Lattice Sieving with Predicate

Luyao Xu[1,2], Zhengyi Dai[3], Baofeng Wu[1,2(✉)] and Dongdai Lin[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3] College of Computer, National University of Defense Technology, Changsha 410073, China
{xuluyao,wubaofeng,ddlin}@iie.ac.cn,daizhengyi@nudt.edu.cn

**Abstract.** Lattice reduction algorithms have been proved to be one of the most powerful and versatile tools in public key cryptanalysis. In this work, we primarily concentrate on lattice attacks against (EC)DSA with nonce leakage via some side-channel analysis. Previous works relying on lattice reduction algorithms such as LLL and BKZ will finally lead to the "lattice barrier": lattice algorithms become infeasible when only fewer nonce is known. Recently, Albrecht and Heninger introduced lattice algorithms augmented with a predicate and broke the lattice barrier (Eurocrypt 2021). We improve their work in several aspects.

We first propose a more efficient predicate algorithm which aims to search for the target lattice vector in a large database. Then, we combine sieving with predicate algorithm with the "dimensions for free" and "progressive sieving" techniques to further improve the performance of our attacks. Furthermore, we give a theoretic analysis on how to choose the optimal Kannan embedding factor.

As a result, our algorithm outperforms the state-of-the-art lattice attacks for existing records such as 3-bit nonce leakage for a 256-bit curve and 2-bit nonce leakage for a 160-bit curve in terms of running time, sample numbers and success probability. We also break the lattice records on the 384-bit curve with 3-bit nonce leakage and the 256-bit curve with 2-bit nonce leakage which are thought infeasible previously. Finally, we give the first lattice attack against ECDSA with a single-bit nonce leakage, which enables us to break a 112-bit curve with 1-bit nonce leakage in practical time.

**Keywords:** ECDSA · Lattice Sieving · Hidden Number Problem · Side-channel Attack · Cryptanalysis

## 1 Introduction

The Hidden Number Problem (HNP) was first introduced by Boneh and Venkatesan [BV96] in 1996, which was used to study the bit-security of the Diffie-Hellman key exchange scheme. Subsequently, Nguyen and Shparlinski [NS02, NS03] extended Boneh and Venkatesan's results on HNP to analyse the security of the (EC)DSA with partially known nonce leakage. When some information about the nonce (e.g., some of most significant bits or least significant bits) used in each signature generation is known to the attacker due to a weak random number generator or some side-channel attacks, the secret signing key can be efficiently recovered by solving an HNP instance.

There are two main approaches for solving HNP. One is the statistical approach, dating back to Bleichenbacher [Ble00], which is based on the discrete Fourier analysis techniques. The other is to transform the HNP instance to a Bounded Distance Decoding (BDD) instance which is a variant of the Closest Vector Problem (CVP) that asks to find the


Received: 2022-10-15 Accepted: 2022-12-15 Published: 2023-03-06

closest lattice vector to a given target point in an Euclidean space. The BDD instance can the be solved by Babai's nearest plane algorithm [Bab86] directly or transformed to a unique-Shortest Vector Problem (uSVP) by Kannan's embedding technique, which can be solved using lattice reduction algorithms. Compared with the statistical approach, lattice attacks require a much smaller number of samples and we only focus on lattice attacks in this work.

Although lattice attacks have already been widely used in side-channel cryptanalysis for two decades, they still remain infeasible when the number of nonce bits leaked becomes small. In 2017, Tibouchi [Tib17] made a point that 3 bits bias for a 256-bit curve was not easy and 2 bits bias was infeasible for lattice attacks on (EC)DSA. Furthermore, lattice attacks have long been considered infeasible for breaking (EC)DSA when only one bit nonce is known. Aranha et al. [AFG+14b] stressed that "due to the underlying structure of the HNP lattice, it is impossible to attack (EC)DSA using a single-bit nonce leak with lattice reduction". How to solve HNP with 1 bits known (break (EC)DSA with 1 nonce bias) by lattice attacks is a long standing open problem.

In 2021, Albrecht and Heninger [AH21] formalized two lattice problems with a predicate, and gave algorithms for solving instances of these problems. They proposed techniques for solving HNP with fewer samples, higher success probabilities and less running time. They also presented experimental evidences of their techniques' ability to solve instances given fewer samples than required by the information theoretic limit for lattice approaches. The main idea is to use the omitted information that the hidden number corresponds to the discrete logarithm of a public value. This non-linear information can be utilized in lattice attacks as a predicate on solving HNP to uniquely determine the target solution.

## 1.1 Our Contributions

In this paper, we follow the work of [AH21] and improve the lattice attacks on leaky (EC)DSA in different aspects. The main contributions are as follows.

Firstly, we propose a more efficient predicate algorithm which aims to uniquely determine the desired solution in lattice attacks. Albrecht and Heninger propose the sieving with predicate algorithm, which first uses lattice sieving algorithms to produce a database that contains plenty of short vectors in the HNP lattice, and then invokes the predicate algorithm for each vector to check whether the target lattice vector is among the database. In addition, it utilizes the information that the hidden number is the discrete logarithm of a public value which can be used to distinguish the target vector. However, this approach requires to compute a scalar multiplication over the elliptic curve for each short vector. Since the number of short vectors in the database grows exponentially with the lattice dimension, the check process will heavily increase the whole running time. To deal with this problem, we propose a linear predicate algorithm that avoids the costly scalar multiplication and only needs to do some vector multiplications.

Secondly, we combine the "sieving with predicate" algorithm and the "dimensions for free" technique [Duc18] and the result can be improved by both techniques. As stressed in [AH21], sieving with predicate algorithm conflicts with the dimensions for free technique, and the authors left investigating an intermediate regime, i.e., fewer dimensions for free, as a future work. We resolve this issue by progressively increasing the dimension of sieving [LM18, ADH+19] over the projected sublattice, and invoke the predicate algorithm to check short vectors in the output database after each subsieving. Thus we can expect an early termination before the full dimension sieving. Thanks to our improvement of the aforementioned predicate algorithm, checking the database after each subsieving will also not cost too much time.

Thirdly, we give a theoretical analysis on how to choose the optimal embedding factor arising in Kannan's embedding technique for the HNP lattice. In [SETA22], Sun et al. observed that lattice attacks on (EC)DSA were very sensitive to the embedding factor, and

**Table 1:** Comparison with the previous records of lattice attacks on (EC)DSA with nonce leakage. Each column corresponds to the size of the curves and each row corresponds to the number of nonce leakage per signatures.

| | 4-bit | 3-bit | 2-bit | 1-bit |
|---|---|---|---|---|
| 112-bit | - | - | - | Ours |
| 160-bit | - | [NS02] | [LN13], [AH21], [SETA22] | - |
| 256-bit | [Rya19], [WSBS20] | [AH21], [SETA22] | Ours | - |
| 384-bit | [AH21], [SETA22] | Ours | - | - |

they performed some experiments to explain that the embedding factor could neither be too small (equal to 1) nor too large (equal to the square of the modulus). In the previous lattice attacks on HNP using Kannan's embedding technique, the embedding factor is set to be the upper bound of the target lattice vector's coordinate [NS02, JSSS20]. We show how to choose the optimal embedding factor in order to further increase the performance of our lattice attacks.

At last, combining all the improvements above, we carry out experiments on lattice attacks on (EC)DSA with nonce leakage. Our experimental results outperform previous records as follows:

- We outperform existing best records for lattice attacks on (EC)DSA with nonce leakage (2-bit leakage for 160-bit modulus, 3-bit leakage for 256-bit modulus, 4-bit leakage for 384-bit modulus) in terms of running time, success probabilities, and number of samples.
- We give the first implementation of lattice attacks on (EC)DSA for 2-bit leakage for 256-bit modulus and 3-bit leakage for 384-bit modulus which are infeasible by previous attacks.
- As for single bit nonce leakage, previous lattice attacks could not work since the target lattice vector would not be the shortest or the closet vector anymore, while our improved sieving with predicate algorithm can deal with this problem. We conduct experiments on 112-bit secp112r1 curve with single bit nonce leakage and successfully recover the whole secret key. This is also the first implementation of lattice attacks against only 1-bit nonce leakage for ECDSA in the literature as far as we know.

### 1.2 Related Work

Attacks on (EC)DSA with nonce leakage has been separated in two routes: Bleichenbacher's attack and lattice attacks. After transforming the key recovery problem to an HNP instance, the former method solves HNP using the fast Fourier transform method together with plenty of signatures and can handle with errors as well as fewer nonce leakage [Ble00, MHMP13, AFG+14b, TTA18, ANT+20].

Our work focus on lattice attacks and can be viewed as a follow-up of [AH21]. We mainly use lattice sieving algorithm while other works mainly use lattice reduction algorithms such as LLL and BKZ. We compare our results with previous lattice records in Table 1.

## 2 Preliminaries

All vectors are denoted by bold lower case letters and are read as row vectors. Matrices are denoted by bold capital letters. We use $\langle \cdot, \cdot \rangle$ to represent the Euclidean inner product of two vectors.

## 2.1 Lattices

A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^d$. Given a set of $m$ independent vectors $\boldsymbol{B} = (\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{m-1}) \subset \mathbb{R}^d$, we denote the lattice spanned by $\boldsymbol{B}$ by $\mathcal{L}(\boldsymbol{B}) = \{\sum_{i=0}^{m-1} z_i \cdot \boldsymbol{b}_i \mid z_i \in \mathbb{Z}\}$, and $\boldsymbol{B}$ is called a basis of $\mathcal{L}(\boldsymbol{B})$. We denote by $\boldsymbol{B}^* = (\boldsymbol{b}_0^*, \boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_{m-1}^*)$ the Gram-Schmidt orthogonalization (GSO) of the matrix $\boldsymbol{B}$ where $\boldsymbol{b}_0^* = \boldsymbol{b}_0$ , $\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \cdot \boldsymbol{b}_j^*$ for $i \in \{0, \ldots, m-1\}$ and $\mu_{i,j} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle}{\langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^* \rangle}$.

We denote the orthogonal projections $\pi_i : \mathbb{R}^d \mapsto span(\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1})^{\perp}$ for $i \in \{0, \ldots, m-1\}$. In particular, $\pi_0(\cdot)$ is the identity map and $\pi_i(\boldsymbol{b}_i) = \boldsymbol{b}_i^*$. We denote by $\boldsymbol{B}_{[i,j]}$ the local projected block $(\pi_i(\boldsymbol{b}_i), \ldots, \pi_i(\boldsymbol{b}_{j-1}))$ for $0 \leq i < j \leq m$ and when the basis is clear from the context, $\mathcal{L}_{[i:j]}$ denotes the lattice generated by $\boldsymbol{B}_{[i,j]}$.

The Euclidean norm of a vector $\boldsymbol{v}$ is denoted by $||\boldsymbol{v}||$. We denote $\lambda_1(\mathcal{L})$ as the first successive minimum of the lattice $\mathcal{L}$ which refers to the length of a shortest non-zero lattice vector. The volume of a lattice $\mathcal{L}(\boldsymbol{B})$ is the absolute value of the determinant of any basis, which is an invariant of the lattice and it holds that $\mathrm{Vol}(\mathcal{L}(\boldsymbol{B})) = \prod_{i=0}^{m-1} ||b_i^*||$.

The Gaussian heuristic predicts that the number of lattice points inside a measurable body $\mathcal{B} \subset \mathbb{R}^d$, i.e., $|\mathcal{L} \cap \mathcal{B}|$, is approximately equal to $\mathrm{Vol}(\mathcal{B})/\mathrm{Vol}(\mathcal{L})$. And it leads to the following prediction of $\lambda_1(\mathcal{L})$ (we denote by $\mathrm{gh}(\mathcal{L})$) for a given lattice $\mathcal{L}$ :

$$\mathrm{gh}(\mathcal{L}) = \left(\frac{\mathrm{Vol}(\mathcal{L})}{\mathrm{Vol}(\mathcal{B}_d(1))}\right)^{1/d} = \frac{\Gamma(1+\frac{d}{2})^{1/d}}{\sqrt{\pi}} \cdot \mathrm{Vol}(\mathcal{L})^{1/d} \approx \sqrt{\frac{d}{2\pi e}} \cdot \mathrm{Vol}(\mathcal{L})^{1/d},$$

where $\mathrm{Vol}(\mathcal{B}_d(1))$ denotes the volume of a $d-$dimensional Euclidean ball with radius 1.

The last step above uses Stirling's formula to simplify the estimation which gives a very intuitive asymptotic relationship between the shortest vector length and the lattice dimension under the Gaussian heuristic. In the actual attack, we do not use this asymptotic estimation but directly calculate the value of the gamma function instead.

## 2.2 Hard Problems

There are many computational hard problems related to lattices. The most famous one is the Shortest Vector Problem (SVP) which asks to find the shortest non-zero vector in a lattice. Another one is the Closest Vector Problem (CVP) which asks to find the lattice vector nearest to a target point for a given lattice.

In [AH21], the authors formalized two lattice problems augmented with a predicate which asks to find a lattice vector $\boldsymbol{v}$ not only to be short or close to some target point, but also can satisfy a predicate $f(\cdot)$ which can help to distinguish the desired vector. They are defined as follows:

**Definition 1** ($\alpha$-Bounded Distance Decoding with Predicate ($\mathrm{BDD}_{\alpha,f(\cdot)}$))**.** Given a lattice basis $\boldsymbol{B}$, a vector $\boldsymbol{t}$, a predicate $f(\cdot)$, and a parameter $\alpha > 0$ such that the Euclidean distance $dist(\boldsymbol{t}, \boldsymbol{B}) < \alpha \cdot \lambda_1(\boldsymbol{B})$, find the lattice vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ satisfying $f(\boldsymbol{v} - \boldsymbol{t}) = 1$ which is closest to $\boldsymbol{t}$.

**Definition 2** (unique Shortest Vector Problem with predicate ($\mathrm{uSVP}_{f(\cdot)}$))**.** Given a lattice $\mathcal{L}$ and a predicate $f(\cdot)$, find the shortest non-zero lattice vector $\mathbf{v} \in \mathcal{L}$ satisfying $f(\boldsymbol{v}) = 1$.

Similar to the reduction from $\mathrm{BDD}_\alpha$ to $\mathrm{uSVP}_\gamma$ [LM09], $\mathrm{BDD}_{\alpha,f(\cdot)}$ can be solved by using a $\mathrm{uSVP}_{f(\cdot)}$ oracle due to Kannan's embedding technique [Kan87], i.e., by constructing the lattice

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{B} & 0 \\ \boldsymbol{t} & \tau \end{pmatrix}$$

where $\tau$ is some embedding factor. If $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ is the closest vector to $\boldsymbol{t}$ then the lattice $\mathcal{L}(\boldsymbol{B})$ contains the vector $(\boldsymbol{t} - \boldsymbol{v}, \tau)$, which is short.

## 2.3 Lattice Sieving

Sieving algorithms were first proposed by Ajtai et al. [AKS01] in 2001. They are the asymptotically the fastest SVP solvers as known so far. The basic sieve algorithm first builds a list $L$ which contains exponentially-sized lattice vectors, then look for pairs of lattice vectors $(\boldsymbol{u}, \boldsymbol{v}) \in L^2$ whose integer combinations can form a shorter lattice vector. Once such a pair is found, sieve algorithm replaces the original long lattice vector by the new shorter one. After performing this process recursively, it will output a list with exponentially-sized lattice vectors which are preferably short and we are expected to find the shortest lattice vector finally by subtracting each other among these vectors. Heuristic variants [NV08, MV10, WLTB11, ZPH13, Laa15, LdW15, BDGL16, BLS16, HK17, HKL18] were proposed successively in order to reduce both time and space complexity.

Recently, new techniques have appeared and they further improve the practical performance of the sieving algorithms. Dimensions for free (D4F) [Duc18] and progressive sieving [LM18] are some rank reduction techniques which can solve the original SVP in lower lattice dimension, i.e., with less running time and memory cost and they can be applied to most variants of sieve algorithms by design. Based on these two strategies, G6K [ADH+19] was proposed by Albrecht et al. which is an abstract stateful machine supporting a wide variety of lattice reduction strategies based on sieving algorithms. The highly optimised and tweakable implementation of G6K outperforms enumeration algorithms [Kan83, GNR10] in practice for solving exact-SVP for dimension as lower as 70. In 2021, Ducas et al. [DSvW21] proposed a GPU implementation of G6K, which largely improve the performance of sieving algorithms in practice.

## 2.4 ECDSA

Elliptic Curve Digital Signature Algorithm (ECDSA) is one of the most popular signature schemes nowadays and can be described as in Algorithm 1.

**Algorithm 1** ECDSA Signature Generation

**Input:** Message $M \in \{0,1\}^*$, domain parameters $\mathcal{D}$, signing key $\mathbf{sk} \in \mathbb{Z}_q$
**Output:** A valid signature $(r, s)$
1: Generate a random integer nonce $k \in \mathbb{Z}_q$
2: Compute $R = (r_x, r_y) \leftarrow [k]G$
3: Compute $s = k^{-1} \cdot (H(m) + r_x \cdot \mathbf{sk}) \mod q$
4: **return** $(r_x, s)$

Here the domain parameters are $\mathcal{D} := (E, p, G, q, H)$ where $E$ is an elliptic curve over $\mathbb{F}_p$, $G$ is a generator point on $E$ of order $q$ and $H$ is a cryptographic hash function.

## 2.5 The Hidden Number Problem

Boneh [BV96] first formalised the Hidden Number Problem (HNP) and we denote HNP$(n, l)$ as follows: There is a $n$-bit sized public modulus $q$ and a secret integer $\alpha \in \mathbb{Z}_q$ which we call the hidden number; $t_0, t_1, \ldots, t_{m-1}$ are some integers chosen uniformly and independently at random in $\mathbb{Z}_q$. For each $t_i$, we are given $a_i$ such that $|t_i \cdot \alpha - a_i|_q < q/2^l$ where $|z|_q$ is defined as the unique integer $0 \leq x < q$ such that $x \equiv z \mod q$ . When given $m$ such pairs, i.e., $(t_i, a_i)$ for $0 \leq i \leq m-1$, the problem asks to recover the hidden number $\alpha$.

## 2.6 Leaky ECDSA as an HNP instance

In [NS03], Nguyen first reduced the (EC)DSA with nonce leakage problem to the hidden number problem. In a side-channel attack against ECDSA, the adversary may retrieve $l$

least significant bits of the signature nonce $k$. We write $k = k_0 \cdot 2^l + k_1$ where $0 \leq k_0 < q/2^l, 0 \leq k_1 < 2^l$ and since we already know $s = k^{-1} \cdot (H(m) + r_x \cdot \mathbf{sk}) \mod q$ from the signature generation phase, we can derive

$$k_0 \cdot 2^l + k_1 = s^{-1} \cdot H(m) + s^{-1} \cdot r_x \cdot \mathbf{sk} \mod q.$$

Multiplying by the inverse of $2^l$ and rearranging yield

$$2^{-l} \cdot s^{-1} \cdot r_x \cdot \mathbf{sk} = k_0 + 2^{-l}(k_1 - s^{-1} \cdot H(m)) \mod q.$$

This forms an HNP$(n, l)$ instance with $(t_i, a_i) = (2^{-l} \cdot s^{-1} \cdot r_x, 2^{-l}(k_1 - s^{-1} \cdot H(m)))$ and the secret key $\mathbf{sk}$ is the hidden number we wish to find. Solving the HNP instance can thus recover the secret key.

# 3 Solving HNP with Lattices

After transforming the key recovery attacks on leaky ECDSA into an HNP instance, the remaining problem is to recover the hidden number for a given HNP instance. In this section, we will first introduce the lattice attack model for solving HNP and some known optimizations. Then we will give a comprehensive analysis on the success condition for different lattice algorithms and estimate the least number of signature samples.

## 3.1 Lattice Attack Model and Kannan's Embedding

In 1996, Boneh and Venkatesan [BV96] gave the following lattice construction to solve the Hidden Number Problem:

$$\boldsymbol{L} = \begin{bmatrix} q & 0 & 0 & \cdots & 0 & 0 \\ 0 & q & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & q & 0 \\ t_0 & t_1 & t_2 & \cdots & t_{m-1} & 1/2^l \end{bmatrix}$$

From the hidden number problem, the equation $a_i + u_i = t_i \cdot \alpha \ mod\ q$ holds for all $0 \leq i \leq m-1$ where $\alpha$ is the hidden number and $0 \leq u_i < q/2^l$. According to the lattice basis, there exists a lattice vector

$$\boldsymbol{v} = (t_0 \cdot \alpha + q \cdot z_0, \ldots, t_{m-1} \cdot \alpha + q \cdot z_{m-1}, \alpha \cdot 1/2^l),\ z_i \in \mathbb{Z},\ i \in \{0, \ldots, m-1\}$$

which comes from multiplying the last row by $\alpha$ and adding the other rows multiply by some integer $z_i$.

This lattice vector is close to the target vector $\boldsymbol{t} = (a_0, \ldots, a_{m-1}, 0)$ and the distance can be bounded by $\|\boldsymbol{v} - \boldsymbol{t}\| = \|(u_0, u_1, \ldots, u_{m-1}, \alpha \cdot 1/2^l)\| \leq \sqrt{m+1} \cdot q/2^l$. When this distance is small enough compared to other distances between the lattice vectors and the target vector, the lattice vector $\boldsymbol{v}$ can be found by solving the BDD problem thus the hidden number $\alpha$ can be recovered.

The BDD problem can be solved directly by the nearest plane algorithm [Bab86] or by transforming it to the Unique-SVP using Kannan's embedding [Kan83] technique. It is concluded in [JSSS20, SETA22] that Kannan's embedding approach always outperforms nearest plane algorithm for solving HNP. We thus follow Kannan's embedding route and

construct the following lattice basis:

$$\begin{bmatrix} q & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & 0 \\ 0 & 0 & 0 & \cdots & q & 0 & 0 \\ t_0 & t_1 & t_2 & \cdots & t_{m-1} & 1/2^l & 0 \\ a_0 & a_1 & a_2 & \cdots & a_{m-1} & 0 & \tau \end{bmatrix} \tag{1}$$

The lattice dimension is $d = m + 2$ where $m$ is the equation number and the target lattice vector becomes

$$\begin{aligned} \boldsymbol{v} &= \pm\left(t_0 \cdot \alpha - a_0 + q \cdot z_0, \ldots, t_{m-1} \cdot \alpha - a_{m-1} + q \cdot z_{m-1}, \alpha \cdot 1/2^l, -\tau\right) \\ &= \pm\left(u_0, u_1, \ldots, u_{m-1}, \alpha \cdot 1/2^l, -\tau\right) \end{aligned} \tag{2}$$

where $\tau$ is the embedding factor which is set to be the upper bound of $u_i$ in the literature, i.e., $\tau = q/2^l$. The norm of the target vector thus can be bounded by $\sqrt{m+2} \cdot q/2^l$. When the norm is shorter than others lattice vector, the target vector is expected to be founded by the lattice reduction algorithm (LLL, BKZ..) or the SVP algorithm (Enumeration, Sieving..). After running the lattice reduction algorithm or the SVP algorithm, we check if any rows of the reduced lattice basis contains the target lattice vector, i.e., it may not be the shortest lattice vector but still in the reduced lattice basis.

There are some known changes to the lattice basis that can be made to enhance the lattice attack. In our attacks, we primarily apply the following two improvements:

### 3.1.1 Recentering

Recentering technique is widely used in the lattice attack on HNP [NS02, MSEH20, JSSS20, AH21, SETA22]. As shown in (2), the target lattice vector we expect is $(u_0, u_1, \ldots, u_{m-1}, \alpha \cdot 1/2^l, -\tau)$ with $0 \leq u_i < q/2^l$. Adding $w = \lfloor q/2^{l+1} \rfloor$ to each $a_i$, we will get a new target vector which is much shorter than the original one:

$$\begin{aligned} &(\alpha \cdot t_0 - (a_0 + w) + q \cdot z_0, \ldots, \alpha/2^l, -\tau) \\ =&(u_0 - w, u_1 - w, \ldots, u_{m-1} - w, \alpha/2^l, -\tau) \end{aligned} \tag{3}$$

### 3.1.2 Eliminate $\alpha$

In our lattice attack model, we expect our target vector shorter than all other vectors in the lattice. However, there is a trivial short vector $(0, 0, \ldots, q/2^l, 0)$ (always the shortest lattice vector) which comes from multiplying the second-to-last basis vector with $q$ and subtract the $(i+1)$-th row vector multiplying with $t_i$ for all $0 \leq i \leq m-1$. [MSEH20, AH21] deal with this issue by eliminating the variable $\alpha$ and constructing a new lattice basis, this method is also mentioned in [SETA22] where they call it a projected lattice technique. Details are as follows:

In the original hidden number problem, we have

$$\begin{cases} a_0 + u_0 = t_0 \cdot \alpha \mod q \\ a_1 + u_1 = t_1 \cdot \alpha \mod q \\ \cdots \\ a_{m-1} + u_{m-1} = t_{m-1} \cdot \alpha \mod q \end{cases} \tag{4}$$

Replace $\alpha = (a_0 + u_0) \cdot t_0^{-1} \mod q$ in (4) and rearrange we have

$$\begin{cases} a_1 - a_0 \cdot t_0^{-1} \cdot t_1 + u_1 = t_0^{-1} \cdot t_1 \cdot u_0 \mod q \\ a_2 - a_0 \cdot t_0^{-1} \cdot t_2 + u_2 = t_0^{-1} \cdot t_2 \cdot u_0 \mod q \\ \cdots \\ a_{m-1} - a_0 \cdot t_0^{-1} \cdot t_{m-1} + u_{m-1} = t_0^{-1} \cdot t_{m-1} \cdot u_0 \mod q \end{cases} \tag{5}$$

Thus we get a new HNP instance $(a'_i, t'_i) = (a_i - a_0 \cdot t_0^{-1} \cdot t_i,\ t_0^{-1} \cdot t_i)$ for all $1 \le i \le m-1$. The hidden number becomes $u_0$ in the new instance and the trivial short vector $(0, 0, \ldots, q/2^l, 0)$ will not exist in the new lattice anymore. In addition, the dimension of the lattice basis can be also reduced by 1.

After the two improvements of the lattice basis, the final lattice basis is as follows:

$$\boldsymbol{B} = \begin{bmatrix} q & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & q & 0 & 0 \\ t'_1 & t'_2 & t'_3 & \cdots & t'_{m-1} & 1 & 0 \\ a'_1 & a'_2 & a'_3 & \cdots & a'_{m-1} & 0 & \tau \end{bmatrix} \tag{6}$$

and the target vector is $\boldsymbol{v} = \pm(u_1 - w, u_2 - w, \ldots, u_{m-1} - w, u_0 - w, -\tau)$ which can be bounded by $\sqrt{m \cdot w^2 + \tau^2}$.

## 3.2 Analysis of The Lattice Attack Model

After transforming the HNP instance into a Unique-SVP instance, most work [NS02, MSEH20, JSSS20, SETA22] then call the lattice reduction algorithms on the lattice basis and check if the target vector is in the reduced basis. Moghimi et al. [MSEH20] elaborate that "the inner workings of these lattice basis reduction algorithms are complex and we use them as a black box." In fact, treating the lattice algorithms as a black box will lead to the "lattice barrier" [AFG+14b, ANT+20]: lattice attacks fail with high probability when the nonce leakage is small (e.g., 1-bit nonce leakage). In this subsection, we give a comprehensive analysis on the the success condition for solving HNP by different lattice algorithms.

We first give some elementary analysis of the constructed Unique-SVP lattice, the crucial thing is to estimate the ratio of the second-shortest lattice vector to the shortest one (target lattice vector), i.e., $\frac{\lambda_2}{\lambda_1}$.

Assuming heuristically that $\mathcal{L}(\boldsymbol{B})$ behaves like a random lattice, the length of shortest vector under Gaussian heuristic in the constructed lattice basis (6) is expected to be

$$\mathrm{gh}(\mathcal{L}(\boldsymbol{B})) \approx \frac{\Gamma(1 + (m+1)/2)^{1/(m+1)}}{\sqrt{\pi}} \cdot \mathrm{Vol}(\mathcal{L}(\boldsymbol{B}))^{1/(m+1)} \tag{7}$$

The volume of the the lattice can be easily calculated by multiply the diagonal element of the lattice basis, i.e., $\mathrm{Vol}(\mathcal{L}(\boldsymbol{B})) = q^{m-1} \cdot \tau$.

The target vector $\boldsymbol{v}$ has a upper bound $\sqrt{m \cdot w^2 + \tau^2}$. [AH21] observed that using the expected norm of a uniformly distributed vector will result in more tractable instances since heuristically the instances are randomly sampled. Following the analysis in [AH21], the expected squared norm of the target vector $\boldsymbol{v}$ will be:

$$\mathrm{E}[\|\boldsymbol{v}\|^2] = \frac{m}{3} \cdot w^2 + m/6 + \tau^2 \tag{8}$$

### 3.2.1 BKZ

Following the idea of solving Unique-SVP using lattice reduction algorithms [GN08, AFG14a, APS15], the lattice reduction algorithms (e.g., LLL,BKZ) is expected to be successful in recovering $\boldsymbol{v}$ if

$$\lambda_2(\mathcal{L}(\boldsymbol{B}))/\lambda_1(\mathcal{L}(\boldsymbol{B})) \geq \omega \cdot {\delta_0}^{m+1}$$

where $\lambda_1(\mathcal{L}(\boldsymbol{B})) = \|\boldsymbol{v}\|$ and $\lambda_2(\mathcal{L}(\boldsymbol{B}))$ is assumed to correspond to $\mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$, $\omega$ is a constant depending on the lattice family and the lattice reduction algorithm used, $\delta_0$ is the root-Hermite factor of the lattice reduction algorithm. In [SETA22], the authors treated $w$ as a function on the lattice dimension $m$, they did some experiments and gave an estimate that $\omega = \frac{3.11}{\log(m)}$. In their experiments, they used BKZ-30 and set $\delta = 1.01$ from their experimental data.

However, it is hard to accurately estimate the minimal number of required samples or how many block sizes for BKZ is enough for a given HNP instance. Therefore, most works choose a small block size for BKZ (e.g., 25 or 30) and experimentally determine the signature samples required for a considerable success probability. While it is feasible when the bias nonce is large (e.g., $l \geq 5$), things get tough for small biases. Tibouchi [Tib17] made a point that 3 bits bias for a 256 bits curve is not easy and 2 bits bias is infeasible, 5 or 4 bits bias for a 384 bits curve is not easy and 3 bits bias is infeasible. Furthermore, [AFG+14b] stressed that "Due to the underlying structure of the HNP lattice, it is impossible to attack (EC)DSA using a single-bit nonce leak with lattice reduction."

### 3.2.2 SVP

Since we expect the target lattice vector to be the shortest lattice vector, a more intuitive thought is to use exact-SVP algorithms such as lattice enumeration or sieving algorithms. In this work, we focus only on lattice sieving algorithms since it has been already outperforming lattice enumeration in practice ( [Duc18, ADH+19]) for solving SVP with relatively high dimensions (e.g., >70). Unlike BKZ algorithm, the performance of sieving depends greatly on the lattice dimensions, thus we have to accurately estimate the number of samples needed which determines lattice sieving dimension and success probability. On the one hand, the success probability gets higher for solving Unique-SVP when the gap $\lambda_2(\mathcal{L}(\boldsymbol{B}))/\lambda_1(\mathcal{L}(\boldsymbol{B})) = \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))/\|\boldsymbol{v}\|$ increase. On the other hand, the gap increase with the lattice dimension by simply using more samples, however it will also lead to a exponentially grow in time and space complexity. So the question is: how many samples do we need at least for solving different HNP instances and how difficult they are?

We predict the number of samples needed theoretically for solving HNP with a SVP solver. For varies of HNP instances, we choose the least number of samples such that $\|\boldsymbol{v}\| \leq \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$:

$$\begin{aligned} M &= \min_m \{\|\boldsymbol{v}\| \leq \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))\} \\ &= \min_m \left\{ \frac{\sqrt{\frac{m}{3} \cdot w^2 + m/6 + \tau^2}}{\frac{\Gamma(1+(m+1)/2)^{1/(m+1)}}{\sqrt{\pi}} \cdot (q^{m-1} \cdot \tau)^{1/(m+1)}} \leq 1 \right\} \end{aligned} \tag{9}$$

$M$ can be found easily according to the above inequality by increasingly enumerating the integer $m$ until the inequality is satisfied. Table 2 shows the number of signatures $M$ expected at least for solving different HNP instances with a SVP oracle and the dimension of the corresponding lattice basis is $d = M + 1$. For example, for 256-bits curve with 3 bits leakage, 93 signature samples are needed with a cost of solving a 94-dimension SVP.

In 2018, Ducas [Duc18] proposed the dimensions for free (D4F) technique which greatly improves the practical performance of sieving. In a nutshell, sieving algorithms are

**Table 2:** Number of signatures expected at least to solve HNP with SVP-oracle.

| log(q) | Bits known | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8-bit | 7-bit | 6-bit | 5-bit | 4-bit | 3-bit | 2-bit | 1-bit |
| 160-bit | 21 | 24 | 28 | 34 | 43 | 58 | 91 | 212 |
| 192-bit | 25 | 29 | 34 | 41 | 51 | 70 | 109 | 254 |
| 256-bit | 33 | 38 | 45 | 54 | 68 | 93 | 146 | 340 |
| 384-bit | 50 | 57 | 67 | 81 | 102 | 140 | 219 | 511 |
| 521-bit | 68 | 77 | 91 | 110 | 139 | 189 | 297 | 695 |

expected to find the shortest vector in a lower lattice dimension due to the fact that sieving algorithms not only derive the shortest vector, but also output exponential lattice vectors whose norm is shorter than $\sqrt{4/3} \cdot \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$. According to D4F, solving a $d$-dimension SVP only needs to sieve in a $d' = d - \Theta(d/\log d)$ dimension sub-lattice where respectively under the pessimistic and optimistic conditions [Duc18]:

$$d' = d - \frac{d \ln 4/3}{\ln(d/2\pi)}(\text{pessimistic}) \quad \text{and} \quad d' = d - \frac{d \ln 4/3}{\ln(d/2\pi e)}(\text{optimistic}) \tag{10}$$

In practical sieving, dimensions for free technique can always be used together with progressive sieving technique [Duc18, LM18, ADH+19] which start at a low sieving dimension and increase the sieving dimension progressively. Thus we do not need to determine $d'$ in advance but increase the sieving dimension progressively until we find the target lattice vector or reaching the max dimension.

According to (10), we are expected to solve HNP(160,2) with 91 samples with the sieving dimension up to 80 approximately.

### 3.2.3 Sieving with Predicate

Inspired by the dimensions for free technique, Albrecht [AH21] proposed the sieving with predicate algorithm (Algorithm 2) for solving $\mathrm{uSVP}_{f(\cdot)}$ and $\mathrm{BDD}_{\alpha,f(\cdot)}$ which also utilize the fact that sieving will output a database containing all lattice vectors that are shorter than $\sqrt{4/3} \cdot \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$ .

**Algorithm 2** Sieving with Predicate ( [AH21])

**Input:** Lattice basis $\boldsymbol{B}$, predicate $f(\cdot)$.
**Output:** $\boldsymbol{v}$ such that $\|\boldsymbol{v}\| \leq \sqrt{4/3} \cdot \mathrm{gh}(\Lambda(\mathrm{B}))$ and $f(\boldsymbol{v}) = 1$ or $\perp$

```
r ← ⊥;
Run sieving algorithm on Λ(B) and denote output list as L;
for v ∈ L do
    if f(v) = 1 and (r = ⊥ or ‖v‖ < r) then
        r ← v
    end if
end for
return r;
```

After getting the database $L$ from the return of a full sieving on the lattice basis, a following step is to check and recover the target vectors in the database and a predicate function is introduced to complete this task. For every short lattice vector in the database, [AH21] recover the candidate secret key $sk$ and then compute the scalar multiplication $[sk]G$ over the elliptic curve to see if it equals to the given public key.

According to this approach, the target lattice vector does no need to be the shortest lattice vector or to be in the reduced basis anymore. It can be found successfully if it is shorter than $\sqrt{4/3} \cdot \mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$. Thus the signature samples needed and lattice dimension can be reduced further:

$$M = \min_m \left\{ \|\boldsymbol{v}\| \leq \sqrt{4/3} \cdot \mathrm{gh}(\mathcal{L}(\boldsymbol{B})) \right\}$$

Table 3 shows the number of signatures $M$ expected at least for solving different HNP instances using sieving with predicate algorithm. Compared with Table 2, sieving with predicate seems to improve more when nonce leakage is small.

Although sieving with predicate algorithm reduces the number of samples needed and the dimension of the lattice basis, it has to do a full sieving which can not utilise the strength of dimensions for free technique. In particular, sieving with D4F outperforms sieving with predicate for most HNP instances since the former may use a less sieve dimension in practice. Besides, sieving with predicate algorithm will invoke exponential many times of predicate algorithm including the costly scalar multiplication operation which will heavily increase the solving time.

**Table 3:** Number of signatures expected at least to solve HNP with by sieving with predicate.

| $\log(q)$ | Bits known | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8-bit | 7-bit | 6-bit | 5-bit | 4-bit | 3-bit | 2-bit | 1-bit |
| 160-bit | 20 | 23 | 27 | 32 | 41 | 54 | 82 | 165 |
| 192-bit | 24 | 28 | 32 | 39 | 49 | 65 | 98 | 199 |
| 256-bit | 32 | 37 | 43 | 52 | 65 | 86 | 130 | 266 |
| 384-bit | 49 | 55 | 65 | 78 | 97 | 130 | 196 | 400 |
| 521-bit | 66 | 75 | 88 | 105 | 132 | 176 | 266 | 544 |

# 4 Improved Sieving with Predicate

## 4.1 Linear Predicate Algorithm

Unlike using the non-linear information to determine the unique target lattice vector, we give a new predicate algorithm which is much faster than the one in [AH21]. Recall that we have equations $a_i + u_i = t_i \cdot \alpha \mod q$ for $0 \leq i \leq m-1$ where $(a_i, t_i)$ are known and the target vector is:

$$\boldsymbol{v} = (u_1 - w, u_2 - w, \ldots, u_{m-1} - w, u_0 - w, -\tau)$$

We give the following linear predicate algorithm (Algorithm 3) to determine whether a vector in database is our target lattice vector.

In our linear predicate algorithm, we first check whether the absolute value of the last coordinate equals to the embedding factor $\tau$, then we compute the candidate hidden number $\alpha$ from the vector and check the candidate hidden number according to the modular systems of linear equations (4).

Compared this with the predicate algorithm in [AH21] which computes a scalar multiplication each time. Our linear predicate algorithm only needs to do some modular multiplications to check the target lattice vector. When enumerating a database of lattice vectors following a 90 dimension lattice sieving, our linear predicate approach is more

than 100 times quicker than the non-linear predicate technique in [AH21] and the strength becomes more for hard HNP instances with the dimension of the lattice basis more than 100. To further reduce the running time, we could also parallelize this algorithm because each enumeration of vectors is independent.

**Algorithm 3** Linear Predicate Algorithm

**Input:** A $d$-dimensional vector $\boldsymbol{v}$, embedding factor $\tau$, modulus bit-length $n$, number of nonce leakage $l$. HNP samples $(a_i, t_i)$

**Output:** True or False

**if** $|\boldsymbol{v}_{d-1}| \neq \tau$ **then**
  **return** False;
**else if** $\boldsymbol{v}_{d-1} = -\tau$ **then**
  Compute the candidate $\alpha = t_0^{-1} \cdot (\boldsymbol{v}_{d-2} + q/2^{l+1} + a_0) \mod q$;
  **for** $i = 1; i < d; i++$ **do**
    **if** $a_i + (\boldsymbol{v}_{i-1} + q/2^{l+1}) \neq t_i \cdot \alpha \mod q$ **then**
      **return** False;
    **end if**
  **end for**
**else if** $\boldsymbol{v}_{d-1} = \tau$ **then**
  Compute the candidate $\alpha = t_0^{-1} \cdot (q/2^{l+1} - \boldsymbol{v}_{d-2} + a_0) \mod q$;
  **for** $i = 1; i < d; i++$ **do**
    **if** $a_i - (\boldsymbol{v}_{i-1} - q/2^{l+1}) \neq t_i \cdot \alpha \mod q$ **then**
      **return** False;
    **end if**
  **end for**
**end if**
**return** True

## 4.2 Combined with D4F

In the original sieving with predicate algorithm proposed by Albrecht and Heninger [AH21], they do a full sieving algorithm on the given lattice $\mathcal{L}(\boldsymbol{B})$ and then call the predicate algorithm on the output list L (Algorithm 2). As mentioned in Subsection 3.2.3, it is conflict with the dimensions for free technique which aims to find the target vector in a lower dimension of sieving. We propose a improved sieving with predicate algorithm (Algorithm 4) that can combine with dimensions for free technique as well as the progressive lattice sieving technique [LM18].

Firstly, we start with a low dimension project lattice $\mathcal{L}_{[i,d]}$(e.g. d-i=50) and run a sieving algorithm on $\mathcal{L}_{[i,d]}$ until we get many short lattice vectors in $\mathcal{L}_{[i,d]}$ and call our predicate algorithm. If the algorithm does not stop (i.e., does not find the target lattice vector in predicate algorithm), we then increase the dimension of the project lattice by reduce $i$ and repeat this process until $i = 0$ (i.e., reaching the original lattice). Compared to [AH21], we call the predicate algorithm each time after we do a subsieving and thus increase the success probability of the attack since there are more chances to check the target lattice vector. Thanks to our linear predicate algorithm introduced in the last subsection, it is still efficient when calling the predicate algorithm multiple times . Besides, we could benefit from both sieving with predicate and D4F technique that we could recover target lattice vector even it is not the shortest one and wish for an early stop.

**Algorithm 4** Improved Sieving with Predicate

**Input:** A $d$ dimension lattice basis $\boldsymbol{B}$, linear predicate $f(\cdot)$.
**Output:** $\boldsymbol{v}$ such that $\|\boldsymbol{v}\| \leq \sqrt{4/3} \cdot \mathrm{gh}(\Lambda(\mathrm{B}))$ and $f(\boldsymbol{v}) = 1$ or $\perp$

```
for i = d − 50; i > 0; i − − do
  Run a sieving algorithm S on L_[i,d] and denote output list as L;
  for v ∈ L do
    if f(v) = 1 then
      return v
    end if
  end for
end for
return ⊥;
```

### 4.3 Optimal Embedding Factor

In [SETA22], Sun et al. observed that lattice attacks on (EC)DSA are very sensitive to the Kannan's embedding factor, they gave a simple analysis for HNP lattice and an explanation of why embedding factor cannot be too large (e.g., $\tau = q^2$) nor too small (e.g., $\tau = 1$). However, they only simply discussed these two extremely cases and did not gave a explicit way on how to choose the optimal embedding factor. Previous works set the embedding factor to be the upper bound of the target lattice vector's coordinate by default. We give a theoretical analysis on how to choose the optimal embedding factor for HNP lattice.

Recall that in our HNP lattice, the target vector $\boldsymbol{v} = (v_0, \ldots, v_{m-1}, -\tau)$ and the ratio $r = \|v\|/\mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$ determine the samples we needed and success probability. For a given instance with a fixed number of samples $m$, we wish to choose the embedding factor such that the ratio $r$ is as small as possible. Specifically, we have

$$\begin{aligned} r &= \|v\|/\mathrm{gh}(\mathcal{L}(\boldsymbol{B})) \\ &= \frac{\sqrt{\sum_{i=0}^{m-1} v_i^2 + \tau^2}}{\frac{\Gamma(1+(m+1)/2)^{1/(m+1)}}{\sqrt{\pi}} \cdot (q^{m-1} \cdot \tau)^{1/(m+1)}} \\ &= \frac{1}{\frac{\Gamma(1+(m+1)/2)^{1/(m+1)}}{\sqrt{\pi}} \cdot (q^{m-1})^{1/(m+1)}} \cdot \sqrt{\frac{\sum_{i=0}^{m-1} v_i^2 + \tau^2}{\tau^{2/(m+1)}}} \end{aligned} \tag{11}$$

Differentiate $r$ with respect to $\tau$, and then set the resulting expression to be equal to 0, we obtain that ratio $r$ become lowest when $\tau = \sqrt{\frac{\sum_{i=0}^{m-1} v_i^2}{m}}$. As $v_i$ are heuristic independently and identically distributed uniformly in $[-w, w-1]$, we choose the optimal embedding factor to be the expected value $\mathbb{E}(\tau) \approx w/\sqrt{3} = \lfloor q/2^{l+1} \rfloor/\sqrt{3}$.

For example, as for the HNP(160,2) with 91 samples, we randomly generate $v_i$ and show the relationship between the ratio $r = \|v\|/\mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$ and $\tau/w$ in **Fig.1**.

## 5 Experimental Results

We show the experimental results to further illustrate our improvements of sieving with predicate algorithm for key recovery attack on (EC)DSA with nonce leakage. Table 4 shows the details of the machines used for our experiments. We choose the machine depending on the lattice sieving dimension $d$, when $d < 100$ we use $M_1$ together with G6K library [G6Ka] and when $d \geq 100$ we use $M_2$ together with G6K-Tensor library [G6Kb].

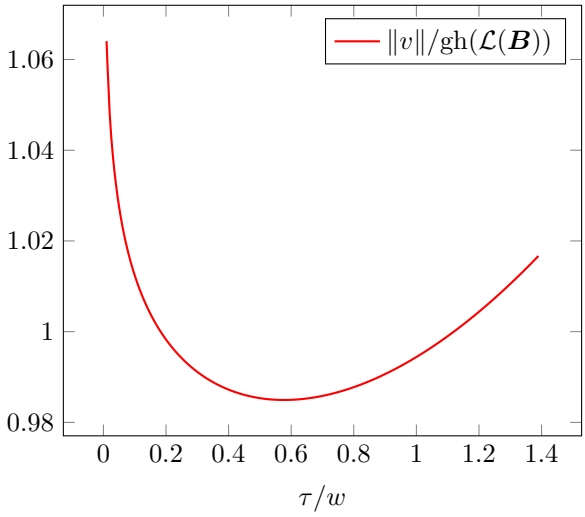


**Figure 1:** The relationship between ratio $r$ and embedding factor.

**Table 4:** Details of the machines used for our experiments.

| Machine | CPU | base freq. | cores | threads | RAM | GPU |
|---|---|---|---|---|---|---|
| $M_1$ | Intel Core i7-8700 | 3.2Ghz | 6 | 12 | 16GB | - |
| $M_2$ | AMD EPYC 7402 | 2.8Ghz | 24 | 48 | 256GB | GeForce RTX 3090 |

## 5.1 Compared with Existing Works

We first conduct experiments on the least number of signature samples needed for successfully recovering the secret key. Table 5 shows that our improved sieving with predicate algorithm outperforms existing works on solving HNP instances for common ECDSA parameters in terms of success probability, number of signature samples and running time.

For example, as for 2 bits nonce leakage on a 160-bit curve which is regarded as a "borderline" case in [SETA22], the authors report applying BKZ-30 together with guessing 15 bits of secret key, they can tackle with 160-bit (EC)DSA with 2-bit nonce leakage for 6% success probability in 10200 seconds on a 32 cores machine. The success probability can be increased if guessing more bits, but it also takes more time correspondingly. [LN13]

**Table 5:** Performance of the attacks using least samples for different instances

| $\log(q)$ | Leakage | Samples | Time | $s/r$ | Previous records |
|---|---|---|---|---|---|
| 160 | 3 bits | 53 | 3s | 54% | $m = 53$, s/r $= 44\%$, 3452s in [AH21] |
| 160 | 2 bits | 82 | 259s | 66% | $m = 90$, s/r $= 6\%$, 10200s in [SETA22] |
| 192 | 3 bits | 65 | 14s | 81% | $m = 63$, s/r $= 56\%$, 851s in [AH21] |
| 256 | 4 bits | 65 | 15s | 72% | $m = 65$, s/r $= 66\%$, 76s in [AH21] |
| 256 | 3 bits | 86 | 924s | 63% | $m = 87$, s/r $= 63\%$, 5400s in [AH21] |
| 384 | 5 bits | 78 | 103s | 81% | $m = 78$, s/r $= 91\%$, 412s in [AH21] |
| 384 | 4 bits | 97 | 11153s | 72% | $m = 97$, s/r $= 88\%$, 49200s in [AH21] |

We compare our experimental results to the most recent records for different instances. Our experiments are all conducted with a PC (e.g., Machine $M_1$ in Table 4 ) using single thread and each instance is taken over 32 experiments.

reports the running time of a few hours with 23% success probability by enumeration with linear prune. Recently, [AH21] solved the same parameters with 63% probability using 87 samples in 4311 CPU-seconds by sieving with predicate. Our experiments show that we can solve this instance in several minutes on a personal computer using a single core with a 66% success probability for only 82 samples.

We can further improve the success probability by using more samples which can reduce the ratio between the length of target vector and the Gaussian heuristic of the lattice basis. Figure 2 shows how success probability varies with the number of signature samples for different instances. The running time can also be reduced by parallelizing the algorithm which can be simply done by adding more threads when using the implementation of G6K library.

We remark that our techniques reduce the running time greatly upon on the implementation of sieving algorithms using G6K together with our linear predicate algorithm. The success probability gains a lot using our improved sieving with predicate algorithm since there are more chances to find and recover target lattice vector from the database containing short lattice vectors after each subsieving. Choosing the optimal embedding factor also does some help.

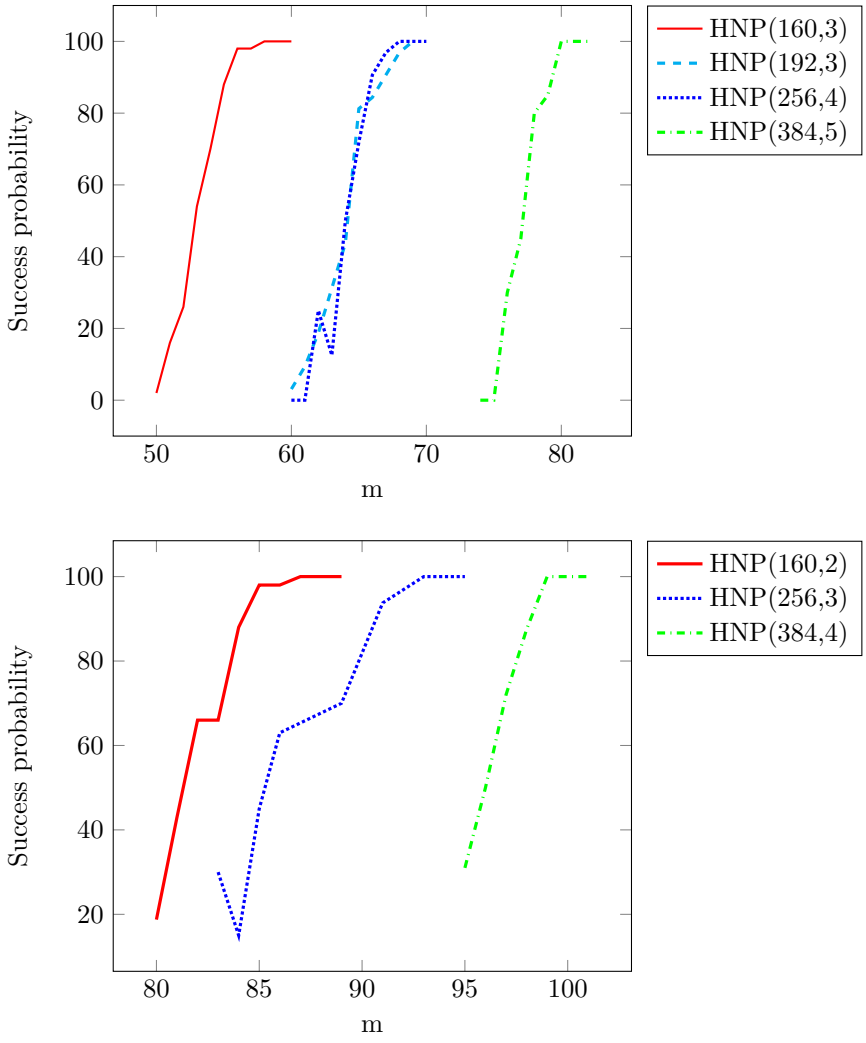


**Figure 2:** Success rates for the generated HNP instances for common ECDSA parameters.

### 5.2 New Instances Record

We also give the first implementation of lattice attacks against ECDSA with 2 bits known for a 256 bit modulus and 3 bits known for a 384 bit modulus which have been considered infeasible in terms of lattice attacks for a long time. Our implementation is based on the GPU version of G6K proposed in [DSvW21]. The details are presented in Table 6.

**Table 6:** Details of solving HNP(384,3) and HNP(256,2).

| $\log(q)$ | Leakage | Samples | MSD | Walltime | Mem GiB | Machine |
|---|---|---|---|---|---|---|
| 384 | 3 | 140 | 117 | 9371s | 28G | $M_2$ with 1 RTX 3090 |
| 256 | 2 | 146 | 129 | 27976s | 150G | $M_2$ with 4 RTX 3090 |

MSD = maximum sieving dimension in practice

### 5.3 Towards Single-bit Nonce Leakage

It has long been believed that lattice attacks against ECDSA with just a single bit of nonce leak are impossible. The main reason is that the target lattice vector is no longer the shortest or closest vector even if it is under the Gaussian heuristic. In such case, there will be some trivial short lattice vectors (e.g., $(q, 0, \ldots, 0), \ldots, (0, \ldots, q, 0, 0)$) which is shorter than the target vector $\boldsymbol{v}$ as well as Gaussian heuristic $\mathrm{gh}(\mathcal{L}(\boldsymbol{B}))$. Lattice reduction algorithms will fail to recover the target lattice vector and always return a lattice basis containing the trivial lattice vectors.

Our improved sieving with predicate algorithm can handle this issue by utilising the fact that sieving algorithms not only return the shortest lattice vector, but also output a database containing all lattice vectors shorter than $\sqrt{4/3} \cdot \mathrm{gh}(\Lambda)$. And when the target lattice vector is in the database, our linear predicate algorithm can efficiently find it and retrieve the entire secret key.

We conduct our experiments on secp112r1 with single-bit nonce leakage and successfully recover the whole secret key, see Table 7 for more details. As for the 160-bit curve with 1-bit nonce leakage, we predict that the sieving dimension will be about 165 which is beyond current computing capacity.

**Table 7:** Details of solving HNP(112,1).

| $\log(q)$ | Leakage | Samples | MSD | Walltime | Mem GiB | Machine |
|---|---|---|---|---|---|---|
| 112 | 1 | 115 | 116 | 15603s | 24.4GB | $M_2$ with 1 RTX 3090 |

## Acknowledgement

We are grateful for the helpful comments from the anonymous reviewers. This work was supported by the National Key Research and Development Program of China (No. 2020YFB1805402) and the National Natural Science Foundation of China (Grants No. 61872359, No. 61972393, No. 61936008 and No. 62172427).

## References

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors,

*EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746. Springer, Heidelberg, May 2019.

[AFG14a] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 2013*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[AFG+14b] Diego F. Aranha, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 262–281. Springer, Heidelberg, December 2014.

[AH21] Martin R. Albrecht and Nadia Heninger. On bounded distance decoding with predicate: Breaking the "lattice barrier" for the hidden number problem. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 528–558. Springer, Heidelberg, October 2021.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. An overview of the sieve algorithm for the shortest lattice vector problem. In Joseph H. Silverman, editor, *CaLC 2001,*, volume 2146 of *LNCS*, pages 1–3. Springer, Heidelberg, March 2001.

[ANT+20] Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. Ladderleak: Breaking ECDSA with less than one bit of nonce leakage. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 225–242. ACM Press, November 2020.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Comb.*, 6(1):1–13, 1986.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *ACM-SIAM SODA 2016*, pages 10–24. SIAM, January 2016.

[Ble00] Daniel Bleichenbacher. On the generation of one-time keys in DL signature schemes. Presentation at IEEE P1363 Working Group meeting, 2000.

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *IACR Cryptol. ePrint Arch.*, page 713, 2016.

[BV96] Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In Neal Koblitz, editor, *CRYPTO 1996*, volume 1109 of *LNCS*, pages 129–142. Springer, Heidelberg, August 1996.

[DSvW21] Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. Advanced lattice sieving on GPUs, with tensor cores. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 249–279. Springer, Heidelberg, October 2021.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145. Springer, Heidelberg, April 2018.

[G6Ka] The g6k development team: G6k. Available at https://github.com/fplll/g6k.

[G6Kb] The g6k-gpu-tensor development team: G6k-gpu-tensor. Available at https://github.com/WvanWoerden/G6K-GPU-Tensor.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, Heidelberg, June 2010.

[HK17] Gottfried Herold and Elena Kirshanova. Improved algorithms for the approximate k-list problem in Euclidean Norm. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 16–40. Springer, Heidelberg, March 2017.

[HKL18] Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. Speed-ups and time-memory trade-offs for tuple lattice sieving. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 407–436. Springer, Heidelberg, March 2018.

[JSSS20] Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sýs. Minerva: The curse of ECDSA nonces systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces. *IACR TCHES*, 2020(4):281–308, 2020.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *ACM STOC*, pages 193–206. ACM, April 1983.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[LdW15] Thijs Laarhoven and Benne de Weger. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 101–118. Springer, Heidelberg, August 2015.

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009.

[LM18] Thijs Laarhoven and Artur Mariano. Progressive lattice sieving. In Tanja Lange and Rainer Steinwandt, editors, *PQCrypto 2018*, volume 10786 of *LNCS*, pages 292–311. Springer, Heidelberg, April 2018.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, Heidelberg, February/March 2013.

[MHMP13] Elke De Mulder, Michael Hutter, Mark E. Marson, and Peter Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 435–452. Springer, Heidelberg, August 2013.

[MSEH20] Daniel Moghimi, Berk Sunar, Thomas Eisenbarth, and Nadia Heninger. TPM-FAIL: TPM meets timing and lattice attacks. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2057–2073. USENIX Association, August 2020.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In Moses Charikar, editor, *ACM-SIAM SODA 2010*, pages 1468–1480. SIAM, January 2010.

[NS02] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptol.*, 15(3):151–176, 2002.

[NS03] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptogr.*, 30(2):201–217, 2003.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.*, 2(2):181–207, 2008.

[Rya19] Keegan Ryan. Return of the hidden number problem. A widespread and novel key extraction attack on ECDSA and DSA. *IACR TCHES*, 2019(1):146–168, 2019.

[SETA22] Chao Sun, Thomas Espitau, Mehdi Tibouchi, and Masayuki Abe. Guessing bits: Improved lattice attacks on (EC)DSA with nonce leakage. *IACR TCHES*, 2022(1):391–413, 2022.

[Tib17] Mehdi Tibouchi. Attacks on (ec)dsa with biased nonces. 2017.

[TTA18] Akira Takahashi, Mehdi Tibouchi, and Masayuki Abe. New Bleichenbacher records: Fault attacks on qdsa signatures. *IACR TCHES*, 2018(3):331–371, 2018.

[WLTB11] Xiaoyun Wang, Mingjie Liu, Chengliang Tian, and Jingguo Bi. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 2011*, pages 1–9. ACM, March 2011.

[WSBS20] Samuel Weiser, David Schrammel, Lukas Bodner, and Raphael Spreitzer. Big numbers - big troubles: Systematically analyzing nonce leakage in (EC)DSA implementations. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 1767–1784. USENIX Association, August 2020.

[ZPH13] Feng Zhang, Yanbin Pan, and Gengran Hu. A three-level sieve algorithm for the shortest vector problem. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 29–47. Springer, Heidelberg, August 2013.