IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 1, pp. 391–413. DOI:10.46586/tches.v2022.i1.391-413

Guessing Bits: Improved Lattice Attacks on (EC)DSA with Nonce Leakage

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

[1] Kyoto University, Kyoto, Japan,
sun.chao.46s@st.kyoto-u.ac.jp,
[2] NTT Corporation, Tokyo, Japan,
{thomas.espitau.ax,mehdi.tibouchi.br,masayuki.abe.cp}@hco.ntt.co.jp

Abstract. The lattice reduction attack on (EC)DSA (and other Schnorr-like signature schemes) with partially known nonces, originally due to Howgrave-Graham and Smart, has been at the core of many concrete cryptanalytic works, side-channel based or otherwise, in the past 20 years. The attack itself has seen limited development, however: improved analyses have been carried out, and the use of stronger lattice reduction algorithms has pushed the range of practically vulnerable parameters further, but the lattice construction based on the signatures and known nonce bits remain the same.

In this paper, we propose a new idea to improve the attack based on the same data in exchange for additional computation: carry out an exhaustive search on some bits of the secret key. This turns the problem from a single bounded distance decoding (BDD) instance in a certain lattice to multiple BDD instances in a fixed lattice of larger volume but with the same bound (making the BDD problem substantially easier). Furthermore, the fact that the lattice is fixed lets us use batch/preprocessing variants of BDD solvers that are far more efficient than repeated lattice reductions on non-preprocessed lattices of the same size. As a result, our analysis suggests that our technique is competitive or outperforms the state of the art for parameter ranges corresponding to the limit of what is achievable using lattice attacks so far (around 2-bit leakage on 160-bit groups, or 3-bit leakage on 256-bit groups).

We also show that variants of this idea can also be applied to bits of the nonces (leading to a similar improvement) or to filtering signature data (leading to a data-time trade-off for the lattice attack). Finally, we use our technique to obtain an improved exploitation of the TPM–FAIL dataset similar to what was achieved in the Minerva attack.

Keywords: ECDSA · Lattice Attacks · Hidden Number Problem · Success-Time Trade-Off · Cryptanalysis

1 Introduction

A lattice is a discrete group of points in space, which can be defined as the set of all integer linear combinations of a certain set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ known as a basis. A lattice has infinitely many bases, but so-called "reduced" bases, that consist of short and close to orthogonal vectors, are much more interesting. Lattice reduction, the mathematical problem of finding such bases, has a long history which can be traced back to the 18th century, but gained particular prominence after Lenstra, Lenstra and Lovász [LLL82] introduced a polynomial-time approximate algorithm for it in 1982 that became known as LLL. Since the advent of LLL, lattice reduction proved to be a powerful tool for cryptanalysis: early examples include attacks on knapsack-based

Received: 2021-07-15 Accepted: 2021-09-15 Published: 2021-11-19

cryptosystems [Sha82] and Coppersmith's small root finding algorithm [Cop97] that broke many variants of RSA in particular.

This paper focuses on another major cryptanalytic application of lattice reduction: lattice attacks against (EC)DSA (and related signature schemes like Schnorr's) when bits of the nonce are known. DSA and ECDSA are well-established standards for digital signature based on the discrete logarithm problem, and that involve the use, for each generated signature, of some fresh random value called the *nonce*. It is well-known that if the same nonce is used twice, the adversary can directly compute the private key due to a linear relation between the nonce and the private signing key. Even worse, *partial* information about the nonces of multiple signatures can lead to recovery of the full private key. The original approach to do so, due to Bleichenbacher, actually relied on discrete Fourier analysis techniques [Ble00, DHMP13, AFG+14b, TTA18, ANT+20], but lattice reduction was also discovered to provide an attack technique, in connection to Boneh and Venkatesan's hidden number problem (HNP) [BV96].

HNP is a number theoretic problem that was originally introduced to establish bit security results for the Diffie–Hellman key exchange. Boneh and Venkatesan showed that it could be seen as a bounded distance decoding (BDD) instance in a lattice, which could be solved with Babai's nearest plane algorithm [Bab86] for suitable parameters. Subsequently, Howgrave-Graham and Smart [HGS01], and later Shparlinski and Nguyen [NS02], observed that the problem of attacking (EC)DSA if some top or bottom bits of the nonces are known is an instance of HNP, and could be attacked using the same lattice techniques. However, when nonce leakage is very small, the attack becomes much more difficult mainly because the hidden lattice vector in BDD is not very close to the target vector. It took significant development in lattice reduction algorithms to advance the state of the art. In 2013, Liu and Nguyen [LN13] were able to attack 160-bit DSA with 2-bit nonce leakage using the BKZ 2.0 algorithm introduced just a few years earlier [CN11], relying on a very high block size of 90, with pruned enumeration as the SVP oracle. In a very recent work [AH21], Albrecht and Heninger utilize the state-of-the-art lattice reduction algorithm G6K [ADH+19] together with the novel idea of predicate sieving to break new records.

1.1 Our Contributions

Lattice attacks on (EC)DSA are in general *all-or-nothing*, in the sense that the attack reveals the entire secret key when it succeeds, and nothing at all otherwise. In contrast, Bleichenbacher's statistical attack, for example, only reveals some bits of the secret key in a single execution; however, it has been observed in previous work that the knowledge of those bits makes subsequent applications of the attack much more efficient.

How the knowledge of some bits of the secret key affects lattice attacks on (EC)DSA, however, does not appear to have been considered in previous work[1]. Perhaps interestingly, we observe that knowledge of some bits of the secret *does* in fact make the attack easier. This results in a simple idea to improve those attacks, which forms the main contribution of this paper: *guess* some bits of the secret key, and solve the resulting, easier lattice problem for each possible guess (in other words: carry out an exhaustive search on those bits).

An interesting feature of this approach is that this reduces the attack to solving many BDD instances with varying target vectors in the *same* lattice, making it possible to rely on various batch-CVP or CVP-with-preprocessing techniques to solve them. At the

[1] How the knowledge of certain types of side information on the secret affects the hardness of *lattice problems* like LWE has been considered, e.g., in [DDGR20]. The context of HNP/nonce leakage, however, is very different: for example, the key in our setting is an element of $\mathbb{Z}_q$, as opposed to a vector in LWE; the nature of the hints (bits vs. linear relations) is different; the lattice is very structured (knapsack-like) for HNP, as opposed to random q-ary for LWE; the BDD parameters are totally dissimilar; the analysis in their case depends on Gaussian noise, etc. So the two questions appear to be mostly unrelated.

Table 1: Tractable parameters for lattice attacks on (EC)DSA.

	Nonce leakage			
Modulus	4-bit	3-bit	2-bit	1-bit
160-bit	Easy	Easy	[LN13], [AH21], Ours	Hard
256-bit	Easy	[AH21], Ours	Hard	Hard
384-bit	[AH21], Ours	Hard	Hard	Hard

simplest level, even just carrying out an initial lattice reduction on the original BDD lattice, and then solving all the BDD instances by reduction to SVP using Kannan's embedding technique turns out to be far more efficient than naively solving the SVP instances without the initial common lattice reduction.

Additionally, this approach parallelizes very easily, and has the convenient property of being very easy to simulate (in the sense that one can certainly make the "correct" guess for self-generated instances), which makes its cost easy to predict even for parameters that are impractical to fully run in a short time.

As additional contributions, we also show that the same idea can be applied to guessing additional bits of some of the signature nonces (on top of those already known; this results in a similar, but usually slightly worse, success-time trade-off than guessing bits of the secret key), as well as filtering some of the signatures to construct lattices that are easier to attack (resulting in a data-time trade-off reminiscent to what can be achieved in Bleichenbacher's attack). Furthermore, we carry out experiments on the TPM–FAIL dataset [MSEH20] and apply our techniques 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 around 800 signatures, which is comparable to the results achieved in Minerva [JSSS20].

1.2 Related Work

The main question that we consider, namely how small of a nonce leakage do we need to recover the signing key in (EC)DSA, has been considered in previous work both for lattice attacks and for Bleichebacher's attack. In the case of lattice attacks, the record-holding works are due to Liu and Nguyen [LN13] and very recently Albrecht and Heninger [AH21]. In the case of Bleichenbacher's attack, the state of the art is presented in [ANT+20]. We briefly describe below how our results compares to theirs.

Comparison with [LN13] and [AH21]. Table 1 presents typical parameters (in terms of group size and number of known nonce bits) for (EC)DSA, and indicates whether they can be tackled easily with lattice attacks ("Easy"), are considered hard so far with lattices ("Hard"), or have been solved in specific papers. In [LN13] and [AH21], strong lattice reduction algorithms (BKZ 2.0 and G6K with predicate respectively) are used to attack the "borderline" cases, namely 160-bit modulus with 2-bit nonce leakage, 256-bit modulus with 3-bit nonce leakage and 384-bit modulus with 4-bit nonce leakage. Our approach all makes those borderline cases tractable, but relies on very different techniques and arguably present various advantages, particularly the following:

- whereas [LN13] and [AH21] are based on specific improvements and modifications of the underlying lattice reduction algorithms, our approach works with any lattice reduction algorithm. In our experiments, we use fplll's implementation of BKZ–30, but any algorithm would work. In particular, it is straightforward to combine our idea with the techniques of those two papers if so desired;

- tailoring the parameters of [LN13] and [AH21] to a specific problem instance or to the specific computational resources of the attack can be quite challenging; in contrast, due to its straightforward simulatability mentioned above, our approach makes this easy, and makes it possible to quantify the cost of attacking a given problem instance in very concrete terms in advance.

Comparison with Bleichenbacher's attack. Although we come up with an approach to improve lattice attacks with more signatures and in some sense bridge the gap between lattice attacks and Bleichenbacher's, it still requires too many signatures compared with Bleichenbacher's 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^{12} signatures for 2-bit leakage case and 2^{27} signatures for the one-bit leakage case [ANT+20]. Besides, with this approach, we still could not attack harder cases, such as 160-bit modulus with 1-bit nonce leakage and 256-bit modulus with 2-bit nonce leakage, which are already tractable using Bleichenbacher's attack [AFG+14b, TTA18, ANT+20]. However, the fact that there exists a way of improving lattice attacks with more signatures might give some ideas for future work. It is still possible that better ways of utilizing more signatures for lattice attacks exist, and we hope that lattice attacks on (EC)DSA could be further improved.

2 Preliminaries

2.1 Lattices

A *lattice*[2] is an additive subgroup of $\mathbb{Z}^n$ for some $n \geq 0$. For any family of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ of $\mathbb{Z}^n$, the set:

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

is a lattice, and conversely, any lattice $\mathcal{L} \subset \mathbb{Z}^n$ can be put in that form for some vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$. In that case, the family $(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is called a basis of $\mathcal{L}$. We can then represent the lattice $\mathcal{L}$ by the $m \times n$ matrix B whose rows are formed by the vectors $\mathbf{b}_i$, and write $\mathcal{L} = \mathcal{L}(B)$.

A given lattice $\mathcal{L}$ can have infinitely many distinct bases, but they all have the same cardinality m, called the *rank* of $\mathcal{L}$. In this paper, we will only consider *full-rank* lattices, whose rank m is equal to n, the dimension of the ambient space. For a full-rank lattice $\mathcal{L}$ with basis matrix B, we define the volume of $\mathcal{L}$ as the quantity:

$$\mathrm{vol}(\mathcal{L}) = |\det(B)|,$$

which does *not* depend on the choice of B.

The Euclidean norm of the shortest non-zero vector in $\mathcal{L}$ is called the first minimum of $\mathcal{L}$ and denoted as $\lambda_1(\mathcal{L})$. More generally, for $1 \leq i \leq n$, the i-th minimum $\lambda_i(\mathcal{L})$ of $\mathcal{L}$ 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, with high probability,

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

[2] This is more properly the definition of an *integral* lattice, but integral lattices are the only ones we consider in this paper.

The approximation factor of a lattice basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined as $\frac{\|\mathbf{b}_1\|}{\lambda_1(\mathcal{L})}$ (where $\|\cdot\|$ henceforth denotes the Euclidean norm), and the root Hermite factor is defined as $(\frac{\|\mathbf{b}_1\|}{\mathrm{vol}(\mathcal{L})^{1/n}})^{1/n}$.

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 $\mathbf{v} \in L$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$. Another problem is the Closest Vector Problem (CVP for short): given a lattice $\mathcal{L}$ and a target vector $\mathbf{t}$, find the vector $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{t}\|$ is minimal.

There exist efficient lattice algorithms for solving approximate versions of SVP and CVP. For approximate SVP, lattice reduction algorithms such as LLL [LLL82] and BKZ [SE94] output lattice basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ such that the approximation factor and the root Hermite factor are relatively small. As a result, the first vector $\mathbf{b}_1$ of the reduced basis is a good approximation of the shortest non-zero vector. For approximate CVP, Babai's nearest plane algorithm [Bab86] and variants of it such as [Kle00, GPV08, EK20] can be used to find a relatively close vector when applied after a lattice reduction algorithm.

2.2 Hidden Number Problem

The Hidden Number Problem can be described as follows: q, l are fixed integers known to the public and α is a unknown integer in $\mathbb{Z}_q$. For many known random $t \in \mathbb{Z}_q$, we have an oracle $\mathcal{O}_\alpha(t)$ that on input t, outputs (t, u) such that $|\alpha \cdot t - u|_q < q/2^l$, where $|z|_q$ represents the unique integer $0 \leq x < q$ such that $x \equiv z \bmod q$. The goal is to recover the hidden secret key α. Suppose that we have queried the oracle d times and have d pairs (t_i, u_i) $(i = 1, 2, \ldots, d)$, we could 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 $\mathbf{h} = (2^l \alpha t_1 + c_1 2^l q, 2^l \alpha t_2 + c_2 2^l q, \ldots, 2^l \alpha t_d + c_d 2^l q, \alpha)$ is a lattice vector (which we call the hidden lattice vector) in $\mathcal{L}$, and set the target vector $\mathbf{v} = (2^l u_1, 2^l u_2, \ldots, 2^l u_d, 0)$. Denote the difference vector $\mathbf{h} - \mathbf{v}$ as $\mathbf{e}$. Since $|2^l \alpha t_i - 2^l u_i + 2^l c_i q| < q$ $(i = 1, 2, \ldots, d)$, it is easy to know that the absolute value of each coefficient of $\mathbf{e}$ is less than q. Therefore, the Euclidean norm of $\mathbf{e}$ is at most $q\sqrt{d+1}$. 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, i.e., the CVP approaches and SVP approaches. In the original paper by Boneh and Venkatesan, they use the 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. We can also use CVP enumeration instead of nearest plane algorithm. Besides, another technique, known as Kannan's embedding method [Kan87], transforms the CVP instance into a SVP instance by embedding the target vector into the original lattice, thus constructing a larger lattice:

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

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

In practice, there are two subtle technical points that we should take care of:

- We might find $q-\alpha$ instead of the secret key α, since $q-\alpha$ is also a good candidate (this can be easily checked). Therefore, we should check both. Note that the checking time is almost negligible compared with the time in lattice reduction, because the only operation is one scalar multiplication (for ECDSA) and checking consistency with public key.
- Typically in practical attacks, the vector that we want is not the first vector of the reduced basis, so we should check every row of the reduced basis. In other words, the attacks are considered successful if we find the vector in any row of the reduced basis (this is typical in the literature).

2.3 (EC)DSA Signature Scheme

Here we only discuss DSA and skip ECDSA, since for the construction of HNP instances, this makes no difference. 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.

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 $y = g^\alpha \bmod p$.

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).

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

2.4 Lattice Attacks on (EC)DSA

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

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

Now in our case, we have l-bit leakage, which means that we know l LSBs of k. In the case of timing attack, MSB is used, where the construction is very similar but slightly subtle. The difference is that when k has some leading zeroes, $k < q/2^l$ might not be true depending on the order q. For more discussion, see Section 4.3 of [JSSS20]. Denote the value of l LSBs as k_1, then we have $k = 2^l k_2 + k_1$ for some integer $0 \le k_2 \le q/2^l$, so:

$$\begin{aligned} \alpha r &\equiv s(2^l k_2 + k_1) - h(m) \pmod q \\ \alpha(rs^{-1} - k_1)2^{-l} &\equiv k_2 - 2^{-l}s^{-1}h(m) \pmod q. \end{aligned}$$

For simplicity of formulas, we set $k_1 = 0$ (without loss of generality, because we know the value of k_1) and have:

$$\begin{aligned} t &\equiv 2^{-l}s^{-1}r \pmod q \\ u &\equiv -2^{-l}s^{-1}h(m) \pmod q \\ k_2 &\equiv \alpha t - u \pmod q. \end{aligned}$$

Note that both t and u can be computed from all the public available information. Since $0 \le k_2 < q/2^l$,

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

In this way, we have constructed an HNP instance for (EC)DSA. Then we solve the HNP either by nearest plane algorithm or Kannan's 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 [NT12]. It works as follows: since

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

there exists some integer c such that

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

Therefore,

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

Now set

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

Then we have

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

Suppose that now we have d signatures (r_i, s_i) $(i = 1, \ldots, d)$ and compute the pairs (t_i, u_i) as previously defined. Then construct a lattice $\mathcal{L}$ spanned by the following matrix B:

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

and everything goes the same.

2.6 Projected Lattice

Typically, in standard lattice attacks, we almost always locate the secret key in the second row (which we hope to be the first) of the reduced basis. In order to deal with this issue, [AH21] makes 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 simple linear combinations of the rows, we could know that $(0, 0, \ldots, 0, q)$ belongs to this lattice. The expected Euclidean norm of the difference vector $\mathbf{e}$ is roughly $\sqrt{\frac{d+1}{3}}q$. With typical parameters such as $d = 85$, $l = 2$, $\|\mathbf{e}\|$ is much larger than q. This means that the difference vector $\mathbf{e}$ will never be the shortest vector in practice. In fact, we can project this lattice orthogonal to $(0, \ldots, 0, q)$ and construct a new lattice:

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

Table 2: Typical number of signatures for 160-bit modulus.

Nonce leakage l	4-bit	3-bit	2-bit	1-bit
Number of signatures d	50	80	100	200

In this new lattice, the hidden vector will be $(|\alpha t_d|_q \cdot 2^{l+1} t_1 (t_d)^{-1} + c_1 2^{l+1} q, \ldots, |\alpha t_d|_q \cdot 2^{l+1} t_{d-1} (t_d)^{-1} + c_d 2^{l+1} q, 2^{l+1} |\alpha t_d|_q)$. The important thing is that the vector $(0, 0, \ldots, 0, q)$ 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

As previously mentioned, there are "borderline" cases that were considered difficult for standard lattice attacks on (EC)DSA, e.g., 160-bit modulus with 2-bit nonce leakage, 256-bit modulus with 3-bit nonce leakage, 384-bit modulus with 4-bit nonce leakage. One important question about this is: *How difficult are those "borderline" cases?* In this section, we explain this question, quantify the difficulty and give intuitive ideas for our attacks in later sections.

3.1 Difficulty When Nonce Leakage is Small

For each HNP 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 vector $\mathbf{v} = (v_1, \ldots, v_d, 0)$ and the hidden lattice vector $\mathbf{h} = (\alpha 2^{l+1} t_1 + c_1 2^{l+1} q, \ldots, \alpha 2^{l+1} t_d + c_d 2^{l+1} q, \alpha)$, thus the Euclidean norm of the difference vector $\mathbf{e}$ is upper bounded by $q\sqrt{d+1}$. The volume of this lattice $\mathcal{L}$ is $q^d 2^{(l+1)d}$, and according to Gaussian Heuristic, 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. Table 2 is the typical number of signatures needed (just information-theoretically, the attack might not be successful at all) to perform the lattice attack on 160-bit (EC)DSA. Now, we give an intuitive explanation of why lattice attacks against (EC)DSA with small nonce leakage are difficult.

When $l = 3$ and $d = 80$ (this case is regarded as "easy" for lattice attacks), the lattice basis matrix B is:

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

The Euclidean norm of the first vector is $16q$, and $\|\mathbf{e}\|$ 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 $\|\mathbf{e}\|$.

When $l = 2$ and $d = 100$ (this case is regarded as "hard" for standard lattice attacks, but have been solved in specific papers), the Euclidean norm of the first vector is $8q$, and $\|\mathbf{e}\|$ is upper bounded by $q\sqrt{d+1} \approx 10q$. To be a bit more precise, we can compute the expected norm, which is roughly $\sqrt{\frac{100}{3}q^2} \approx 6q$.

When $l = 1$ and $d = 200$ (this case remains "hard" so far), similarly, the Euclidean norm of the first vector is $4q$, and $\|\mathbf{e}\|$ is upper-bounded by $q\sqrt{d+1} \approx 14q$. With similar computation, we can know that the expected norm is around $8q$. This means that many linear combinations of the first d rows will have smaller Euclidean norm than the difference vector $\mathbf{e}$. In other words, there are exponentially many lattice vectors that are closer to the target vector than the hidden vector, thus making decoding extremely difficult.

3.2 Modeling Lattice Attacks

Following the idea of [AFG14a], we consider lattice attacks on (EC)DSA as Unique-SVP instances. In [GN08], it is 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 $\mathcal{L}$, the algorithm will be successful in disclosing a shortest non-zero vector with "high probability" when $\frac{\lambda_2}{\lambda_1} \geq \tau \cdot {\delta_0}^n$ (we call $\frac{\lambda_2}{\lambda_1}$ the gap), where τ is a constant depending both on the nature of the lattices involved and lattice reduction algorithm being used. However, in [GN08], they do not explain what "high probability" means. Therefore, in some subsequent work [AFG14a], the success rate is fixed to some number (10 percent, for example) and the dimension n is taken as the smallest possible in practice in order to achieve the same success rate.

Here we slightly change the model such that τ is not a constant, but a function $\tau_n = \frac{k}{\log(n)}$ (k is some constant) on the dimension n. Besides, we choose BKZ–30 as the lattice reduction algorithm and fix the success rate to be 20%. In the context of this section, the modulus q is 160-bit.

Recall that the lattice we construct is:

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

so the lattice dimension $n = d + 1$, where d is the number of signatures being used. As before, we denote the difference vector between the target vector and the hidden lattice vector as $\mathbf{e}$. Besides, we are using Kannan's embedding method to perform lattice attacks on a larger lattice:

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

Regarded as a Unique-SVP instance, the success rate of the attack crucially depends on the ratio $\frac{\lambda'_2}{\lambda'_1}$, where λ'_1, λ'_2 are the first and second minimum of the embedded lattice $\mathcal{L}(C)$. According to the relation between $\mathcal{L}(B)$ (lattice spanned by the matrix B) and $\mathcal{L}(C)$ (lattice spanned by the matrix C), $\lambda'_1 \approx \|\mathbf{e}\|$ and $\lambda'_2 \approx \lambda_1$, where λ_1 is the first minimum of the original lattice $\mathcal{L}(B)$. Therefore, the success rate of lattice attacks increases as the ratio $\frac{\lambda_1}{\|\mathbf{e}\|}$ increases.

Table 3: Experimental result: gap needed to achieve $\geq 20\%$ success rate.

Leakage l	Signatures d	Gap $\lambda_1/\|\mathbf{e}\|$	Success rate
3	54	0.93	20/100
4	40	0.91	25/100
5	32	0.88	20/100
6	27	0.87	21/100
7	23	0.86	25/100
8	20	0.85	23/100

As previously mentioned, we assume that in order to achieve 20% success rate, the requirement is

$$\text{gap} = \frac{\lambda_1}{\|\mathbf{e}\|} \geq \frac{k}{\log(d+1)} \cdot {\delta_0}^{d+1},$$

where δ_0 is the root Hermite factor and k is some constant. Before proceeding, we have to determine the root Hermite factor as well as the number of signatures d. First we do some experiments to determine the root hermite factor δ_0 for BKZ–30 on this type of lattice (for HNP attack). After doing numerous experiments, we determine that $\delta_0 \approx 1.01$ for BKZ–30. In addition, we take d as the binary length of the modulus $qlen$ divided by the leakage l. Intuitively and information theoretically, one HNP inequality with l-bit leakage contains l bits of information, so in order to recover the secret key α that has $qlen$ bits, at least $\frac{qlen}{l}$ inequalities are necessary.

Table 3 shows the experimental results of the gap ($\frac{\lambda_1}{\|\mathbf{e}\|}$) that is necessary to achieve 20% success rate for 160-bit (EC)DSA. Now we do a linear regression to determine the constant k. After carrying out the regression depicted in Figure 1, we find that $k \approx 3.11$.

Now we estimate the computation cost for the "borderline" case 160-bit (EC)DSA with 2-bit nonce leakage. The dimension $d = \frac{160}{2} = 80$, and the requirement is

$$\text{gap} = \frac{\lambda_1}{\|\mathbf{e}\|} \geq \frac{k}{\log(d+1)} \cdot {\delta_0}^{d+1} = \frac{3.11}{\log(81)} \cdot 1.01^{81} \approx 1.098.$$

According to Gaussian Heuristic, we have

$$\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d+1}{2\pi e}} \text{vol}(\mathcal{L})^{1/(d+1)} = \sqrt{\frac{81}{2\pi e}} \text{vol}(\mathcal{L})^{1/81}.$$

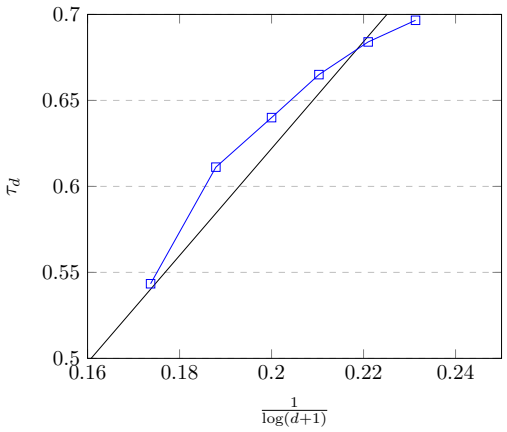

Figure 1: Linear regression to estimate the constant k.

The expected length of $\mathbf{e}$ is roughly $\sqrt{\frac{81}{3}}q$, so we have

$$\frac{\sqrt{\frac{81}{2\pi e}}\mathrm{vol}(\mathcal{L})^{1/81}}{\sqrt{\frac{81}{3}}q} \geq 1.098,$$

which is equivalent to

$$\mathrm{vol}(\mathcal{L}) \geq 2.61^{81} \cdot q^{81},$$

but the real volume of the lattice is $8^{81} \cdot q^{80}$, so the ratio between them is

$$\frac{2.61^{81} \cdot q^{81}}{8^{81} \cdot q^{80}} \approx \frac{q}{2^{130}} \approx 2^{30}.$$

This means that if we could increase the volume of the lattice by 2^{30} times and keep $\|\mathbf{e}\|$ unchanged, then we have 20% success rate for 160-bit modulus with 2-bit nonce leakage. The number 2^{30} somewhat shows the magnitude of computation cost for 2-bit nonce leakage case.

3.3 One Intuitive Idea to Improve the Attacks

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 while keeping $\|\mathbf{e}\|$ almost unchanged. Our attack is directly based on this idea. In later sections, we will show that by brute-forcing some bits of the secret key (or the nonces), we could modify the original lattice and increase the volume of the lattice, while $\|\mathbf{e}\|$ is almost unchanged. Thus, according to the property of Unique-SVP, we will have significantly better success rate.

4 Guessing Bits of Secret Key

In our context, we are considering those "borderline" cases, so in this section, the modulus q has 160 bits and the nonce leakage $l = 2$ (for other moduli, it is similar). In standard lattice attacks, either we find the secret key or get nothing. Even if we set the secret key having only 10 bits, it still does not make lattice attacks any easier (of course, if it has only 10 bits, then we could brute-force the secret key, but it is irrelevant here, since we only care about lattice attacks). Therefore, it is somewhat believed that partial information of the secret key do not help the attack. Perhaps surprisingly, 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 $|\alpha t_i - u_i|_q < q/2^l$ $(i = 1, \ldots, d)$ 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 $(2^{l+1}u_1 + q, 2^{l+1}u_2 + q, \ldots, 2^{l+1}u_d + q, 0)$, and the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \ldots, \alpha 2^{l+1}t_d + c_d 2^{l+1}q, \alpha)$. Again we denote the difference vector between them as $\mathbf{e}$ and we already know that each coefficient of $\mathbf{e}$ is

less than q, so $\|\mathbf{e}\| < q\sqrt{d+1}$. As we have 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 $\|\mathbf{e}\|$ almost 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 the hidden lattice vector will not be close to the target vector anymore, because the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \ldots, \alpha 2^{l+1}t_d + c_d 2^{l+1}q, 2^{100}\alpha)$, and the last coefficient of $\mathbf{e}$ 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 upper-bounded by $2^{160} \approx q$, so this means $\|\mathbf{e}\|$ keeps almost unchanged, and we have increased the volume of the lattice by 2^{100} times, thus making the success probability significantly better. We carry 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 the above matrix B, 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

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

where c is any arbitrary predetermined integer between 1 and 160. Then 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

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

Then substitute α with $\alpha_1 \cdot 2^c + \alpha_2$ and we have

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

Then set

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

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

$$|\alpha_1 \cdot t'_i - u'_i|_q < q/2^l \quad (i = 1, 2, \ldots, d).$$

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 $(\alpha_1 2^{l+1}t'_1 + c_1 2^{l+1}q, \alpha_1 2^{l+1}t'_2 + c_2 2^{l+1}q, \ldots, \alpha_1 2^{l+1}t'_d + c_d 2^{l+1}q, \alpha_1 2^c)$ and the target vector is $(2^{l+1}u'_1 + q, 2^{l+1}u'_2 + q, \ldots, 2^{l+1}u'_d + q, 0)$. Now we have increased

the volume of the lattice by 2^c times while keeping $\|\mathbf{e}\|$ almost unchanged, since $\alpha_1 2^c$ is upper bounded by q. Of course we do not 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 computation cost we want to pay).
- Step 2: Collect d signatures and construct t_i, u_i as previously defined ($i = 1, 2, \ldots, d$).
- 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's embedding method.

With this method, we are able to attack those "borderline" cases: 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. For more detail, see the section of experimental results.

One typical question for this attack would be: *What is the difference between our approach and directly applying BKZ with larger block size?* We make a comparison:

- In some sense, our approach has similar effect as directly applying BKZ with larger block size. While BKZ with larger block size outputs lattice basis with smaller root Hermite factor (thus better chance of finding the vector $\mathbf{e}$), our approach aims to increase the gap for Unique-SVP and have better success rate due to the property of Unique-SVP.
- Our approach is easy to simulate and control. In simulation experiments, we could assume that we have guessed the correct bits, thus avoding the enumeration, which is difficult to carry out in a short time.
- Our approach can be easily parallelized, because each enumeration of bits is independent. While we do not deny the fact that BKZ with larger block size could also be parallelized, it requires another implementation of the SVP oracle (changing the internal code of fplll library [dt20]), which needs a lot of work.

5 Guessing Bits of Nonces

Another similar approach could be made to increase the volume of the lattice. Again in our context, the modulus q has 160 bits, the leakage $l = 2$. For other moduli, 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) $(i = 1, \ldots, d)$ with 2-bit nonce leakage and computed $t_i = |r \cdot 2^{-2} s^{-1}|_q$ and $u_i = |-h(m) \cdot 2^{-2} s^{-1}|_q$ as in 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 we set:

$$t_i' = |r \cdot 2^{-3} s^{-1}|_q \quad \text{and} \quad u_i' = |-h(m) \cdot 2^{-3} s^{-1}|_q.$$

If the third bit is 1, we have:

$$\begin{aligned} k_i &= 2^3 b'_i + 2^2 \\ \alpha r s^{-1} &\equiv 2^3 b'_i + 2^2 - h(m)s^{-1} \pmod q \\ \alpha r s^{-1} 2^{-3} &\equiv b'_i + 2^{-1} - h(m)s^{-1}2^{-3} \pmod q, \end{aligned}$$

and we then set:

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

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 these 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 is the following matrix B.

$$B = \begin{pmatrix} 8q & 0 & \cdots & 0 & 0 \\ 0 & 8q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 8q & 0 \\ 8t_1 & 8t_2 & \cdots & 8t_{90} & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 16q & 0 & \cdots & 0 & 0 \\ 0 & 16q & \cdots & 0 & 0 \\ & \vdots & & \vdots & \\ 0 & 0 & \cdots & 16q & 0 \\ 16t'_1 & 16t'_2 & \cdots & 16t'_{90} & 1 \end{pmatrix}.$$

By guessing one more bit for all the signatures, we can construct the above matrix C with all the inequalities having 3-bit leakage. Of course, with this new matrix, we could attack 160-bit (EC)DSA easily, since we know that for 3-bit leakage, standard lattice attacks work well. However, we are paying a price of 2^{90} for 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:

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

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

This approach can be summarised as the following steps:

- Step 1: Determine integer constant k and 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 and use Kannan's embedding method to find the secret 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 those "borderline" cases: 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. Here we make a comparison with the approach in Section 4. Generally, the approach in Section 4 performs better than the approach in this section. The lattice attack on HNP essentially amounts to decoding a lattice point in a hypercube. When we guess bits of some of the signature nonces, we reduce the length of certain sides of this hypercube. On the contrary, when we guess bits of the secret key, we uniformly shrink the hypercube. For the same exhaustive search cost, the two decoding regions have the same volume, but the average (squared) error length is smaller in the second case.

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 [Ble00]. The main idea of Bleichenbacher's 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 in the community (mentioned by cryptanalysis experts on different occasions, e.g., ECC-17 by Tibouchi [Tib17], Lattice Camp-20 by Heninger [Hen20]): *Is it possible to improve lattice attacks with many more signatures?* We give a solution to this question and again we are in the context of 160-bit modulus with 2-bit nonce leakage.

6.1 From Bleichenbacher to Lattice

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

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

and write the secret key α as

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

Where α_1 is the $(160 - c)$ MSBs of α and α_2 is the remaining LSBs. If t_i $(i = 1, 2, \ldots, d)$ is small enough, $\alpha_2 \cdot t_i$ $(i = 1, 2, \ldots, d)$ 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:

$$|\alpha_1 \cdot 2^c \cdot t_i - u_i|_q < q/2^l \quad (i = 1, 2, \ldots, d).$$

Then construct the lattice as:

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

In this lattice, $(\alpha_1 2^c \cdot 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha_1 2^c \cdot 2^{l+1}t_2 + c_2 2^{l+1}q, \ldots, \alpha_1 2^c \cdot 2^{l+1}t_d + c_d 2^{l+1}q, \alpha_1 2^c)$ 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 $\|\mathbf{e}\|$ almost keeps unchanged, thus making the attack much easier. Note that now we do not do enumeration of bits as in previous sections.

This attack can be summarised as the following steps:

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

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

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

- Step 2: Keep doing step 1 until we get d pairs (t_i, u_i) $(i = 1, \ldots, d)$.
- Step 3: Construct the above lattice and use Kannan's embedding method to do lattice attacks.
- Step 4: Find α_1 which is the $(160 - c)$ MSBs of α.
- Step 5: Find the remaining bits of α (for example, we can construct a HNP instance for the remaining bits).

As we previously discussed, once we have recovered many MSBs of α, recovering the remaining bits becomes pretty easy.

6.2 A Concrete Example

In order to make it clear, we show a concrete example here. For 160-bit (EC)DSA with 2-bit nonce leakage, we collect t which is less than 2^{140} and write the secret key α as:

$$\alpha = \alpha_1 \cdot 2^{10} + \alpha_2 \ (0 \le \alpha_2 < 2^{10})$$

and we have:

$$|\alpha \cdot t_i - u_i|_q < q/2^l \quad (i = 1, 2, \ldots, d),$$
$$|\alpha_1 \cdot 2^{10} \cdot t_i + \alpha_2 \cdot t_i - u_i|_q < q/2^l \quad (i = 1, 2, \ldots, d).$$

Since α_2 is less than 2^{10}, $\alpha_2 \cdot t$ is upperbounded by 2^{150}, $q/2^l$ has about 158 bits, so as long as the value $|\alpha \cdot t_i - u_i|_q$ does not lie on the edge of the interval $(0, q/2^l)$ (which happens with small probability), we could just throw the term $\alpha_2 \cdot t_i$ away and have:

$$|\alpha_1 \cdot 2^{10} \cdot t_i - u_i|_q < q/2^l \quad (i = 1, 2, \ldots, d).$$

In order to collect 90 signatures where $t < 2^{140}$, we have to sample about $90 \cdot 2^{20} \approx 2^{27}$ signatures. The advantage is that we increase the volume of the lattice almost for free (considering the fact that sampling a signature is much more efficient than doing one BKZ–30 operation). Therefore, at the cost of using 2^{27} signatures, we are able to attack 160-bit (EC)DSA with 2-bit nonce leakage in just one BKZ–30 operation, which is significantly faster than previous results. For more detail, 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:

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

Table 4: 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

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. 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. For simplicity, 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 is satisfied, 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, 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 save a lot of computation. With this preprocessing, 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. To the best of our knowledge, there are only a few works that discuss how to choose the factor. For example, in Galbraith's book [Gal12], the embedding factor is set to 1 by default, and in [WAT17], Kannan embedding factor for LWE lattices (very different context) is discussed. Therefore, we give a simple analysis for HNP lattice for completeness. As we can see from Table 4, if the factor is either too small or too large, the success rate becomes very low.

Here we give an explanation why this happens. Denote the Kannan embedding factor as γ. For simplicity, we analyze LLL reduction.

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, and if γ is too large, after LLL reduction on the submatrix B, the exchange condition will not be satisfied, then only one round of size reduction will be performed (reduce the last row from the first (d + 1) rows) and the

Table 5: A comparison of the CVP and SVP approaches.

Modulus	Leakage	Nearest plane	Kannan's embedding method
160-bit	4-bit	37/100	100/100
160-bit	3-bit	0/100	91/100

algorithm terminates. By contrast, if γ is properly valued, the exchange condition will be satisfied 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 happens many times, so many rounds of size reduction will be performed. Therefore, if γ is too large, the lattice will get much less reduced.

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 $\mathbf{v}$ 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 $\alpha \cdot (2^{l+1}t_1, 2^{l+1}t_2, \ldots, 2^{l+1}t_d, 1) - \mathbf{v}$, 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 the CVP and SVP Approaches

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

As we can see from Table 5, for 160-bit (EC)DSA with 4-bit nonce leakage, both approaches work well. However, nearest plane algorithm never succeeds for 3-bit leakage, while Kannan's embedding method works quite well.

Reason for the Gap. Essentially, nearest plane algorithm can be regarded as one round of size reduction in the embedded lattice. 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 be satisfied 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's embedding method will make the target vector more reduced compared with nearest plane algorithm.

9 Experimental Results

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. The source code is available in [Sun21].

9.1 Guessing Bits of Secret Key

As we can see in Figure 2, as the number of guessed bits increases, the success rate increases. Take 160-bit (EC)DSA with 2-bit nonce leakage for example, 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 it gives us a clear impression of the time complexity and this is independent of the machine being used. Besides, it is easy to estimate the practical attack time. For instance, with Ryzen 3970x and batch SVP technique described in section 7, one BKZ–30 operation on a 90-dimensional lattice takes 40 seconds (on average) on a single core, so the expected time is $\frac{2^{14} \cdot 40s}{32} \approx 10200s$, which is several hours.

9.2 Guessing Bits of Nonces

Similarly, 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.

9.3 Improving Lattice Attacks with More Data

Recall that in 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, summarized in Table 6, 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 seem too many in practical setting, but the advantage is that we could recover about 150 MSBs of the secret key in just one BKZ-30 operation.

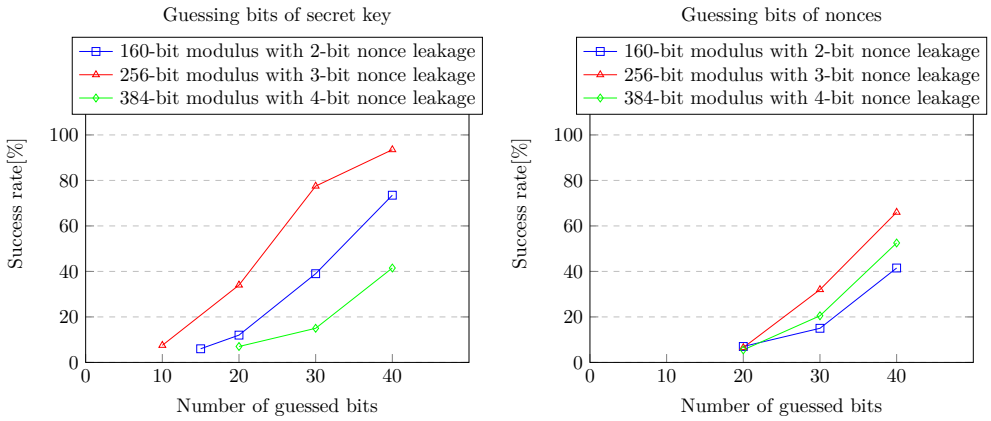

Figure 2: Experimental results: guessing bits of the secret key vs. of the nonces.

Table 6: Utilizing more data to improve lattice attacks.

Modulus	Leakage	Upper bound on t	Signatures	Time complexity	Success rate
160-bit	2-bit	2^{140}	2^{27}	1 BKZ-30	30/200
256-bit	3-bit	2^{226}	2^{37}	1 BKZ-30	27/200
384-bit	4-bit	2^{344}	2^{47}	1 BKZ-30	62/200

9.4 Experiments on the TPM–FAIL Dataset

We also carry 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 use about 40000 signatures and in Minerva [JSSS20], a new technique of geometric assignment of leakage is 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 some LSBs of the secret key and perform lattice attacks described in section 4.

We do 100 experiments and succeed 3 times. In this way, with only 800 signatures available, we are able to recover the secret key for TPM–FAIL dataset. For the most part of this paper, we are in a setting where there is no noise in the sense that leakage is assigned correctly for each signature. However, this is not the case in general in practice. If the number of signatures is enough, it is easy to assign the leakage correctly with high probability, but if $N = 800$, it is unavoidable that some of the assignment are wrong, which is somewhat annoying and makes the success rate very low. There are many robust techniques in Minerva [JSSS20] for dealing with noise, which are very important contribution of that paper. For example, the *random subset technique* in Minerva could be utilized: instead of choosing d out of N signatures, we could choose $1.5d$ signatures and collect a random subset having d elements. Besides, the *CVP + flip technique* can be applied: change $\mathbf{u}$ to correct errors (this part can even be generalized with our nonce guessing technique by flipping more bits). Considering that our work is largely orthogonal and complementary to Minerva and we only use BKZ–30 (which could be replaced with stronger lattice reduction algorithms, e.g., [CN11, AWHT16, ADH+19, EK20, KEF21]), it is fair to say that our approaches help improving the attack.

Acknowledgement

We are thankful to Masaya Yasuda for helpful discussions. We also would like to thank the anonymous reviewers for their useful suggestions and comments.

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 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.

[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.

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

[AWHT16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 789–819. Springer, Heidelberg, May 2016.

[Bab86] László Babai. On Lovász' lattice 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.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2020.

[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.

[EK20] Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm. Cryptology ePrint Archive, Report 2020/694, 2020. https://eprint.iacr.org/2020/694.

[Gal12] Steven D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[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.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[Hen20] Nadia Heninger. Using lattices for cryptanalysis, 2020. https://simons.berkeley.edu/talks/using-lattices-cryptanalysis.

[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.

[KEF21] Paul Kirchner, Thomas Espitau, and Pierre-Alain Fouque. Towards faster polynomial-time lattice reduction. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 760–790, Virtual Event, August 2021. Springer, Heidelberg.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *11th SODA*, pages 937–941. ACM-SIAM, January 2000.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):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.

[NT12] Phong Q. Nguyen and Mehdi Tibouchi. Lattice-based fault attacks on signatures. In Marc Joye and Michael Tunstall, editors, *Fault Analysis in Cryptography*, pages 201–220. Springer, 2012.

[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 *SFCS 1982*, pages 145–152. IEEE, 1982.

[Sun21] Chao Sun. Source code for the algorithms in this paper. https://github.com/security-kouza/Lattice-Attacks-on-EC-DSA, 2021.

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

[Tib17] Mehdi Tibouchi. Attacks on Schnorr signatures with biased nonces, 2017. https://ecc2017.cs.ru.nl/slides/ecc2017-tibouchi.pdf.

[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.

[WAT17] Yuntao Wang, Yoshinori Aono, and Tsuyoshi Takagi. An experimental study of Kannan's embedding technique for the search LWE problem. In Sihan Qing, Chris Mitchell, Liqun Chen, and Dongmei Liu, editors, *ICICS 17*, volume 10631 of *LNCS*, pages 541–553. Springer, Heidelberg, December 2017.