Guessing Bits: Improved Lattice Attacks on (EC)DSA

Chao Sun[1], Thomas Espitau[2], Mehdi Tibouchi[1,2] and Masayuki Abe[1,2]

[1] Kyoto University, Kyoto, Japan
[2] NTT Secure Platform Laboratories, Tokyo, Japan

Abstract. In the past 30 years, lattice reduction has proved to be one powerful tool of public-key cryptanalysis. Since the advent of the Hidden Number Problem, there has been an extensive study on attacks on (EC)DSA with nonce leakage. While lattice attacks require only a few signatures, it can't deal with small nonce bias compared with Bleichenbacher attack. Prior to this work, it is unknown how to utilize more signatures to improve lattice attacks on (EC)DSA.

In this paper, we propose several approaches to improve lattice attacks. The key idea is that we can guess some bits of the secret key(or the nonces) and modify the standard lattice to increase the volume, thus making the lattice attack much easier. Besides, we observe that by filtering some specific signatures we are able to modify the lattice, so we can collect a large number of signatures and construct a lattice that is much easier to attack. With a combination of these techniques, we are able to improve lattice attacks on (EC)DSA. On the one hand, we are able to attack 160-bit modulus(and other modulus as well) (EC)DSA with 2-bit leakage within 2^{15} BKZ-30 operations with 90 signatures. On the other hand, with 2^{27} signatures available, we are able to attack 160-bit (EC)DSA with 2-bit leakage in just one BKZ-30 operation.

As a second contribution, we give an explanation for several questions unexplained in previous works. It was observed that SVP approaches(Kannan embedding) always outperform CVP approaches(nearest plane) and lattice attack is very sensitive to the Kannan Embedding factor, but these questions are not discussed in previous works. We give an explanation for completeness.

Last, we carry out some experiments on the TPM-Fail dataset. While the original attack utilizes around 40000 signatures, with a combination of our method, we are able to recover the secret with only 800 signatures available.

Keywords: HNP · Lattice Reduction · (EC)DSA · Embedding Method · Guessing Bits

1 Introduction

A lattice is a discrete group in space, which can also be interpreted as all integer combinations of a certain set of vectors $\mathbf{b}_1, \cdots, \mathbf{b}_d$ known as bases. One lattice has infinitely many bases, but bases with small norms and orthogonal properties are much more interesting. Finding such kind of bases is a mathematical problem with a long history, which dates back to the work of mathematicians in the 18th century. However, it was not until 1982 that Lenstra, Lenstra, Lovász[LLL82] introduced a polynomial time lattice reduction algorithm that became known as LLL algorithm. Since the advent of LLL, it has proved to be one powerful cryptanalytic tool: it was used to attack the knapsack-based cryptosystem[Sha82] and find small roots for polynomial equations[Cop97](known as Coppersmith's method).

DSA and ECDSA are well established standards for digital signature based on the discrete logarithm problem. It is well known that if the same nonce k is used twice, the

adversary can directly compute the private key due to the linear relation between the nonce and private key. Even worse, partial information about the nonces can lead to recovery of the full private key, either via lattice reduction technique or Fourier analysis technique (called Bleichenbacher's attack). In this paper, we mainly focus on the lattice approach. For Fourier analysis approach, we refer to [Ble00], [AFG+14b], [DHMP13], [TTA18], [ANT+20].

In 1996, the Hidden Number Problem(HNP for short) was originally proposed by Boneh and Venkatesan[BV96] to prove the bit security of Diffie-Hellman key exchange. They transformed the HNP into a Closest Vector Problem instance and solved it by LLL reduction and Babai's nearest plane algorithm[Bab86]. After that, Howgrave-Graham and Smart[HGS01], Shparlinski and Nguyen[NS02] used the HNP to attack (EC)DSA if some bits of the nonces are known. However, when nonce leakage is very small, such as 1 or 2, the attack becomes much more difficult mainly because the hidden lattice vector is not very close to the target vector. With the development of lattice reduction algorithm, Chen and Nguyen[CN11] invented the so-called BKZ 2.0 that made several improvements on the original BKZ[SE94]. In 2013, using BKZ 2.0 combined with pruning techniques for BDD enumeration, Liu and Nguyen[LN13] were able to attack 160-bit DSA with 2-bit nonce leakage in a few hours on a workstation. In a very recent work[AH] by Albrecht and Heninger, they proposed new ways of improving lattice attacks and made new records.

1.1 Our Contribution

We give the following contributions:

- We propose several approaches to improve lattice attacks on EC(DSA).
  - Approach 1: in standard lattice attacks, either we find the secret key or get nothing. Even if we know some information about the secret key, standard lattice attacks do not utilize it. By guessing some bits of the secret key and modifying the standard lattice, we are able to construct a lattice that is much easier to attack. With this approach, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage(and other modulus as well) within 2^{15} BKZ-30 operations on a 90-dimensional lattice.
  - Approach 2: in order to make the hidden lattice vector more close to the target vector, we can also guess some bits of the nonces for some signatures, thus constructing a hybrid lattice. With this approach, similar result with approach 1 could be achieved.
  - Approach 3: motivated by the Bleichenbacher attack, we are able to filter some signatures and construct HNP instances with smaller value. By utilizing such kind of information, we are able to construct a lattice that is much easier to attack. For instance, with 2^{27} signatures available, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage within one BKZ-30 operation on a 90-dimensional lattice.
- With these approaches, we are able to quantify the computation cost even for 1-bit nonce leakage case despite the fact that it still remains difficult in practice. Besides, our approaches are quite general and actually work for any HNP instance. Furthermore, our approaches are parallelizable, so the attack can be carried out in a short time if given a reasonable number of cores.
- There are some phenomenons reported in previous works, such as the fact that SVP approaches(Kannan Embedding) always outperform CVP approaches(Nearest Plane), the fact that lattice attack is very sensitive to the Kannan embedding factor.

These questions remain unexplained in previous works, so we give an explanation for completeness.

- We carried out experiments on the TPM-FAIL dataset[MSEH20] and apply our approaches to key recovery. While the original attack requires about 40000 signatures, with the method of guessing bits, we are able to recover the secret key with only 800 signatures available.

Although we come up with a way of improving lattice attacks with more signatures, it still requires too many signatures compared with Bleichenbacher Attack. For instance, for 160-bit (EC)DSA with 2-bit nonce leakage, our method requires 2^{27} signatures, while the Bleichenbacher attack requires about 2^{15} signatures for 2-bit leakage case and 2^{27} signatures for the one-bit leakage case[ANT+20]. However, the fact that there exists a way of improving lattice attacks with more signatures might give some ideas for future work and we hope that lattice attacks on (EC)DSA could be further improved.

2 Preliminaries

2.1 Lattices

In our context, we only consider integral lattices. Let $\mathbf{b}_1, \cdots, \mathbf{b}_m$ be arbitrary vectors in $\mathbb{Z}^n$. Denote by $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_m)$ the set of all integral linear combinations of the vectors $\mathbf{b}_1, \cdots, \mathbf{b}_m$:

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

We call $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_m)$ the lattice generated by $\mathbf{b}_1, \cdots, \mathbf{b}_m$. If $\mathbf{b}_1, \cdots, \mathbf{b}_m$ are linearly independent, we say that $\mathbf{b}_1, \cdots, \mathbf{b}_m$ is a basis of this lattice. In this paper, we mainly focus on full-rank lattice of dimension n.

The Euclidean norm of the shortest non-zero vector in $\mathcal{L}$ is called the first minimum of $\mathcal{L}$ and denoted as $\lambda_1(\mathcal{L})$. For $1 \leq i \leq n$, the i-th minimum $\lambda_i(\mathcal{L})$ is defined as the minimum radius r such that a ball centered at origin with radius r contains i linearly independent vectors.

It is proved in [Ajt06] that a random n-dimensional lattice satisfies asymptotically with overwhelming probability

$$\forall 1 \leq i \leq n, \lambda_i(\mathcal{L}) \approx \sqrt{\frac{n}{2\pi e}} \mathrm{vol}(\mathcal{L})^{1/n}$$

There are many computational problems related to lattices. The most famous one is the shortest vector problem(SVP for short): given a lattice $\mathcal{L}$, find the shortest vector $v \in L$ such that the Euclidean norm $|v| = \lambda_1(\mathcal{L})$. Another problem is the closest vector problem(CVP for short): given a lattice $\mathcal{L}$ and a target vector t, find the vector $v \in L$ such that $|v - t|$ is minimal.

There exists efficient lattice algorithms for solving approximate versions of SVP and CVP. Algorithms such as LLL[LLL82], BKZ[SE94] output lattice basis $[b_1, \cdots, b_n]$ such that the approximation factor $\frac{|b_1|}{\lambda_1(\mathcal{L})}$ and the hermite factor $\frac{|b_1|}{\mathrm{vol}(\mathcal{L})^{1/n}}$ are small.

2.2 Hidden Number Problem

The Hidden Number Problem can be described as follows: q and l are fixed numbers, for many known random t, we have an oracle $\mathcal{O}_\alpha(t)$ that on input t computes the l most(or least) significant bits of $(\alpha t \bmod q)$, the goal is to recover the hidden secret α. In this

paper, we only consider the oracle that outputs the LSB, but for MSB, it is essentially the same despite the fact that the result might have slight difference. Suppose that we have queried the oracle d times and have d pairs $(t_i, u_i)(i = 1, 2, \cdots, d)$ where $u_i = \mathcal{O}_\alpha(t_i)$, since u_i is the l least significant bits of αt_i, we have $|\alpha t_i - u_i|_q < q/2^l$, where $|z|_q$ means the unique integer $0 \leq x < q$ such that $x \equiv z \bmod q$. Boneh and Venkatesan proposed a way to transform this into a lattice problem. Construct a lattice $\mathcal{L}$ spanned by the following matrix B:

$$B = \begin{pmatrix} 2^l q & 0 & \cdots & 0 & 0 \\ 0 & 2^l q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^l q & 0 \\ 2^l t_1 & 2^l t_2 & \cdots & 2^l t_d & 1 \end{pmatrix}$$

Since $|\alpha t_i - u_i|_q < q/2^l$, there exists some integer c_i such that $|\alpha t_i - u_i + c_i q| < q/2^l$, so $|2^l \alpha t_i - 2^l u_i + 2^l c_i q| < q$, and $(2^l \alpha t_1 + c_1 2^l q, 2^l \alpha t_2 + c_2 2^l q, \cdots, 2^l \alpha t_d + c_d 2^l q, \alpha)$ is a lattice vector(which we call hidden lattice vector) in $\mathcal{L}$, and set the target vector $\mathbf{v} = (2^l u_1, 2^l u_2, \cdots, 2^l u_d, 0)$. The distance between the target vector $\mathbf{v}$ and the lattice $\mathcal{L}$ is at most $q\sqrt{d+1}$. Therefore, when l is not too small, the target vector $\mathbf{v}$ is a close vector to the lattice $\mathcal{L}$, so this becomes a CVP instance(or more precisely, BDD instance). Generally, there are two ways to solve the HNP, the CVP approaches and SVP approaches. In the original paper by Boneh and Venkatesan, they used LLL algorithm to reduce the lattice basis and Babai's nearest plane algorithm to find the hidden lattice vector. The LLL reduction can be replaced with BKZ. One can also use CVP enumeration instead of nearest plane algorithm. Besides, another technique, known as Kannan embedding method[Kan87], transforms the CVP instance into a Shortest Vector Problem by embedding the target point into the original lattice, thus constructing a larger lattice:

$$C = \begin{pmatrix} B & 0 \\ \mathbf{v} & q \end{pmatrix}$$

Then, one can solve SVP by lattice reduction. In this paper, we mainly use the Kannan embedding method to solve HNP.

2.3 DSA Signature Scheme

DSA is an El Gamal-like signature scheme, which is included in Digital Signature Standard(DSS) issued by NIST. DSA can be described as follows.

2.3.1 Parameters

The parameters are p, q, g, where p and q are primes satisfying $q|(p-1)$, $g \in \mathbb{Z}_p^*$ has order q. Besides, we have a hash function h that maps any arbitrary-length string into $\mathbb{Z}_q$. The signing key α is a uniformly random number in $\mathbb{Z}_q^*$ and the public key is $y = g^\alpha \bmod p$.

2.3.2 Signing Phase

To sign a message m, the nonce k is chosen uniformly at random from $\mathbb{Z}_q$, and we compute $r = (g^k \bmod p) \bmod q$, and $s = k^{-1}(h(m) + \alpha r) \bmod q$, the signature is the pair (r, s).

2.3.3 Verification Phase

Given a signature pair(r, s) of the message m, if $r = (g^{h(m)s^{-1}} y^{h(m)s^{-1}} \bmod p) \bmod q$, the signature is regarded as valid, otherwise invalid.

2.4 Lattice Attacks on DSA

From the signing phase of DSA, we have already known that $s \equiv k^{-1}(h(m) + \alpha r) \bmod q$, so

$$\alpha r \equiv sk - h(m) \bmod q$$

Now in our case, we have l-bit leakage, so the l least significant bits of k are zero(actually, having l-bit leakage means knowing the value of l least significant bits, but it is essentially the same for the attack), $k = 2^l b$ for some integer $b \geq 0$, so we have

$$\alpha r \equiv s2^l b - h(m) \bmod q$$
$$\Leftrightarrow \alpha 2^{-l} s^{-1} r \equiv b - 2^{-l} s^{-1} h(m) \bmod q$$

Now set

$$t = 2^{-l} s^{-1} r \bmod q$$
$$\text{and } u = -2^{-l} s^{-1} h(m) \bmod q$$
$$\text{so } b \equiv \alpha t - u \bmod q$$

Note that both t and u can be computed from all the public available information. Since $0 \leq b < q/2^l$,

$$|\alpha t - u|_q < q/2^l$$

In this way, we have constructed a HNP instance for DSA. Then we solve the HNP either by nearest plane algorithm or Kannan embedding method.

2.5 Recentering Technique

In order to further improve the lattice attack on (EC)DSA, there is a well-known technique in the community called recentering. It works as follows: since

$$|\alpha t - u|_q < q/2^l$$

there exists some integer c such that

$$0 \leq \alpha t - u + cq < q/2^l$$
$$\Leftrightarrow -q/2^{l+1} \leq \alpha t - u - q/2^{l+1} + cq < q/2^{l+1}$$

Therefore,

$$|\alpha t - u - q/2^{l+1}|_q \leq q/2^{l+1}$$

Now let

$$v = 2^{l+1} u + q$$

we have

$$|\alpha t - v/2^{l+1}|_q \leq q/2^{l+1}$$

Suppose that now we have d signatures $(r_i, s_i)(i = 1, \cdots, d)$ and it is not difficult to compute the pairs (t_i, u_i) where t_i and u_i are defined above. Then construct a lattice $\mathcal{L}$ spanned by the following matrix B:

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 \end{pmatrix}$$

For each inequality, there exists some integer c_i such that

$$|\alpha 2^{l+1} t_i - v_i + c_i 2^{l+1} q| < q$$

Let the target point $\mathbf{v} = (v_1, \cdots, v_d, 0)$ and the hidden lattice point $\mathbf{u} = (\alpha 2^{l+1} t_1 + c_1 2^{l+1} q, \cdots, \alpha 2^{l+1} t_d + c_d 2^{l+1} q, \alpha)$, Thus the Euclidean distance between the target point $\mathbf{v}$ and the hidden lattice point $\mathbf{u}$ is bounded by $q\sqrt{d+1}$. If we can find the hidden lattice point, then we attack DSA successfully, since the last coefficient of $\mathbf{u}$ is the private key α. The volume of this lattice $\mathcal{L}$ is $q^d 2^{(l+1)d}$, according to Gaussian Heuristics, the Euclidean norm of the shortest vector is roughly

$$\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d+1}{2\pi e}} (\mathrm{vol})^{\frac{1}{d+1}} \approx \sqrt{\frac{d+1}{2\pi e}} 2^{\frac{(l+1)d}{d+1}} q^{\frac{d}{d+1}}$$

Therefore, the requirement is that the distance is much smaller than $\lambda_1(\mathcal{L})$:

$$q\sqrt{d+1} < \sqrt{\frac{d+1}{2\pi e}} 2^{\frac{(l+1)d}{d+1}} q^{\frac{d}{d+1}}$$

After solving this inequality, we get

$$d \geq \frac{\log_2(q)}{l - \log_2(\sqrt{\pi e/2})}$$

This can be used to estimate the number of signatures needed for the attack to succeed.

2.6 Difficulty When Leakage is Small

First, we give an explanation of why lattice attack against (EC)DSA with small nonce leakage is difficult. Now we are in the context of 160-bit modulus. With this formula

$$d \geq \frac{\log_2(q)}{l - \log_2(\sqrt{\pi e/2})}$$

and performance in practical experiments, the following table is the typical number of signatures(the number can be decreased a little bit) needed to perform the lattice attack on 160-bit (EC)DSA.

leakage: l	number of signatures: d
4	50
3	80
2	100
1	200

When the leakage is 3, $d = 80$, $l = 3$, and the lattice basis matrix B would be

$$B = \begin{pmatrix} 16q & 0 & \cdots & 0 & 0 \\ 0 & 16q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 16q & 0 \\ 16t_1 & 16t_2 & \cdots & 16t_d & 1 \end{pmatrix}$$

The Euclidean norm of the first vector is $16q$, the Euclidean distance between the hidden lattice vector $\mathbf{u}$ and the target vector $\mathbf{v}$ is upper-bounded by $q\sqrt{d+1} = 9q$. Therefore, any linear combination of the first d rows will have significantly larger Euclidean norm than the distance between the hidden lattice vector and the target vector.

When the leakage is 2, $d = 100$, $l = 2$, and the lattice basis matrix B is

$$B = \begin{pmatrix} 8q & 0 & \cdots & 0 & 0 \\ 0 & 8q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 8q & 0 \\ 8t_1 & 8t_2 & \cdots & 8t_d & 1 \end{pmatrix}$$

The Euclidean norm of the first vector is $8q$, the Euclidean distance between the hidden lattice vector $\mathbf{u}$ and the target vector $\mathbf{v}$ is upper-bounded by $q\sqrt{d+1} \approx 10q$. To be a bit more precise, we can compute the expected distance. Each coefficient of the difference vector between the hidden vector and target vector is uniformly distributed in $\mathbb{Z}_q$, thus the expected norm for one coefficient is

$$\sqrt{\frac{1}{q}\sum_{i=0}^{i=q-1} i^2} \approx \sqrt{\frac{q^2}{3}}$$

Thus the expected distance is roughly

$$\sqrt{\frac{100}{3}q^2} \approx 6q$$

When the leakage is 1, $d = 200$, $l = 1$, and the lattice basis matrix B is

$$B = \begin{pmatrix} 4q & 0 & \cdots & 0 & 0 \\ 0 & 4q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 4q & 0 \\ 4t_1 & 4t_2 & \cdots & 4t_d & 1 \end{pmatrix}$$

The Euclidean norm of the first vector is $4q$, the Euclidean distance between the hidden lattice vector $\mathbf{u}$ and the target vector $\mathbf{v}$ is upper-bounded by $q\sqrt{d+1} \approx 14q$. With similar computation, we can know that the expected distance is around $8q$. This means that many linear combinations of the first d rows will have smaller Euclidean norm than the difference vector. In other words, there are exponentially many lattice vectors that is more close to the target vector than the hidden vector, thus making decoding extremely difficult.

2.7 Projected Lattice

Typically, in the standard lattice attacks, we almost always locate the secret key in the second row(which we hope to be the first row) of the reduced basis. In order to deal with this issue, [AH] made a modification to the original lattice. Recall that the matrix that we construct is

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 \end{pmatrix}$$

With some linear combinations of the rows, we could know that $(0, 0, \cdots, 0, q)$ belongs to this lattice. The expected Euclidean distance of difference vector between the target vector and hidden vector is roughly $\sqrt{\frac{d+1}{3}}q$. With typical parameters such as $d = 85, l = 2$,

this Euclidean distance is much larger than q. This means that the difference vector will never be the shortest vector in practice. In fact, we can project this lattice orthogonal to $(0, \cdots, 0, q)$ and construct a new lattice

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1(t_d)^{-1} & 2^{l+1}t_2(t_d)^{-1} & \cdots & 2^{l+1}t_{d-1}(t_d)^{-1} & 2^{l+1} \end{pmatrix}$$

In this new lattice, the hidden vector will be $(|\alpha t_d|_q 2^{l+1}t_1(t_d)^{-1} + c_1 2^{l+1}q, |\alpha t_d|_q 2^{l+1}t_2(t_d)^{-1} + c_2 2^{l+1}q \cdots, |\alpha t_d|_q 2^{l+1}t_{d-1}(t_d)^{-1} + c_d 2^{l+1}q, 2^{l+1}|\alpha t_d|_q)$. The important thing is that the vector $(0, 0, \cdots, 0, q)$ does not belong to the new lattice, so we are able to locate the private key in the first row of the reduced basis.

3 Analysis: Modeling Lattice Attacks on (EC)DSA

Following the idea of [AFG14a], we first propose a model that can guide the parameter setting of lattice attacks on (EC)DSA. In [GN08], it was concluded that given a lattice-reduction algorithm which we assume to be characterised by a root-Hermite factor δ_0 and a n-dimensional lattice Λ, the algorithm will be successful in disclosing a shortest non-zero vector with high probability when $\frac{\lambda_2}{\lambda_1} \geq \tau \cdot {\delta_0}^n$, where τ is a constant depending both on the nature of the lattices involved and lattice reduction algorithm that we use. We assume that in order to get the same success probability, the gap $\frac{\lambda_2}{\lambda_1}$ needed keeps the same for this type of lattice(for attacking (EC)DSA) and the same lattice reduction algorithm.

Although this is a somewhat non-standard assumption, the result is consistent with the practical experiments and what we want is an intuitive model(not necessarily very accurate) to guide the parameter setting, so it suffices for our purpose to do this.

3.1 Modeling Lattice Attacks

First we do some experiments to determine the root hermite factor δ_0 for BKZ-30 on this type of lattice:

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 2^{l+1} \end{pmatrix}$$

After doing several experiments, we determined that $\delta_0 \approx 1.01$ for BKZ-30. Then we set the desired success probability level to be 20%. In our assumption, τ only depends on the desired success probability level, the nature of this type of lattice and the lattice reduction algorithm. Since now everything is fixed, we do some experiments to determine the value of τ. We set $l = 3, d = 57$ for 160-bit modulus q, after doing 100000 experiments, we find that when

$$\frac{\lambda_1(\mathcal{L})}{|\mathbf{e}|} \approx 0.94(\mathbf{e} \text{ is the difference vector })$$

we have about 20% success rate, and compute

$$\tau = \frac{0.94}{\delta_0^{58}} \approx 0.527$$

What we want is a rough estimate for 160-bit (EC)DSA with 2-bit nonce leakage. Typically, the number of signatures needed d = 85. So we can compute the probability such that

$$\frac{\lambda_1(\mathcal{L})}{|\mathbf{e}|} \geq \tau \cdot \delta_0^{86}$$

From Gaussian Heuristic,

$$\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d+1}{2\pi e}} \mathrm{vol}(\mathcal{L})^{1/(d+1)}$$

The volume

$$\mathrm{vol}(\mathcal{L}) = 2^{(l+1)(d+1)} \cdot q^d$$

So the requirement is

$$\begin{aligned} |\mathbf{e}| &< \frac{2.243 \cdot 8 \cdot q^{d/(d+1)}}{0.527 \cdot 1.01^{86}} \\ \Leftrightarrow |\mathbf{e}| &< 14.47 q^{d/(d+1)} \end{aligned}$$

Since q is a 160-bit modulus, $q^{\frac{1}{d+1}} \approx (2^{161})^{1/86} \approx 3.66$, so

$$\Leftrightarrow |\mathbf{e}| < 14.47q/3.66 \approx 3.95q \tag{1}$$

Each coefficient $e_i (i = 1, \cdots, d+1)$ of $\mathbf{e}$ is uniformly distributed in $(-q, q)$, e_i^2 is uniformly distributed in $\{0, 1, 4, 9, \cdots, q^2\}$(to be precise, the probability at 0 is different, but this can be ignored). According to some standard results from probability theory in the appendix of [KL20],

Lemma 1. *Let $X_1, \cdots, X_m$ be pairwise-independent random variables with the same expectation μ and variance σ^2. Then for every $\delta > 0$,*

$$Pr[|\frac{\sum_{i=1}^m X_i}{m} - \mu| \geq \delta] \leq \frac{\sigma^2}{\delta^2 m}$$

The expectation of square of Euclidean norm of difference vector $\mathbf{e}$

$$\begin{aligned} E(|\mathbf{e}|^2) &= E(e_1^2 + \cdots + e_{d+1}^2) \\ &= (d+1)E(e_1^2) \\ &= (d+1)\frac{1}{q}\sum_{i=1}^{i=q} i^2 \approx (d+1)\frac{q^2}{3} \end{aligned}$$

$$\text{so } E(|\mathbf{e}|) \approx \sqrt{\frac{d+1}{3}} q$$

The variance $\sigma^2(e_1^2)$ for one coefficient is

$$\begin{aligned} \sigma^2(e_1^2) &\approx \frac{1}{q}\sum_{i=1}^{i=q}(i^2 - \frac{1}{3}q^2)^2 \\ &= \frac{1}{q}\sum_{i=1}^{i=q}(i^4 - \frac{2}{3}q^2 i^2 + \frac{1}{9}q^4) \\ &\approx \frac{1}{q}(\frac{1}{5}q^5 - \frac{2}{3}q^2 \cdot \frac{1}{3}q^3 + \frac{1}{9}q^5) \\ &= \frac{4}{45}q^4 \end{aligned}$$

With the tail bound from Lemma 1, and set $\delta = \frac{1}{7}q^2, d = 85$

$$\Pr[|e_1^2 + \cdots + e_{d+1}^2 - \frac{d+1}{3}q^2| \geq \frac{86}{7}q^2] \leq \frac{4 \cdot 49}{45 \cdot (d+1)}$$

so

$$\Pr[|e_1^2 + \cdots + e_{d+1}^2 - \frac{86}{3}q^2| \geq \frac{86}{7}q^2] \leq 5\%$$

Then with probability at least 95%, $|\mathbf{e}|^2$ will stay in the interval

$$(16.38q^2, 40.95q^2)$$

As previously mentioned in equation 1, in order to gain a 20% success rate, $|\mathbf{e}|$ should be less than 3.95q, so this is unlikely to happen.

3.2 One Intuitive Idea to Improve the Attack

The direct idea is to increase the gap

$$\frac{\lambda_1(\mathcal{L})}{|\mathbf{e}|}$$

Since

$$\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d+1}{2\pi e}}\mathrm{vol}(\mathcal{L})^{1/(d+1)}$$

we could increase the volume of the lattice(which we will explain later) while keeping the $|\mathbf{e}|$ unchanged. Recall that in order to have 20% success rate, we need

$$\frac{\lambda_1(\mathcal{L})}{|\mathbf{e}|} \geq \tau \cdot \delta_0^{86}$$

If we increase the volume of the lattice by 2^{20} times, $\lambda_1(\mathcal{L})$ will increase by $(2^{20})^{1/86} \approx 1.174$ times, so the requirement 1 will become

$$\Leftrightarrow |\mathbf{e}| < 3.95q \cdot 1.174 \approx 4.64q \tag{2}$$

we will have greater success probability.

4 Guessing Bits of Secret Key

In our context, the modulus q has 160 bits, the leakage $l = 2$.(for other modulus, it is quite similar) In standard lattice attacks, either we find the secret key or get nothing. It is somewhat believed that making assumptions on the secret key does not help the attack. However, this is not true. We find that the length of the secret key is closely related to the difficulty of the attack. Take 160-bit (EC)DSA with 2-bit leakage for instance, if we assume that the secret key has less than 60 bits, we can modify the original lattice and make the attack very easy.

Recall that the HNP inequality is $|\alpha t_i - u_i|_q < q/2^l (i = 1, \cdots, d)$ and the lattice we construct is

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 \end{pmatrix}$$

the target vector is $(2^{l+1}u_1+q, 2^{l+1}u_2+q, \cdots, 2^{l+1}u_d+q, 0)$, and the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \cdots \alpha 2^{l+1}t_d + c_d 2^{l+1}q, \alpha)$. As we discussed in the previous section, in order to improve the success rate of lattice attacks, one direct idea is to increase the volume of the lattice while keeping the difference vector between target vector and hidden vector unchanged. For instance, we could modify the lattice as

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 2^{100} \end{pmatrix}$$

In this way, we increase the volume of the lattice by 2^{100} times, but the problem is that hidden lattice vector will not be close to the target vector anymore, because the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \cdots \alpha 2^{l+1}t_d + c_d 2^{l+1}q, 2^{100}\alpha)$, and the difference in the last coefficient is very large$(2^{100}\alpha)$, thus making the modification meaningless.

However, if we assume that the secret key has less than 60 bits, then $2^{100}\alpha$ is still bounded by $2^{160} \approx q$, so this means the distance between the hidden vector and the target vector almost keeps unchanged, and we have increased the volume of lattice by 2^{100} times, thus making the success probability significantly better. We carried out some simulation experiments and find that if the secret key only has 60 bits for 160-bit (EC)DSA with 2-bit nonce leakage, after modifying the lattice as

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 2^{100} \end{pmatrix}$$

we can recover the secret key in just one BKZ-20 operation with 100% success rate, so this becomes almost trivial.

This observation leads to the following attack. First write the secret key in the following format:

$$\alpha = \alpha_1 * 2^c + \alpha_2 (0 \leq \alpha_2 < 2^c)$$

where c is any arbitrary predetermined integer between 1 and 160. So α_1 is the $(160-c)$ most significant bits of α and α_2 is the remaining c bits of α. Suppose that we have constructed d HNP inequalities with leakage l:

$$|\alpha * t_i - u_i|_q < q/2^l \ (i = 1, 2, \cdots d)$$

Then substitute α with $\alpha_1 * 2^c + \alpha_2$

$$|\alpha_1 * 2^c * t_i + \alpha_2 * t_i - u_i|_q < q/2^l (i = 1, 2, \cdots d)$$

and set

$$t'_i = 2^c * t_i$$
$$u'_i = -\alpha_2 * t_i + u_i$$

so we have new HNP inequalities for t'_i and u'_i

$$|\alpha_1 * t'_i - u'_i|_q < q/2^l \ (i = 1, 2, \cdots d)$$

Then construct the lattice as

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t'_1 & 2^{l+1}t'_2 & \cdots & 2^{l+1}t'_d & 2^c \end{pmatrix}$$

The hidden vector is $(\alpha_1 2^{l+1}t'_1 + c_1 2^{l+1}q, \alpha_1 2^{l+1}t'_2 + c_2 2^{l+1}q, \cdots \alpha_1 2^{l+1}t'_d + c_d 2^{l+1}q, \alpha_1 2^c)$ and the target vector is $(2^{l+1}u'_1 + q, 2^{l+1}u'_2 + q, \cdots, 2^{l+1}u'_d + q, 0)$. Now we have increased the volume of the lattice by 2^c times while keeping the Euclidean norm of the difference vector almost unchanged. Of course we don't know the value of α_2, but we can enumerate α_2 from 0 to 2^c, so this is a trade-off: we increase the volume of the lattice by 2^c times(thus making the attack easier) at the cost of 2^c enumerations. We formalize the attack as the following steps:

- Step 1: Determine the integer constant c.(It depends on how much enumeration cost we want to pay)
- Step 2: Collect d signatures and construct t_i, u_i as previously defined.$(i = 1, 2, \cdots, d)$.
- Step 3: Enumerate α_2 from 0 to 2^c:
  - Construct the corresponding HNP instance for α_1.
  - Solve the new HNP instance by Kannan embedding.

With this method, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage, 256-bit (EC)DSA with 3-bit nonce leakage, 384-bit (EC)DSA with 4-bit nonce leakage. See the section of experimental results.

5 Guessing Bits of Nonce: Hybrid Attack

Another similar approach could be done to increase the volume of the lattice. Again in our context, the modulus q has 160 bits, the leakage $l = 2$. For other modulus, it is essentially the same, so we will not discuss it again.

Suppose that now we have d 160-bit (EC)DSA signatures$(r_i, s_i)(i = 1, \cdots, d)$ with 2-bit leakage and computed $t_i = |r \cdot 2^{-2}s^{-1}|_q$, $u_i = |-h(m) \cdot 2^{-2}s^{-1}|_q$ as in the previous sections, so the nonce $k_i = 2^2 b_i$ where b_i is some integer. We can guess the third least significant bit of the nonce, thus constructing a HNP inequality with 3-bit leakage with probability $\frac{1}{2}$.

If the third bit is zero, then

$$k_i = 2^3 b'_i$$

and set

$$t'_i = |r \cdot 2^{-3}s^{-1}|_q$$
$$\text{and } u'_i = |-h(m) \cdot 2^{-3}s^{-1}|_q$$

If the third bit is 1, then

$$k_i = 2^3 b'_i + 2^2$$
$$\Leftrightarrow \alpha r s^{-1} \equiv 2^3 b'_i + 2^2 - h(m)s^{-1} \bmod q$$
$$\Leftrightarrow \alpha r s^{-1} 2^{-3} \equiv b'_i + 2^{-1} - h(m)s^{-1}2^{-3} \bmod q$$

And then set

$$t'_i = |r \cdot 2^{-3} s^{-1}|_q$$
$$\text{and } u'_i = |2^{-1} - h(m) \cdot 2^{-3} s^{-1}|_q$$

Note that here 2^{-1} means the inverse of 2 mod q, not the fractional number $\frac{1}{2}$. Although we do not know whether the third least significant bit is 0 or 1, by trying this two new settings of t'_i and u'_i, we are essentially guessing the third bit and construct t'_i and u'_i with 3-bit leakage, of which the success probability is $\frac{1}{2}$. Recall that typically, for 2-bit leakage, we need about 90 signatures to perform the attack, thus the lattice basis matrix is:

$$B = \begin{pmatrix} 8q & 0 & \cdots & 0 & 0 \\ 0 & 8q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 8q & 0 \\ 8t_1 & 8t_2 & \cdots & 8t_{90} & 1 \end{pmatrix}$$

By guessing bits for all the signatures, we can construct a new matrix with all the inequalities having 3-bit leakage:

$$B = \begin{pmatrix} 16q & 0 & \cdots & 0 & 0 \\ 0 & 16q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 16q & 0 \\ 16t'_1 & 16t'_2 & \cdots & 16t'_{90} & 1 \end{pmatrix}$$

Of course, with this new matrix, we could attack 160-bit (EC)DSA easily, since we know that for 3-bit leakage, standard lattice attack works well. However, we are paying a price of 2^{90} of guessing one more bit for all the signatures, which is unacceptable. In order to avoid the huge computation, instead of guessing one more bit for all the signatures, we could guess one more bit for part of the signatures, thus constructing a hybrid lattice. For instance, we can guess one more bit for 20 out of the 90 signatures and keep the other 70 signatures unchanged as follows:

$$B = \begin{pmatrix} 16q & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ & \vdots & 16q & 0 & & \vdots & \vdots \\ 0 & \cdots & 0 & 8q & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \vdots & \cdots & \vdots & \vdots \\ 16t'_1 & \cdots & 16t'_{20} & 8t_{21} & \cdots & 8t_{90} & 1 \end{pmatrix}$$

Now we have increased the volume of the lattice by 2^{20} times and then we perform the lattice attacks on the new matrix at the cost of 2^{20} operations for guessing bits.

To make everything clear, our approach can be summarised as the following steps:

- Step 1: Collect d signatures $(r_i, s_i)(i = 1, \cdots, d)$, and construct t_i and u_i with the original 2-bit leakage.
- Step 2: For k of them, guess and enumerate the third least significant bit of nonces and construct t'_i and u'_i with 3-bit leakage. For all the other signatures, keep t_i and u_i unchanged.
- Step 3: Construct the hybrid lattice, then use Kannan's embedding method to find the private key(for lattice reduction we use BKZ-30), if failed, go back to step 2.

Under worst circumstances, we have to perform 2^k times step 2 and 3, since there are 2^k possibilities of the third bits of the nonces. With this method, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage, 256-bit (EC)DSA with 3-bit nonce leakage, 384-bit (EC)DSA with 4-bit nonce leakage. See the section of experimental results.

6 Utilizing More Data to Improve Lattice Attacks

In 2000, Bleichenbacher presented a purely statistical attack technique against biased nonces at the IEEE P1363 meeting. The main idea of Bleichenbacher attack is to define a bias function and search for a candidate value that is near the secret key, thus finding many MSBs of the secret key. An advantage of Bleichenbacher attack is that it can deal with small biases in principle at the cost of using many signatures as input. There is a question that whether we can improve lattice attacks with more signatures when leakage is small. We give an answer to this question. Again we are in the context of 160-bit modulus with 2-bit leakage, and for other modulus it is the same.

6.1 From Bleichenbacher to Lattice

Motivated by the Bleichenbacher attack, similar ideas could be applied to lattice attacks. Suppose that we have d HNP samples with l-bit leakage

$$|\alpha t_i - u_i|_q < q/2^l (i = 1, 2, \cdots d)$$

Of course, α is the best candidate for those d inequalities, but there are also many numbers β that satisfy all(or most) of the d inequalities. Those candidates scatter around $(0, q)$. From the view of lattice reduction, when leakage is large, α outperforms other candidates significantly, so it is easy to find α. But when leakage is small, lattice reduction will find one of those candidates. All of the candidates have some internal relation with α, but we don't know how to express it.

Suppose that β is a number near α(let's say sharing c MSBs with α and differ in (160 - c) LSBs), and if $t_i (i = 1, 2, \cdots, d)$ is small, then the value of

$$(\alpha - \beta) t_i (i = 1, 2, \cdots d)$$

is a very small perturbation compared with $q/2^l$, so this means that with high probability, β will satisfy all the d inequalities. The advantage we get here is that all the 2^{160-c} numbers that share c MSBs with α will be a good candidate for lattice reduction. Recall that the original lattice we construct is

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 \end{pmatrix}$$

and $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \cdots, \alpha 2^{l+1}t_d + c_d 2^{l+1}q, \alpha)$ is the hidden lattice vector. The advantage is that now we have 2^{160-c} hidden lattice vectors $(\beta 2^{l+1}t_1 + c_1 2^{l+1}q, \cdots, \beta 2^{l+1}t_d + c_d 2^{l+1}q, \beta)$, where β is any arbitrary integer in $\mathbb{Z}_q$ such that β shares the c MSBs with α. All the hidden vectors are close to the target vectors and carry information about α(c MSBs of α) as well. In this way, the success rate of lattice attack will increase significantly, since we have exponentially many number of hidden vectors. Once we recover one of those candidates, we get c MSBs of the secret key α. As soon as sufficiently many bits of α are known, Pollards lambda method [Pol00] can be used to derive the remaining bits. Besides,

we can also construct new HNP instance for the remaining bits. As we discussed in the previous section, when the length of the secret key is small, lattice attacks become much easier.

6.2 Formalizing the Attack

As mentioned before, we have d HNP inequalities with l-bit leakage

$$|\alpha \cdot t_i - u_i|_q < q/2^l (i = 1, 2, \cdots d)$$

and write the secret key α as

$$\alpha = \alpha_1 \cdot 2^c + \alpha_2 (0 \leq \alpha_2 < 2^c)$$

Where α_1 is the c MSBs of α and α_2 is the remaining LSBs. If $t_i (i = 1, 2, \cdots d)$ is small enough, $\alpha_2 \cdot t_i (i = 1, 2, \cdots d)$ will be a very small perturbation compared with $q/2^l$. This means that with high probability, $\alpha_1 \cdot 2^c$ will satisfy all the d inequalities. And then construct the lattice as:

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^c \cdot 2^{l+1}t_1 & 2^c \cdot 2^{l+1}t_2 & \cdots & 2^c \cdot 2^{l+1}t_d & 2^c \end{pmatrix}$$

Then $(\alpha_1 2^c \cdot 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha_1 2^c \cdot 2^{l+1}t_2 + c_2 2^{l+1}q \cdots, \alpha_1 2^c \cdot 2^{l+1}t_d + c_d 2^{l+1}q, \alpha_1 2^c)$ will be the hidden lattice vector. The advantage that we get is that the volume of the lattice is increased by 2^c times, while the Euclidean norm of the difference vector almost keeps unchanged, thus making the attack much easier. This attack can be summarised as the following steps.

- Step 1: Collect signatures (r, s) and construct

  $$t = 2^{-l} s^{-1} r \bmod q$$
  $$u = -2^{-l} s^{-1} h(m) \bmod q$$

  If t is small enough(smaller than some predetermined bound), then keep the (t, u) pairs, otherwise throw it.

- Step 2: Keep doing step 1 until we get d pairs $(t_i, u_i)(i = 1, \cdots, d)$.

- Step 3: Construct the above lattice and use Kannan embedding to do lattice attacks.

- Step 4: Find α_1 which is the c MSBs of α.

- Step 5: Find the remaining bits of α.

As we previously discussed, once we have recovered many MSBs of α, recovering the remaining bits becomes pretty easy. With this method, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage, 256-bit (EC)DSA with 3-bit nonce leakage, 384-bit (EC)DSA with 4-bit nonce leakage. See the section of experimental results.

7 Batch SVP and Kannan Embedding Factor

7.1 Batch SVP

In section 4 and section 5, we have to do 2^c(typically we set $c = 15, 20$) BKZ-30 operations on the following matrices:

$$C = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 & 0 \\ & \vdots & & \vdots & & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 & 0 \\ v_1 & v_2 & \cdots & v_d & 0 & q \end{pmatrix}$$

Write C as

$$C = \begin{pmatrix} B & 0 \\ \mathbf{v} & q \end{pmatrix}$$

Each time we perform BKZ operations, only the last row of matrix C is changed and B is fixed. Since one BKZ-30 operation on a 90-dimensional lattice typically takes about 3 minutes with fplll[dt20] library on Sagemath[The20]. If c = 2^{15}, the time complexity will be $2^{15} \cdot 3$ minutes without using multiple cores. Although this is practical time, we could further improve the time complexity. We use LLL as an example here, for BKZ it is similar. In LLL algorithm[LLL82], there is an index k starting from 1, which represents the row currently being reduced. Besides, there is an exchange condition, and if it satisfies, two adjacent rows will be exchanged. After exchanging rows and recomputing the Gram-Schmidt norm, size reduction will be performed.

If we consider the process of LLL reduction on the matrix C, essentially it will first reduce the submatrix B and in the last round, it will reduce the whole matrix C. So every time the reduction on B is repeated, which is not necessary. We come up with a simple solution:

- Step 1: BKZ-reduce the submatrix B.(preprocessing)
- Step 2: Do Kannan embedding and construct the matrix C.
- Step 3: Do BKZ on the matrix C again.

This actually means that we preprocess the submatrix B. In this way, we saved a lot of computation. With this method, one BKZ-30 operation typically takes several seconds, while the original one takes about 3 minutes.

7.2 Kannan Embedding Factor

In our experiments, we observe that lattice attacks on (EC)DSA are very sensitive to the Kannan embedding factor. As we can see from this table, if the coefficient is either too small or too large, the success rate becomes very low.

Table 1: Kannan Embedding factor test

modulus	leakage	signatures	Kannan Embedding factor	success rate
160-bit	3	80	q	93/100
160-bit	3	80	$(q-1)/2$	97/100
160-bit	3	80	q^2	5/100
160-bit	3	80	1	0/100

Here we give an explanation why this happens. Denote the Kannan embedding factor as γ. For simplicity, we analyze LLL reduction, but for BKZ, it is quite similar.

7.2.1 Case 1: Kannan embedding factor is too large

Recall that the embedded matrix C is

$$C = \begin{pmatrix} B & 0 \\ \mathbf{v} & \gamma \end{pmatrix}$$

The Gram-Schmidt norm of the last row is γ, if γ is too large, after LLL reduction on the submatrix B, the exchange condition will not satisfy, then only one round of size reduction will be performed(reduce the last row from the first (d + 1) rows). By contrast, if γ is properly valued, the exchange condition will satisfy and the last row will be exchanged to some other row. Then Gram-Schmidt norm will be recomputed and one round of size reduction will be performed. Typically, the exchange will happen many times, so many rounds of size reduction will be performed. Therefore, if γ is too large, the lattice will get much less reduced.

In order to make this clear, we illustrate the idea by a simple example. In this example, modulus q = 89, leakage l = 4. The matrix B

$$B = \begin{pmatrix} 2848 & 0 & 0 & 0 \\ 0 & 2848 & 0 & 0 \\ 0 & 0 & 2848 & 0 \\ 416 & 1600 & 96 & 1 \end{pmatrix}$$

the target vector is $\mathbf{v} = (761, 601, 2393, 0)$. Now we set the Kannan embedding factor $\gamma = q^2 = 7921$ and the embedded matrix C is

$$C = \begin{pmatrix} 2848 & 0 & 0 & 0 & 0 \\ 0 & 2848 & 0 & 0 & 0 \\ 0 & 0 & 2848 & 0 & 0 \\ 416 & 1600 & 96 & 1 & 0 \\ 761 & 601 & 2393 & 0 & 7921 \end{pmatrix}$$

Now let us consider the process of LLL-reduction on C. After many rounds of reduction, the matrix becomes

$$C = \begin{pmatrix} 0 & 0 & 0 & 89 & 0 \\ 96 & -288 & -416 & -34 & 0 \\ -160 & 480 & -256 & 27 & 0 \\ 832 & 352 & 192 & 2 & 0 \\ 761 & 601 & 2393 & 0 & 7921 \end{pmatrix}$$

Note that up to now, the last row still remains unchanged. Now the index k for LLL becomes 5, which means reducing the fifth row. First, LLL algorithm checks whether the exchange condition satisfies, since the Kannan embedding factor γ is large, the exchange does not happen. After that, LLL algorithm does one round of size reduction, which means

- Reduce the fifth row from the fourth row.
- Reduce the fifth row from the third row.
- Reduce the fifth row from the second row.
- Reduce the fifth row from the first row.

and the algorithm terminates.

By contrast, if we set the Kannan embedding factor $\gamma = q = 89$ and consider LLL-reduction, at some point, it will reach the same state

$$C = \begin{pmatrix} 0 & 0 & 0 & 89 & 0 \\ 96 & -288 & -416 & -34 & 0 \\ -160 & 480 & -256 & 27 & 0 \\ 832 & 352 & 192 & 2 & 0 \\ 761 & 601 & 2393 & 0 & 89 \end{pmatrix}$$

Now the Gram-Schmidt norm of the last row is not so large, and the exchange condition will satisfy, so the target vector $\mathbf{v}$ goes to the fourth row.

Then a recomputation of Gram-Schdimt norm takes place and one round of size reduction happens.

$$C = \begin{pmatrix} 0 & 0 & 0 & 89 & 0 \\ 96 & -288 & -416 & -34 & 0 \\ -160 & 480 & -256 & 27 & 0 \\ -391 & 1209 & 1689 & 52 & 89 \\ 832 & 352 & 192 & 2 & 0 \end{pmatrix}$$

Then LLL algorithm checks exchange condition again and the targe vector $\mathbf{v}$ goes to the third row. Again, a recomputation of Gram-Schdimt norm takes place and one round of size reduction happens:

$$C = \begin{pmatrix} 0 & 0 & 0 & 89 & 0 \\ 96 & -288 & -416 & -34 & 0 \\ -7 & 57 & 25 & -84 & 89 \\ -160 & 480 & -256 & 27 & 0 \\ 832 & 352 & 192 & 2 & 0 \end{pmatrix}$$

Then LLL algorithm checks exchange condition again and the targe vector $\mathbf{v}$ goes to the second row. Again, a recomputation of Gram-Schdimt norm takes place and one round of size reduction happens:

$$C = \begin{pmatrix} 0 & 0 & 0 & 89 & 0 \\ -7 & 57 & 25 & 5 & 89 \\ 82 & -174 & -366 & -24 & 178 \\ -146 & 366 & -306 & 17 & -178 \\ 846 & 238 & 142 & -8 & -178 \end{pmatrix}$$

As we can see from this process, when $\gamma = q$, three rounds of size reduction happen, so this means that the target vector $\mathbf{v}$ gets more reduced.

7.2.2 Case 2: Kannan Embedding factor is too small

Since the Gram-Schmidt norm of the last row is γ, if the Kannan embedding factor is too small, the Gram-Schmidt norm of the last row will be very small. After exchanging rows, size reduction will be performed. Since the Gram-Schmidt norm is small, when performing size reduction on other rows, many multiples of the target vector will be added to other rows. However, since

$$B = \begin{pmatrix} 2^{l+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{l+1}q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 2^{l+1}q & 0 \\ 2^{l+1}t_1 & 2^{l+1}t_2 & \cdots & 2^{l+1}t_d & 1 \end{pmatrix}$$

what we want is $\alpha \cdot (2^{l+1}t_1, 2^{l+1}t_2, \cdots, 2^{l+1}t_d, 1) - \mathbf{v}$, so we do not want to use the target vector $\mathbf{v}$ to reduce other vectors. If γ is too small, we will find that all the vectors in the reduced basis will have a very large coefficient of $\mathbf{v}$, which is not our goal.

8 Gap between CVP and SVP Approaches

As mentioned in [JSSS20], we also observed a certain gap between the nearest plane algorithm and Kannan's embedding method. In this attack, Kannan's embedding always outperforms nearest plane algorithm to some extent.

Table 2: A comparison of CVP and SVP approach

modulus	leakage	nearest plane	Kannan Embedding
160	4	37/100	100/100
160	3	0/100	91/100

As we can see from the table, for 160-bit (EC)DSA with 4-bit nonce leakage, both CVP approach and SVP approach work well. However, CVP approach seldom succeeds for 3-bit leakage, while Kannan's embedding method works quite well.

8.1 Reason for the Gap

Essentially, nearest plane algorithm can be regarded as one round of size reduction in the embedded lattice. Again, we illustrate the idea by the same simple example as in the previous section. In this example, modulus q = 89, leakage l = 4. The matrix B

$$B = \begin{pmatrix} 2848 & 0 & 0 & 0 \\ 0 & 2848 & 0 & 0 \\ 0 & 0 & 2848 & 0 \\ 416 & 1600 & 96 & 1 \end{pmatrix}$$

the target vector is $\mathbf{v} = (761, 601, 2393, 0)$. Now we set the Kannan Embedding factor $\gamma = q^2 = 7921$ and the embedded matrix C is

$$C = \begin{pmatrix} 2848 & 0 & 0 & 0 & 0 \\ 0 & 2848 & 0 & 0 & 0 \\ 0 & 0 & 2848 & 0 & 0 \\ 416 & 1600 & 96 & 1 & 0 \\ 761 & 601 & 2393 & 0 & 7921 \end{pmatrix}$$

Recall the process in the previous section, if the Kannan embedding factor is large enough, nearest plane algorithm will be the same as Kannan embedding, because for the last row of the embedded lattice, the exchange condition will not satisfy and only one round of size reduction takes place, which is essentially the same as nearest plane. However, if the Kannan embedding factor is properly valued, many exchanges will happen. After one exchange, one round of size reduction will take place, which means that Kannan embedding will make the target vector more reduced compared with nearest plane.

9 Experimental Result

In this section, we show the result of our practical experiments. All the experiments are carried out on AMD Ryzen 3970x with Sagemath[The20] and fplll[dt20] library. For lattice reduction algorithms, we are using BKZ-30.

9.1 Guessing Bits of Secret Key

In this table, the column "bits guessed" means the number of bits of the secret key that we try to guess and enumerate. As we can see in the table, for 160-bit (EC)DSA with 2-bit nonce leakage, if we guess 15 bits for the secret key, we succeed in recovering the secret key 12 times among 200 experiments. Since we enumerate 15 bits of the secret key, the time complexity is upper bounded by 2^{15} BKZ-30 operations(the expected number is 2^{14}). In this way, we are able to quantify the complexity in terms of BKZ operations. Instead of directly using real-time, the advantage is that quantification in terms of BKZ operations gives us a clear impression of the time complexity and this is independent of the machine being used. Besides, it is pretty easy to estimate the practical attack time. For instance, with a 32-cores machine available and batch SVP technique described in section 7, one BKZ-30 operation on a 90-dimensional lattice typically takes 40 seconds, so the expected time is $\frac{2^{14} \cdot 40s}{32} \approx 10200s$, which is several hours.

Table 3: Guessing Bits of Secret key

modulus	leakage	signatures	bits guessed	success rate
160	2	90	40	147/200
160	2	90	30	78/200
160	2	90	20	24/200
160	2	90	15	12/200
256	3	100	40	187/200
256	3	100	30	135/200
256	3	100	20	68/200
256	3	100	10	15/200
384	4	100	40	83/200
384	4	100	30	30/200
384	4	100	20	14/200

9.2 Guessing Bits of Nonce

Similarly, the column "nonce guessed" means the number of nonces that we guess and enumerate for 1 more bit. For 160-bit (EC)DSA with 2-bit nonce leakage, if guessing 1 more bit for 20 of the 90 signatures, we succeed in recovering the secret key 14 times out of 200 experiments, so the time complexity is 2^{20} BKZ-30 operations. Actually, we could even estimate the time complexity for 1-bit nonce leakage. What we could do is to guess 2 more bits for 20 of the signatures and guess 1 more bit for the other 70 signatures, so the time complexity is $4^{20} \cdot 2^{70} = 2^{110}$ BKZ-30 operations. Although this is not practical(thus not so meaningful), it is an estimate of computation cost for 1-bit leakage.

Table 4: Guessing Bits of Nonce

modulus	leakage	signatures	nonce guessed	success rate
160	2	90	40	83/200
160	2	90	30	30/200
160	2	90	20	14/200
256	3	100	40	133/200
256	3	100	30	64/200
256	3	100	20	13/200
256	3	100	10	1/200
384	4	100	40	105/200
384	4	100	30	41/200
384	4	100	20	11/200
384	4	100	10	2/200

9.3 Improving Lattice Attacks with More Data

Recall that in the section 6, we discussed that for one HNP inequality $|\alpha t - u|_q < q/2^l$, if we get small t, then we can construct a lattice that has larger volume. In our experiments, we find that for 160, 256, 384-bit modulus q, if t has less than 140, 226, 344 bits respectively, we can perform the attack. Take 160-bit modulus for example, in order to get 90 inequalities where all the $t \leq 2^{140}$, we have to sample $2^{20} \cdot 90 \approx 2^{27}$ signatures. This may seems too many in practical setting, but the advantage is that we could recover about 140 MSBs of the secret key in just one BKZ-30 operation.

Table 5: Utilizing More Data to Improve Lattice Attacks

modulus	leakage	upperbound for t	signatures	time complexity	success rate
160	2	2^{140}	2^{27}	1 BKZ-30	30/200
256	3	2^{226}	2^{37}	1 BKZ-30	27/200
384	4	2^{344}	2^{47}	1 BKZ-30	62/200

9.4 Experiments on the TPM-Fail Dataset

We also carried out experiments on the TPM-Fail[MSEH20] dataset(256-bit ECDSA). The first row of the dataset contains the public key and the message being signed. Each of the other rows contains (r, s) and t, where (r, s) is the signature and t is the signing time. One typical way to perform the attack is:

- Collect N signatures.
- Choose d out of the N signatures, whose signing time is the fastest.
- For each of the d signatures, assign leakage l.
- Construct HNP inequalities and perform lattice attacks.

For 256-bit modulus, if setting $l = 3$, typically $d \approx 90$. In [MSEH20], the authors used about 40000 signatures and in [JSSS20], a new technique of geometric assignment of leakage was proposed: assign half of the d signatures with leakage $l = 3$, one fourth of them having leakage $l = 4$, and so on. In our experiments, we combine these techniques with our method of guessing bits of the secret key and come up with the following attack:

- Randomly collect 800 signatures.

- Choose 90 out of the 800 signatures, whose signing time is the fastest.
- Geometrically assign the leakage l.
- Guess and enumerate 20 LSBs of the secret key and perform lattice attacks described in section 4.

We did 100 experiments and succeeded 3 times. In this way, with only 800 signatures available, we are able to recover the secret key for TPM-Fail dataset.

References

[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 13*, 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.

[AH] Martin R Albrecht and Nadia Heninger. On bounded distance decoding with predicate: Breaking the lattice barrier for the hidden number problem.

[Ajt06] Miklós Ajtai. Generating random lattices according to the invariant distribution. *Draft of March*, 2006, 2006.

[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 20*, pages 225–242. ACM Press, November 2020.

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

[Ble00] Daniel Bleichenbacher. On the generation of one-time keys in dl signature schemes. In *Presentation at IEEE P1363 working group meeting*, page 81, 2000.

[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'96*, volume 1109 of *LNCS*, pages 129–142. Springer, Heidelberg, August 1996.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, September 1997.

[DHMP13] 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.

[dt20] The FPLLL development team. fplll, a lattice reduction library, Version: 5.4.0. Available at https://github.com/fplll/fplll, 2020.

[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.

[HGS01] Nick A Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, 23(3):283–290, 2001.

[JSSS20] Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sys. Minerva: The curse of ECDSA nonces. *IACR TCHES*, 2020(4):281–308, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8684.

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

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

[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.

[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.

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

[Pol00] John M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, September 2000.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[Sha82] Adi Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 145–152. IEEE, 1982.

[The20] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.2)*, 2020. https://www.sagemath.org.

[TTA18] Akira Takahashi, Mehdi Tibouchi, and Masayuki Abe. New Bleichenbacher records: Fault attacks on qDSA signatures. *IACR TCHES*, 2018(3):331–371, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7278.