Extended Attacks on ECDSA with Noisy Multiple Bit Nonce Leakages

Shunsuke Osaki$^{1[0000-0002-9221-8095]}$ and Noboru Kunihiro$^{1[0000-0003-1822-7476]}$

University of Tsukuba, Ibaraki, Japan
`s2220571@u.tsukuba.ac.jp`
`kunihiro@cs.tsukuba.ac.jp`

Abstract. It is well known that in ECDSA signatures, the secret key can be recovered if more than a certain number of tuples of random nonce partial information, corresponding message hash values, and signatures are leaked. There exist two established methods for recovering a secret key, namely lattice-based attack and Fourier analysis-based attack. When using the Fourier analysis-based attack, the number of signatures required for the attack can be evaluated through a precise calculation of the modular bias even if the leaked nonce contains errors. Previous works have focused on two cases: error-free cases and the case for the first MSB has errors among all of the nonce leakage. In this study, we extend the technique to the noisy multiple bits case to calculate the precise value of the modular bias for the case that multiple bits (say, l bits from MSB) have errors. Aranha et al. (ACM CCS 2020) introduced a linear programming problem with parameters to evaluate the number of signatures, time, and memory required for a Fourier analysis-based attack. They also employed a SageMath module to optimize the number of signatures and time required for the attack. Furthermore, we show by experiments that 131-bit ECDSA is vulnerable when the first MSB of the nonce is leaked without error and when 2 MSBs are leaked with an error rate 0.1 each, which implies that total error rate is about 0.19. We then show that the latter case requires less signatures to recover the secret key.

Keywords: ECDSA · Fourier analysis-based attack · Side-channel attack

1 Introduction

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a digital signature algorithm that utilizes elliptic curves. It is widely used in various systems such as SSH, SSL/TLS, Bitcoin, and others. Therefore, evaluating the potential for leakage of secret information and the effect it may have on the overall security of a system is critical.

A nonce (Number used only ONCE) is secret information that is randomly generated during the signing process. However, it is possible to leak nonces through side-channel attacks. An attack is reduced to the Hidden Number Problem (HNP) if a certain number of pairs of nonce partial bits, corresponding message hash values, and signatures are available [5]. Lattice-based and Fourier-analysis-based attacks are known as methods that solve HNPs.

A lattice-based attack can find a secret key with a relatively small number of signatures if the MSBs of the nonce are known without errors. If the secret key is 160-bit and the 2 bits in a nonce is leaked [1][7][9]; if the secret key is 256-bit and the 3 bits in a nonce is leaked [1][9]; or if the secret key is 384-bit and the 4 bits in a nonce is leaked [1][9], then several dozens to several thousands of signatures can be used to recover the secret key in a few minutes to hours. Lattice-based attacks require more than 2 bits of nonce information without errors but do not require many signatures.

In a Fourier analysis-based attack, recovering the secret key is possible when the MSBs of the nonce are known without errors. If the key length is 192-bit [3] or 256-bit [11], the signatures can be solved with a 1 or 2 bits leak with small errors, respectively. It was reported that several hundreds of millions of signatures and several days were required to solve the problem using workstations and clusters in those cases. In addition, the attack can also be successful if more MSBs are obtained with errors, but it requires many signatures, computational cost and time.

Aranha et al. [3] found vulnerabilities in OpenSSL 1.0.2 and 1.1.0, etc., against side-channel attacks that leak the MSB of ECDSA nonce, and used these vulnerabilities in their attacks. They estimated the number of signatures and costs of time, and memory of an attack when the 1 bit nonce is leaked with errors by estimating the modular bias. The number of signatures, cost of time, and memory required for the attack are also obtained by using the 4-list sum algorithm for linear combination, which is critical in Fourier analysis-based attacks. They then reduced the problem of optimizing the number of signatures to a linear programming problem and solved it using the Mixed Integer Linear Program module of SageMath to optimize the number of signatures, costs of memory, and time required for the attack [10].

1.1 Our contributions

In this paper, we estimate the number of signatures, costs of time, and memory required for an attack in the case of multiple bits by estimating the modular bias when multiple MSBs with errors are obtained. In previous studies, modular bias has only been formulated for MSB leakage with errors or multiple bit leakage without errors. We have successfully generalized the formulation of the module bias. This allows us to estimate the modular bias in any case and to obtain an estimate of the number of signatures needed to recover a secret key.

We also focus on changes to the number of signatures when the error rate changes. Then, the optimal parameters are selected based on the evaluation of the number of obtained signatures. We extend their optimization program with a generalized modular bias to find the number of signatures required to recover the secret key. We also perform an actual attack against 131-bit ECDSA and confirm that it is possible to recover the secret key. Furthermore, we show from both theoretical analysis for modular bias and experiment that the secret key is successfully recovered with fewer signatures when each of the 2 bits is leaked with an error rate of 0.1 than when the nonce is leaked with 1 bit without error.

2 Preliminaries

2.1 ECDSA signature generation algorithm

The set of solutions $(x, y) \in \mathbb{F} \times \mathbb{F}$ of an elliptic curve E defined over a field F with an infinity point O is a commutative group derived from the chord-and-tangent rule.

The signature generation algorithm of the ECDSA is shown in Algorithm 1. The secret key sk is λ-bit. The secret information (i.e., nonce k) is randomly generated in the first line of Algorithm 1. In this study, we consider the case in which the MSBs of k are leaked.

Algorithm 1 ECDSA signature generation

```
Input: prime number $q$, secret key $\mathrm{sk} \in \mathbb{Z}_q$, message $\mathsf{msg} \in \{0,1\}^*$, base point on
    elliptic curve $G$, and cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}_q$
Output: valid signatures $(r, s)$
1: $k$ is chosen at random from $\mathbb{Z}_q$
2: $R = (r_x, r_y) \leftarrow kG; r \leftarrow r_x \bmod q$
3: $s \equiv (H(\mathsf{msg}) + r \cdot \mathrm{sk}) / k \bmod q$
4: return $(r, s)$
```

2.2 Hidden number problem with errors

The function $\mathrm{MSB}_n(x)$ returns the top n bits of x for a positive integer x. Let b be a positive integer, $\{0,1\}^b$ be a fixed distribution on χ_b, and the error bit sequence e be sampled from χ_b. The probabilistic algorithm $\mathrm{EMSB}_{\chi_b}(x)$ takes x, b as input and returns $\mathrm{MSB}_b(x) \oplus e$. For each $i = 1, \ldots, M$, let z_i be $z_i \equiv k_i - h_i \cdot \mathrm{sk} \bmod q$ and h_i, k_i be uniform random values on $\mathbb{Z}_q$. The HNP is the problem of finding sk that satisfies the aforementioned equations given the $h_i, z_i, \mathrm{EMSB}_{\chi_b}(k_i)$ obtained for each $i = 1, \ldots, M$.

The ECDSA signature (r, s) is generated according to Algorithm 1, nonce $k \in \mathbb{Z}_q$ is chosen uniformly at random, and $s \equiv (H(\mathsf{msg}) + r \cdot \mathrm{sk}) / k \pmod q$ is satisfied. This yields the following equation.

$$H(\mathsf{msg}) / s \equiv k - (r/s) \cdot \mathrm{sk} \bmod q$$

If the MSBs of k are obtained, we obtain an instance of HNP as $z \equiv H(\mathsf{msg}) / s \pmod q$ and $h \equiv r/s \pmod q$,

2.3 Bias function and sample bias

We follow the idea of [3] and first show definitions of bias function and sample bias.

Definition 1. *Let $\boldsymbol{K}$ be a random variable over $\mathbb{Z}_q$. The modulus bias $B_q(\boldsymbol{K})$ is defined as*

$$B_q(\boldsymbol{K}) = \boldsymbol{E}\left[\exp\left((2\pi \boldsymbol{K}/q)\,\mathrm{i}\right)\right]$$

Let $\boldsymbol{E}(\boldsymbol{K})$ denote the expected value of random variable $\boldsymbol{K}$ and let i *be an imaginary unit. In the same way, the sample bias of the set of points $K = \{k_i\}_{i=1}^{M}$ in $\mathbb{Z}_q$ is defined as*

$$B_q(K) = \frac{1}{M}\sum_{i=1}^{M} \exp\left((2\pi k_i/q)\,\mathrm{i}\right) \tag{1}$$

By fast Fourier transform (FFT), the computational complexity is $O(M \log M)$. For some positive integer l, let the higher l bits of $\boldsymbol{K}$ be fixed to a certain constant, and the remaining $(\lambda - l)$ bits be random. The following equation is given in [11].

$$\lim_{q\to\infty} |B_q(\boldsymbol{K})| = \frac{2^l}{\pi} \cdot \sin\left(\frac{\pi}{2^l}\right) \tag{2}$$

If no bits are fixed, its absolute value of sample bias is estimated as $1/\sqrt{M}$. In addition, we can easily see that $\lim_{l\to\infty}\lim_{q\to\infty} |B_q(\boldsymbol{K})| = 1$ from Equation (2).

The following lemma is given in [3].

Lemma 1. *Suppose that the random variable $\boldsymbol{K}$ follows the following distribution on $\mathbb{Z}_q$ for $b \in \{0,1\}$, all $\varepsilon \in [0, 1/2]$ and even $q > 0$.*

$$\begin{cases} \Pr[\boldsymbol{K} = k_i] = (1-b)\cdot\frac{1-\varepsilon}{q/2} + b\cdot\frac{\varepsilon}{q/2} & \text{if } \ 0 \le k_i < q/2 \\ \Pr[\boldsymbol{K} = k_i] = b\cdot\frac{1-\varepsilon}{q/2} + (1-b)\cdot\frac{\varepsilon}{q/2} & \text{if } \ q/2 \le k_i < q \end{cases}$$

Letting $\boldsymbol{K}_b$ be a uniform distribution over $[bq/2, (b+1)q/2)$, the modular bias of $\boldsymbol{K}$ is given by

$$B_q(\boldsymbol{K}) = (1-2\varepsilon)\, B_q(\boldsymbol{K}_b)\,. \tag{3}$$

It can be easily verified that $|B_q(\boldsymbol{K}_0)| = |B_q(\boldsymbol{K}_1)|$. Note that Equation (3) considers only 1 bit leakage. The absolute value of $B_q(\boldsymbol{K})$ is given by

$$|B_q(\boldsymbol{K})| = (1-2\varepsilon)\cdot\frac{2}{\pi}\sin\frac{\pi}{2}. \tag{4}$$

2.4 Fourier analysis-based attack

Bleichenbacher introduced Fourier analysis based attack in [4]. First, we consider a naive search method to obtain the secret key sk using the bias function, which is shown in Algorithm 2. Let M be the number of signatures obtained. In the case in which the input sample $\{(z_i, h_i)\}_{i=1}^{M}$ is biased K_i, we randomly select a candidate secret key $w \in \mathbb{Z}_q$ and then calculate $K_w = \{z_i + h_i w \bmod q\}_{i=1}^{M}$. Next, we compute $|B_q(K_w)|$ under Equation (1). If $w = \mathrm{sk}$, then K_w is biased and $|B_q(K_w)|$ has the peak. Then finding the correct key sk is possible. However, this method is inefficient because it must search w in all $\mathbb{Z}_q$.

Algorithm 2 Naive search

```
Input: $(h_i, z_i)_{i=1}^{M}$: HNP samples over $\mathbb{Z}_q$
Output: Correct secret key sk
1: // Select a candidate $w$ for the secret key.
2: for $w = 1$ to $q - 1$ do
3:     Calculate $K_w = \{z_i + h_i w \bmod q\}_{i=1}^{M}$.
4:     Calculate $|B_q(K_w)|$.
5: end for
6: return $w$ which maximizes $|B_q(K_w)|$.
```

De Mulder et al. [8] and Aranha et al. [2] proposed a method to efficiently search for a secret key without performing an exhaustive search. Their methods perform a linear combination of input samples to satisfy $h'_j < L_{\mathrm{FFT}}$ until M' samples are obtained. Consequently, a new linear combined sample $\left\{\left(h'_j, z'_j\right)\right\}_{j=1}^{M'}$ is generated. The width of the peak w is extended from 1 to approximately q/L_{FFT}, showing that recovering the higher $\log L_{\mathrm{FFT}}$ bits of the secret key is possible. In a Fourier analysis-based attack, the entire secret key is recovered by repeating this process.

Let λ' be the number of already recovered bits in sk. At the first step of Fourier analysis-based attack, $\lambda' = \log L_{\mathrm{FFT}}$. Letting the higher λ' bits of sk be $\mathrm{sk_{hi}}$ and the unknown lower $(\lambda - \lambda')$ bits be $\mathrm{sk_{lo}}$, sk can be expressed as $\mathrm{sk} = 2^{\lambda-\lambda'}\mathrm{sk_{hi}} + \mathrm{sk_{lo}}$. Thus, the new HNP formula for the case in which the higher λ' bits of sk has already been recovered can be rewritten as

$$\begin{aligned} k &\equiv z + h \cdot \left(2^{\lambda-\lambda'}\mathrm{sk_{hi}} + \mathrm{sk_{lo}}\right) \bmod q \\ k &\equiv z + h \cdot 2^{\lambda-\lambda'}\mathrm{sk_{hi}} + h \cdot \mathrm{sk_{lo}} \bmod q \\ k &\equiv \hat{z} + h \cdot \mathrm{sk_{lo}} \bmod q, \end{aligned}$$

where $\hat{z} = z + h \cdot 2^{\lambda-\lambda'}\mathrm{sk_{hi}}$. Thus, we obtain the new HNP samples $\{(\hat{z}_i, h_i)\}_{i=1}^{M}$. When the Fourier analysis-based attack is repeated, λ' increases. The $\hat{z}$ is updated in each repetition and, finally, the whole of sk can be recovered.

Algorithm 3 shows Bleichenbacher's attack framework for a Fourier analysis-based attack. The range reduction phase of the algorithm considers two constraints on linear combinations for efficient key searches, namely, small and sparse linear combinations.

Algorithm 3 Bleichenbacher's attack framework

```
Input: $\{(h_i, z_i)\}_{i=1}^{M}$: Sample of HNP over $\mathbb{Z}_q$. $M'$: Number of linear combinations to find. $L_{\mathrm{FFT}}$: FFT table size.
Output: $\mathrm{MSB}\,(\mathrm{sk})_{\log L_{\mathrm{FFT}}}$.
 1: Range reduction
 2: For all $j \in [1, M']$, the coefficients are $\omega_{i,j} \in \{-1, 0, 1\}$, and the linear combination pairs are denoted as $\left(h'_j, z'_j\right) = \left(\sum_i \omega_{i,j} h_i, \sum_i \omega_{i,j} z_i\right)$. In this case, we generate $M'$ sample $\left\{\left(h'_j, z'_j\right)\right\}_{j=1}^{M'}$ that satisfies the following two conditions.
      (1) Small : $0 \le h'_j < L_{\mathrm{FFT}}$.
      (2) Sparse : $|B_q(K)|^{\Omega_j} \gg 1/\sqrt{M'}$, where $\Omega_j := \sum_i |\omega_{i,j}|$ for all $j \in [1, M']$.
 3: Bias computation
 4: $Z := (Z_0, \ldots Z_{L_{\mathrm{FFT}}-1}) \leftarrow (0, \ldots, 0)$
 5: for $j = 1$ to $M'$ do
 6:     $Z_{h'_j} \leftarrow Z_{h'_j} + \exp\left(\left(2\pi z'_j/q\right)\mathrm{i}\right)$
 7: end for
 8: Let $w_i = iq/L_{\mathrm{FFT}}$, $\{B_q(K_{w_i})\}_{i=0}^{L_{\mathrm{FFT}}-1} \leftarrow \mathrm{FFT}\,(Z)$
    $= \left(B_q(K_{w_0}), B_q(K_{w_1}), \ldots, B_q\left(K_{w_{L_{\mathrm{FFT}}-1}}\right)\right)$.
 9: Find $i$ that maximizes $|B_q(K_{w_i})|$.
10: return $\mathrm{MSB}\,(w_i)_{\log L_{\mathrm{FFT}}}$.
```

In the small linear combination constraint, it should be satisfied that $\omega_{i,j} \in \{-1, 0, 1\}$ and $h'_j = \sum_{i=1}^{M} \omega_{i,j} h_i < L_{\mathrm{FFT}}$. This constraint is used to reduce the search range by linear combinations. To enable h'_j to be smaller, we can take linear combinations with a greater number of h_i (i.e., a fewer number of $\omega_{i,j} = 0$). The fewer the number of linear combinations, the smaller L_{FFT} becomes, and thus the width of the peak, q/L_{FFT} increase. However, if too many linear combinations are taken, the peak value decreases exponentially. Although the original peak value is $|B_q(\boldsymbol{K})|$, the peak bias after linear combinations is $|B_q(\boldsymbol{K})|^{\Omega_j}$, due to constraint, which exponentially decreases if we take Ω_j linear combinations. If the peak value is sufficiently larger than the average of the noise $1/\sqrt{M'}$, it can be distinguished. Therefore, constraints are imposed as sparse linear combinations to distinguish them from noise values.

The constraints of sparse linear combinations limit the number of linear combinations that can be taken such that the peak value is prevented from becoming too small. Now, estimating the number of samples M' after the linear combination (assuming that Ω_j is constant) depends only on $|B_q(\boldsymbol{K})|$, and finding the modular bias in a rigorous manner is critical. In a Fourier analysis-based attack, bias computation is performed using FFT, which has a computational complexity of $O\left(L_{\mathrm{FFT}} \log L_{\mathrm{FFT}}\right)$ and can thus be calculated efficiently. However, range reduction is not known to be inefficient and requires considerable computational time. Table 3 in [3] shows that the bias computation (FFT) consumes 1 hour, but range reduction (collision) consumes 42 hours when the key length is 162-bit, and the nonce is 1 bit leak with $\varepsilon = 0.027$.

2.5 $\mathcal{K}$-list sum problem

Let the birthday problem be the problem of choosing $x_1 \in \mathcal{L}_1$ and $x_2 \in \mathcal{L}_2$ from 2 lists $\mathcal{L}_1$ and $\mathcal{L}_2$ with random n bits elements that satisfy $x_1 \oplus x_2 = 0$. In addition, given a list of $\mathcal{K}$ with n bits values, the problem of selecting 1 of elements from each list and finding a pair of values for which the XOR of those $\mathcal{K}$ values is 0 is known as the Generalized Birthday Problem (GBP). In [12], Bleichenbacher observed similarities between GBP and the Fourier analysis-based attack [4]. The $\mathcal{K}$-list sum algorithm solves $\mathcal{K}$-list sum problem [6] which is the GBP subproblem.

Aranha et al. [3] used the $\mathcal{K}$-list sum algorithm to increase the number of samples while increasing the widths of peaks through linear combination. Algorithm 4 shows a 1-fold 4-list sum algorithm. Algorithm 4 first finds the pairs from two of the given four lists such that the higher a bits of the sum is a certain value, and it stores the sum in sorted lists $\mathcal{L}'_1$ and $\mathcal{L}'_2$. Next, from $\mathcal{L}'_1$ and $\mathcal{L}'_2$, select a pair (x'_1, x'_2) whose higher n bits are equal and calculate the absolute difference $|x'_1 - x'_2|$, where the higher n bits are 0. We then obtain sorted lists with $(\lambda - n)$ bits elements. Because the higher a bits are first chosen to be equal, we only need to check whether $(a - n)$ bits are equal. The algorithm increases the $M = 2^m = 2^{a+2}$ sequences of length λ received as input to 2^{3a+v-n} sequences of length $(\lambda - n)$ by linear combination.

Algorithm 4 Parameterized 4-list sum algorithm based on Howgrave–Graham–Joux

```
Input: $\{\mathcal{L}_i\}_{i=1}^4$: Sorted list of uniform random samples of $\lambda$ bits uniform random
    samples of length $2^a$. $n$: Number of higher bits to be discarded in each round.
    $v \in [0, a]$: Parameter
Output: $\mathcal{L}'$: List of $(\lambda - n)$-bit samples
 1: For each $c \in [0, 2^v)$:
    (a) Search for a pair $(x_1, x_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ satisfying $\mathrm{MSB}_a(x_1 + x_2) = c$. Output a
        new sorted list $\mathcal{L}'_1$ with $x_1 + x_2$ as $2^a \cdot 2^a \cdot 2^{-a} = 2^a$ elements. Similarly, for
        $\mathcal{L}_3, \mathcal{L}_4$, the sorted list $\mathcal{L}'_2$ is obtained.
    (b) Search for a pair $(x'_1, x'_2) \in \mathcal{L}'_1 \times \mathcal{L}'_2$ satisfying $\mathrm{MSB}_n(|x'_1 - x'_2|) = 0$. Output a
        new sorted list $\mathcal{L}'$ with $|x'_1 - x'_2|$ as $2^a \cdot 2^a \cdot 2^{-(n-a)} = 2^{3a-n}$ elements.
 2: return $\mathcal{L}'$
```

Algorithm 5 is an iterative 4-list sum algorithm that calls Algorithm 4 as a subroutine. If 2^a is the length of each sublist, it can be expressed as $M = 2^m = 4 \cdot 2^a = 2^{a+2}$. Let n be the number of higher bits to be nullified, and let $N = 2^n$. $M' = 2^{m'} < 2^{2a}$ is the number of samples output with the higher n bits as 0. In addition, v is the number of iterations in range reduction with $v \in [0, a]$, and $T = 2^t = 2^{a+v}$ and T is the time complexity. From [6], it holds that $TM^2 = N$. Now, the N is $2^4 M' N$ and therefore the following holds.

$$2^4 M' N = TM^2 \tag{5}$$

From Equation (5), we obtain

$$m' = 3a + v - n \tag{6}$$

Let r be the number of times the attacker repeats the 4-list sum algorithm. By iterating, find a small linear combination of 4^r integers that satisfies the budget parameter of the FFT table so that it is less than $L_{\text{FFT}} = 2^{\ell_{\text{FFT}}}$ and so that the FFT computation is tractable. In this case, the trade-off equation for each round $i = 0, \ldots, r-1$ can be rewritten as

$$m'_i = 3a_i + v_i - n_i, \tag{7}$$

where $m_{i+1} = m'_i$. The output of the i-th round is used for the input of the $i+1$-th round.

Table 1 lists the constraints of a linear programming problem when Algorithm 5 is optimized in terms of time, memory, and the number of signatures. Consider the optimization case in which m_{in} is minimized. Let $t_{\max}$ be the maximum time spent in each round, $m_{\max}$ be the maximum memory, and $\ell_{\text{FFT}} = \log L_{\text{FFT}}$ be the memory size for the FFT. These are quantities determined by the amount that can be spent (i.e., cost). The α is a slack parameter that enables the peak to be more observable and depends on the maximum possible noise value. This value can be estimated by examining the distribution of $\left\{h'_j\right\}_{j=1}^{M'}$ and is given by approximately $\sqrt{2\ln\left(2L_{\text{FFT}}/\varepsilon\right)}$ [3].

Letting $m_r := \log M'$, $m_r = 2\left(\log\alpha - 4^r \log\left|B_q\left(\boldsymbol{K}\right)\right|\right)$ is derived from the constraint of sparse linear combinations. Estimating $\left|B_q\left(\boldsymbol{K}\right)\right|$ is sufficient to estimate the number of samples M' required after linear combination. In addition, $\left|B_q\left(\boldsymbol{K}\right)\right|$ is the only value related to the number of bits l in the leaked nonce. Depending on the length λ of the secret key, each n_i is differently chosen and the choice of n_is affects other parameters.

Algorithm 5 Iterative HGJ 4-list sum algorithm

```
Input: $\mathcal{L}$: List of $M = 4 \times 2^a$ uniforml random $\lambda$-bit samples. $\{n_i\}_{i=0}^{r-1}$: The number of
    higher bits to be discarded in each round. $\{v_i\}_{i=0}^{r-1}$: Parameters where $v_i \in [0, a_i]$.
Output: $\mathcal{L}'$: List of $\left(\lambda - \sum_{i=0}^{r-1} n_i\right)$-bit samples with length $2^{m_r}$.
 1: Let $a_0 = a$.
 2: For each $i = 0, \ldots, r-1$ :
     (a) Divide $\mathcal{L}$ into four lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ of length $2^{a_i}$ and sort each list.
     (b) Give parameters $n_i$ and $v_i$ and $\{\mathcal{L}_i\}_{i=1}^4$ to Algorithm 4. Obtain a single list $\mathcal{L}'$
         of length $2^{m_{i+1}} = 2^{3a_i+v_i-n_i}$. Let $\mathcal{L} := \mathcal{L}'$ and $a_{i+1} = m_{i+1}/4$.
 3: return $\mathcal{L}'$
```

Table 1. Linear programming problem based on iterative HGJ 4-list sum algorithm (Algorithm 5). Each column is a constraint to optimize time and space and data [3].

	Time	Space	Data
minimize	$t_0 = \ldots = t_{r-1}$	$m_0 = \ldots = m_{r-1}$	m_{in}
subject to	—	$t_i \le t_{\max}$	$t_i \le t_{\max}$
subject to	$m_i \le m_{\max}$	—	$m_i \le m_{\max}$
subject to	$m_{i+1} = 3a_i + v_i - n_i$		$i \in [0, r-1]$
	$t_i = a_i + v_i$		$i \in [0, r-1]$
	$v_i \le a_i$		$i \in [0, r-1]$
	$m_i = a_i + 2$		$i \in [0, r-1]$
	$m_{i+1} \le 2a_i$		$i \in [0, r-1]$
	$m_{\text{in}} = m_0 + f$		
	$\lambda \le \ell_{\text{FFT}} + f + \sum_{i=0}^{r-1} n_i$		
	$m_r = 2\left(\log\alpha - 4^r \log\left(\lvert B_q(\boldsymbol{K})\rvert\right)\right)$		

3 Modular bias for multiple bit leakage

Aranha et al. [3] discussed the security of ECDSA only for 1 bit noisy leakage. Considering practical circumstances, more bit leakage can be obtained. This section will analyze the security for the case where more noisy bits are obtained.

3.1 Modular bias for 2 bits leakage

We extend the evaluation of the modular bias for a single noisy bit case presented in Equation (3) to one when the nonce leaks multiple bits with errors. We begin with the most simple case: modular bias for $l = 2$ and extend the result for general l. The modular bias is also given for the case in which each bit has a different error rate. The nonce obtained by a side-channel attack is not necessarily completely error-free. Thus far, evaluation of the case of nonce leakage with errors has been limited to the case of 1 bit leakage as done by Aranha et al. [3]. This work allows us to evaluate and discuss the security of the ECDSA in more detail by estimating the modular bias in the case of multiple bit leakage.

Lemma 2 (Modular bias for $l = 2$). *Suppose that the random variable $\boldsymbol{K}$ follows the following distribution over $\mathbb{Z}_q$ for $b \in \{0, 1, 2, 3\}$, $\varepsilon_1, \varepsilon_2 \in [0, 1/2]$ and* *even $q > 0$.*

$$\begin{cases}
\Pr\left[\boldsymbol{K} = k_i\right] & = \frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4} + \frac{b(2-b)(3-b)}{2} \cdot \frac{\varepsilon_1 \varepsilon_2}{q/4} \\
 & -b\left(1-b\right)\left(3-b\right) \cdot \frac{\varepsilon_1(1-\varepsilon_2)}{q/4} + \frac{b(1-b)(2-b)}{6} \cdot \frac{(1-\varepsilon_1)\varepsilon_2}{q/4} \ \text{ if } \ 0 \leq k_i < q/4 \\
\Pr\left[\boldsymbol{K} = k_i\right] & = \frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{(1-\varepsilon_1)\varepsilon_2}{q/4} + \frac{b(2-b)(3-b)}{2} \cdot \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4} \\
 & -b\left(1-b\right)\left(3-b\right) \cdot \frac{\varepsilon_1 \varepsilon_2}{q/4} + \frac{b(1-b)(2-b)}{6} \cdot \frac{\varepsilon_1(1-\varepsilon_2)}{q/4} \ \text{ if } \ q/4 \leq k_i < q/2 \\
\Pr\left[\boldsymbol{K} = k_i\right] & = \frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\varepsilon_1(1-\varepsilon_2)}{q/4} + \frac{b(2-b)(3-b)}{2} \cdot \frac{(1-\varepsilon_1)\varepsilon_2}{q/4} \\
 & -b\left(1-b\right)\left(3-b\right) \cdot \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4} + \frac{b(1-b)(2-b)}{6} \cdot \frac{\varepsilon_1 \varepsilon_2}{q/4} \ \text{ if } \ q/2 \leq k_i < 3q/4 \\
\Pr\left[\boldsymbol{K} = k_i\right] & = \frac{(1-b)(2-b)(3-b)}{6} \cdot \frac{\varepsilon_1 \varepsilon_2}{q/4} + \frac{b(2-b)(3-b)}{2} \cdot \frac{\varepsilon_1(1-\varepsilon_2)}{q/4} \\
 & -b\left(1-b\right)\left(3-b\right) \cdot \frac{(1-\varepsilon_1)\varepsilon_2}{q/4} + \frac{b(1-b)(2-b)}{6} \cdot \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4} \ \text{ if } \ 3q/4 \leq k_i < q
\end{cases}$$

Let $\boldsymbol{K}_b$ be a uniform distribution over $[bq/4, (b+1)\,q/4)$. The modular bias of $\boldsymbol{K}$ is then given by

$$B_q\left(\boldsymbol{K}\right) = \left\{\left(1-2\varepsilon_1\right)\left(1-\varepsilon_2\right) + \mathrm{i}\left(1-2\varepsilon_1\right)\varepsilon_2\right\} B_q\left(K_b\right).$$

Proof. See Appendix A.

Remark 1. We now consider the case in which $\varepsilon_2 = 0.5$, (i.e., the same case in which no bias exists in the second bit, which is completely random). In this case, the absolute value of the bias is given by

$$\left|B_q\left(\boldsymbol{K}\right)\right| = \left|\left(1-2\varepsilon_1\right) \times 0.5 + \mathrm{i}\left(1-2\varepsilon_1\right) \times 0.5\right| \cdot \frac{2^2}{\pi} \sin\frac{\pi}{2^2} = \left(1-2\varepsilon_1\right) \cdot \frac{2^1}{\pi} \sin\frac{\pi}{2^1}.$$

We can easily verify that the value is equal to Equation (4), which is the expression for $l = 1$. In addition, it is better to point out that the bias is 0 regardless of the value of ε_2 in the case of $\varepsilon_1 = 0.5$.

3.2 Generalization to modular bias for multiple bit leakage

We next generalize the modular bias to the case in which the higher l bits of the nonce leaks with errors. To simplify the discussion, consider the case where each bit contains an error with probability ε. Given l, let $\boldsymbol{K}_b$ be a uniform distribution over $\left[bq/2^l, (b+1)\,q/2^l\right)$. $b \in \left\{0, 1, \ldots, 2^l - 1\right\}$. We can easily verify that all of $\left|B_q\left(\boldsymbol{K}_b\right)\right|$ are equal regardless of the value of b. Therefore, it is enough to obtain $B_q\left(\boldsymbol{K}_0\right)$. Let $\mathcal{H}\left(j\right)$ be the Hamming weight when j is expressed in binary. If the higher l bits of the nonce are all 0 and no errors occur, $\boldsymbol{K}_0$ corresponding to $b = 0$ is uniformly distributed over $\left[0, q/2^l\right)$. When an error is contained in each bit with probability ε, each bit is 1 with probability ε. Thus, the number of bits containing errors is the same as the number of 1 bits and can be expressed in terms of Hamming weights. In addition, the number of error-free bits is $l - \mathcal{H}\left(j\right)$. From this, for the higher l bits, if an error occurs in each bit with an error rate of ε, the modular bias is expressed as

$$\left\{\sum_{j=0}^{2^l-1} \exp\left(\frac{2j\pi}{2^l}\mathrm{i}\right) \varepsilon^{\mathcal{H}(j)} \left(1-\varepsilon\right)^{l-\mathcal{H}(j)}\right\} B_q\left(\boldsymbol{K}_0\right). \tag{8}$$

We next simplify the term $\sum_{j=0}^{2^l-1} \exp\left(2j\pi\mathrm{i}/2^l\right) \varepsilon^{\mathcal{H}(j)} \left(1-\varepsilon\right)^{l-\mathcal{H}(j)}$ appeared in Equation (8).

$$\begin{aligned}
&\sum_{j=0}^{2^l-1} \exp\left(\frac{2j}{2^l}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(j)} \left(1-\varepsilon\right)^{l-\mathcal{H}(j)} \\
&= \sum_{j=0}^{2^{l-1}-1} \exp\left(\frac{4j}{2^l}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(2j)} \left(1-\varepsilon\right)^{l-\mathcal{H}(2j)} + \sum_{j=0}^{2^{l-1}-1} \exp\left(\frac{4j+2}{2^l}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(2j+1)} \left(1-\varepsilon\right)^{l-\mathcal{H}(2j+1)} \\
&= \sum_{j=0}^{2^{l-1}-1} \exp\left(\frac{2j}{2^{l-1}}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(2j)} \left(1-\varepsilon\right)^{l-1+1-\mathcal{H}(2j)} \\
&+ \sum_{j=0}^{2^{l-1}-1} \exp\left(\frac{2j}{2^{l-1}}\pi\mathrm{i} + \frac{2}{2^l}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(2j)+1} \left(1-\varepsilon\right)^{l-(\mathcal{H}(2j)+1)} \\
&= \sum_{j=0}^{2^{l-1}-1} \exp\left(\frac{2j}{2^{l-1}}\pi\mathrm{i}\right) \varepsilon^{\mathcal{H}(j)} \left(1-\varepsilon\right)^{l-1-\mathcal{H}(j)} \times \left\{\left(1-\varepsilon\right) + \varepsilon \exp\left(\frac{2}{2^l}\pi\mathrm{i}\right)\right\} \\
&= \prod_{j=1}^{l} \left(\left(1-\varepsilon\right) + \varepsilon \exp\left(\frac{2\pi\mathrm{i}}{2^j}\right)\right)
\end{aligned}$$

During the equation transformation, we use the equation $\mathcal{H}\left(2j+1\right) = \mathcal{H}\left(j\right)+1$ for a non-negative integer j. Note that in the case of $b = 0$, we just consider the Hamming distance to the binary representation $00\cdots0$ of the l-bit. In the general b case, we slightly modify to consider the Hamming distance to the binary representation of b. From this, the bias with error is expressed by the following theorem.

Theorem 1. *The modular bias for the l-bit nonce leakage with error rate ε is given by*

$$\prod_{j=1}^{l} \left(\left(1-\varepsilon\right) + \varepsilon \exp\left(\frac{2\pi\mathrm{i}}{2^j}\right)\right) B_q\left(\boldsymbol{K}_b\right). \tag{9}$$

For the absolute value of the modular bias, the following holds and can be expressed without using complex numbers.

Corollary 1. *The absolute value of the modular bias for the l-bit nonce leakage with error rate ε is given by*

$$\begin{aligned}
&\left|\prod_{j=1}^{l} \left(\left(1-\varepsilon\right) + \varepsilon \exp\left(\frac{2\pi\mathrm{i}}{2^j}\right)\right)\right| \left|B_q\left(\boldsymbol{K}_b\right)\right| \\
&= \sqrt{\prod_{j=1}^{l} \left(1 - 4\varepsilon\left(1-\varepsilon\right)\sin^2\frac{\pi}{2^j}\right)} \left|B_q\left(\boldsymbol{K}_b\right)\right|.
\end{aligned} \tag{10}$$

Here, a simple calculation confirms that the absolute value of the modular bias is 0 in Equation (8)–(10) if $\varepsilon = 0.5$.

Corollary 1 can be used to find the absolute value of the modular bias for a given number of bits and the leakage error rate. The concrete values are shown in Table 2. Each column is the number of bits leaked by the nonce, and each row is the value of the nonce's error rate.

Only the values for $\varepsilon = 0$ are shown in Table 1 of [8]. Only the values for $l = 1$ are shown in Lemma 4.2 of [3]. With the help of Corollary 1, we can calculate the precise absolute value of the modilar bias for arbitrary ε and l (as shown in yellow in the table).

These values are extended to Figure 1 shows the modular bias plotted for each error rate. We can find that the value increases as l increases and depends on the error rate. It converges to some value that depends on the error rate ε at approximately $l = 6$. Moreover, we can see that the graph for $\varepsilon = 0.01$ has almost the same shape as that for $\varepsilon = 0$. In [3], they attacked in $\varepsilon = 0.01$ and $\varepsilon = 0.027$ cases and succeeded in recovering the secret keys.

The modular bias for $\varepsilon = 0.1$ and $l \geq 2$ is larger than that for $\varepsilon = 0$ and $l = 1$. This means that the number of signatures for a 2 bits leak with an error rate of 0.1 is less than that for a 1-bit leak with no errors. Thus, fewer signatures are required for a successful attack. We give experimental reults comaring two cases in 4.2.

Table 2. Absolute values of modular bias

l	1	2	3	4	5	6
$\varepsilon = 0$	0.6366	0.9003	0.9749	0.9935	0.9983	0.9999
$\varepsilon = 0.01$	0.6238	0.8735	0.9427	0.9605	0.9649	0.9660
$\varepsilon = 0.1$	0.5092	0.6522	0.6870	0.6957	0.6978	0.6984
$\varepsilon = 0.3$	0.2546	0.2742	0.2780	0.2788	0.2790	0.2791
$\varepsilon = 0.4$	0.1273	0.1298	0.1302	0.1303	0.1304	0.1304

3.3 Case for different error rate of each bit

Equation (10), as presented in Section 3.2, shows the absolute value of the modular bias for which the error rates of each bit are equal (say, ε). We next show the modular bias for the different error rates of each bit.

Again, we consider $\boldsymbol{K}_0$ and we attempt to update the value corresponding to $\prod_{j=1}^{l} \left((1-\varepsilon) + \varepsilon \exp\left(2\pi \mathrm{i}/2^j\right)\right)$ in Equation (9). The values up to $j = 1$ and $j = 2$ are $(1-\varepsilon) - \varepsilon$ and $((1-\varepsilon) - \varepsilon)((1-\varepsilon) + \varepsilon \mathrm{i})$, respectively. Thus, we are considering cases when the MSBs do not contain errors and when MSBs contain errors. Multiplying by $1 - \varepsilon$ and $\varepsilon \mathrm{i}$ enables us to consider those cases in which the second MSB error is not included and when it is included, respectively. In general j, $1 - \varepsilon$ and $\varepsilon \exp\left(2\pi \mathrm{i}/2^j\right)$ can be considered as the error-free

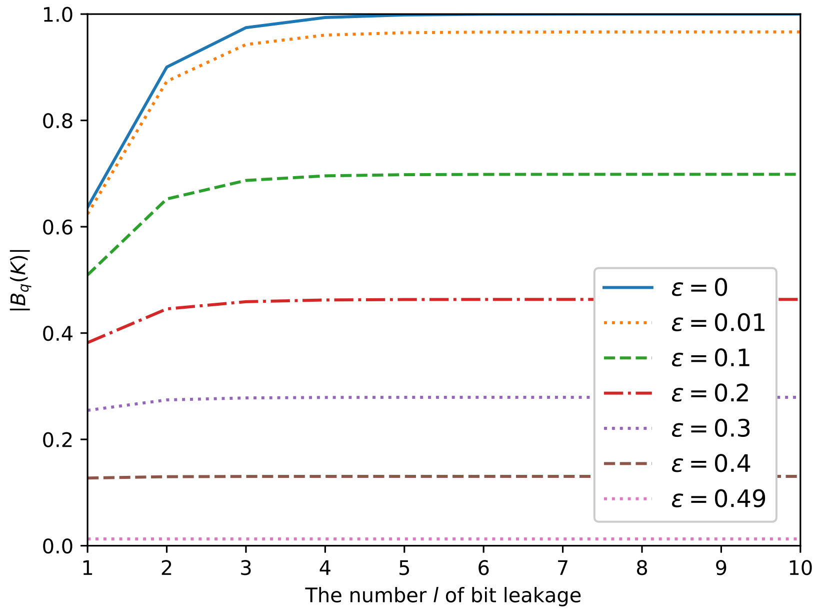


Fig. 1. Modular bias under multiple bit leakage with errors.

bits and error-containing bits, respectively. In other words, at the j-th factor, $((1-\varepsilon)-\varepsilon)\cdots\left((1-\varepsilon)+\varepsilon\exp\left(2\pi\mathrm{i}/2^j\right)\right)$ is considered as each 2^{j-1} combination of the $(j-1)$ bits from the MSB to the $(j-1)$-th bit, with and without errors. Therefore, to establish the case in which the j-th bit does not include an error, we multiply by $1-\varepsilon$. To create the case where the j-th bit contains an error, we multiply by $\varepsilon\exp\left(2\pi\mathrm{i}/2^j\right)$. From this, we can say that the j-th ε represents the error rate of the j-th bit from the MSB. If the error rate of each leaked bit in the nonce is different, we denote ε_j as the error rate of the j-th MSB. The modular bias in the case in which the error rate is different for each bit of Theorem 1 is as in the following theorem.

Theorem 2. *The modular bias when the nonce leaks l-bit with an error rate ε_j is given by*

$$\prod_{j=1}^{l}\left((1-\varepsilon_j)+\varepsilon_j\exp\left(\frac{2\pi\mathrm{i}}{2^j}\right)\right)B_q\left(\boldsymbol{K}_b\right). \tag{11}$$

In the case of $l=2$, it matches Lemma 2.

The absolute value of the modular bias for the different error rates of each bit is given by

$$\sqrt{\prod_{j=1}^{l}\left(1-4\varepsilon_j\left(1-\varepsilon_j\right)\sin^2\frac{\pi}{2^j}\right)}\left|B_q\left(\boldsymbol{K}\right)\right|. \tag{12}$$

If $\varepsilon_j = \varepsilon$ for all j, Equation (12) is equal to Equation (10). As Equation (12) shows, the error rates of the higher bits have a greater effect on the modular bias. Table 3 shows the values of $\sin^2\left(\pi/2^j\right)$ for each j. This shows that the contribution for $j = 1$ is much greater than for the other cases. Figure 2 shows the exact values of the modular bias when the error rates between the first bit and second and after bits are different. The figure indicates that the absolute value of the bias is greater when the error rate of the first bit is smaller than that of the second and subsequent bits as compared to when the error rate of the first bit is greater than that of the second and subsequent bits. In other words, if the error rate is different for each bit of the nonce, the modular bias is highly dependent on the first MSB. This can be seen from the approximated equation, since for small x, $\sin^2 x$ is approximated as x^2. That is, $\sin^2\left(\pi/2^j\right) \approx \pi^2/2^{2j}$ if $j \geq 5$. A visual explain of the bias function for multi-bit leakage associated with this value is shown in Appendix B.

Table 3. Values of $\sin^2\dfrac{\pi}{2^j}$ at each j.

j	1	2	3	4	5	6	$\cdots$	10
$\sin^2(\pi/2^j)$	1	0.5	0.146	0.038	0.009	0.002	$\cdots$	0.000009

The term $\sqrt{1-4\varepsilon_j\left(1-\varepsilon_j\right)\sin^2\left(\pi/2^j\right)}$ in Equation (12) can be expressed as

$$\sqrt{1-4\varepsilon_j\left(1-\varepsilon_j\right)\left(\pi^2/2^{2j}\right)} \tag{13}$$

from the above approximation. We can see that this term rapidly converges to 1 as $l \to \infty$, regardless of the value of ε_j.

Intuitively, the proof of Lemma 2 shows that $\Pr\left[\boldsymbol{K} = k_i\right]$ is designed so that one term of each $\Pr\left[\boldsymbol{K} = k_i\right]$ remains depending on the value of b. Related figures are shown in Figures 3 and 4. Figure 3 shows the modular bias for the 2 bits case, and Figure 4 shows the modular bias for the 3 bits case. The sum of the absolute values of the four or eight vectors is 1, respectively. The absolute value of the sum of these vectors is the absolute value of the modular bias. An interesting fact is that a vector with 2 or 3 bits wrong has a smaller effect on the absolute value of the modular bias than a vector with only 1 bits wrong. In addition, Figures 3 and 4 show the sum of the vectors is 0 if $\varepsilon_1 = 0.5$, which is mentioned at the end of Remark 1.

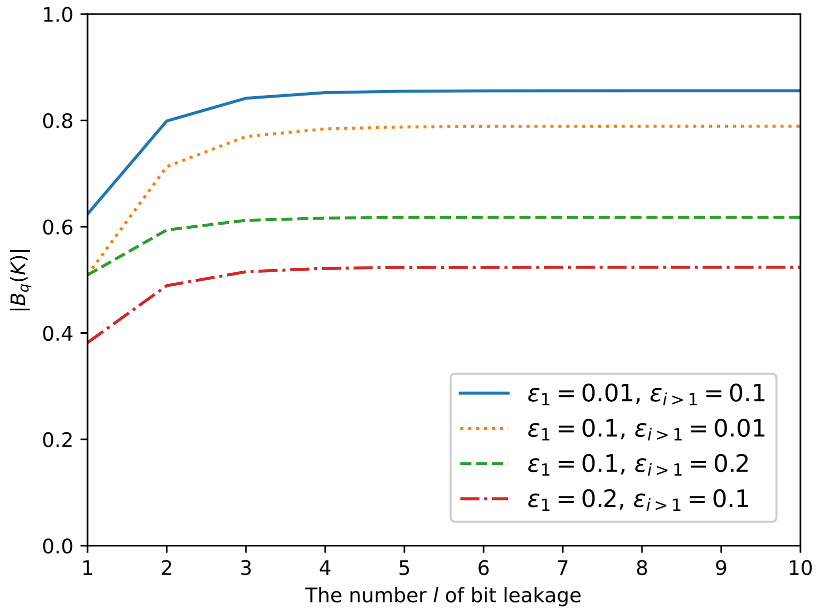


Fig. 2. Modular bias for different error rate for each bit of nonce

3.4 The number of signatures required for key-recovery and error rates

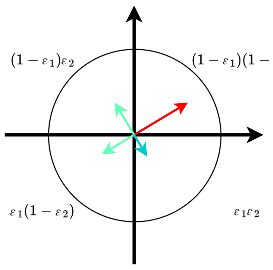
The constraint of sparse linear combinations is given by $|B_q(K)|^{\Omega_j} \gg \alpha/\sqrt{M'}$. Suppose that $|B_q(K)|$ α are given for this inequality. We can satisfy the inequality by choosing smaller Ω_j, which is the number of linear combinations, or larger M', which is the number of samples after linear combination. The number of samples after linear combination required for r rounds is given by the following equation based on the error rate and bias.

$$M' \gg \alpha / \left(\sqrt{\prod_{j=1}^{l} \left(1 - 4\varepsilon_j (1 - \varepsilon_j) \sin^2 \frac{\pi}{2^j}\right)} |B_q(\boldsymbol{K}_b)| \right)^{2\times 4^r} \tag{14}$$

From the fact that $v_i \le a_i$ in the input of Algorithm 5 and Equation (7), we obtain $m_{i+1} \le 4m_i - n_i - 8$. We then have the following.

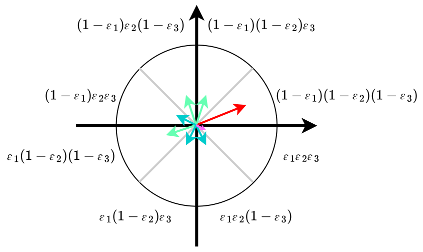
$$m' = m_r \le 4^r m_0 - \sum_{i=0}^{r-1} 4^{r-i-1} n_i - \frac{8}{3}(4^r - 1) \tag{15}$$

In the 4-list sum algorithm, it holds that $t_i = a_i + v_i$, $m_i = a_i + 2$, and $v_i \le a_i$ in the input of Algorithm 5. Accordingly, the inequations $t_i \le 2m_i - 4$ are obtained.



$(1-\varepsilon_1)\varepsilon_2$
$(1-\varepsilon_1)(1-\varepsilon_2)$
$\varepsilon_1(1-\varepsilon_2)$
$\varepsilon_1\varepsilon_2$

Fig. 3. Modular bias illustrated on the unit circle with 2 bits leakage.



$(1-\varepsilon_1)\varepsilon_2(1-\varepsilon_3)$
$(1-\varepsilon_1)(1-\varepsilon_2)\varepsilon_3$
$(1-\varepsilon_1)\varepsilon_2\varepsilon_3$
$(1-\varepsilon_1)(1-\varepsilon_2)(1-\varepsilon_3)$
$\varepsilon_1(1-\varepsilon_2)(1-\varepsilon_3)$
$\varepsilon_1\varepsilon_2\varepsilon_3$
$\varepsilon_1(1-\varepsilon_2)\varepsilon_3$
$\varepsilon_1\varepsilon_2(1-\varepsilon_3)$

Fig. 4. Modular bias illustrated on the unit circle with 3 bits leakage.

We then have the sum of time complexity as follows.

$$\sum_{i=0}^{r-1} t_i \leq 2\sum_{i=0}^{r-1} m_i - 4r \tag{16}$$

From Equations (14) and (15), the estimated number of signatures required for the attack is bounded by

$$M \geq \frac{1}{\prod_{j=1}^{l}\left(1-4\varepsilon_j\left(1-\varepsilon_j\right)\sin^2\left(\pi/2^j\right)\right)} \times \frac{1}{\{(2^l/\pi)\cdot\sin\left(\pi/2^l\right)\}^2} \times 2^{\mathcal{A}}, \tag{17}$$

where

$$\mathcal{A} = \sum_{i=0}^{r-1} 4^{-i-1} n_i + \frac{8}{3}\left(1-4^{-r}\right). \tag{18}$$

From Equation (17), we can see that a higher error rate increases the number of signatures required and that an increase in the length of the known nonce reduces the number of signatures required. For example, from Table 2, a comparison of $\varepsilon = 0.01$ and $\varepsilon = 0.1$ when $l = 2$ reveals that $0.8725^2/0.6522^2 \approx 1.79$ times increase. In addition, comparing $l = 3$ and $l = 1$ for $\varepsilon = 0.01$, we see that $0.9427^2/0.6238^2 \approx 2.284$ times increase in the number of signatures. Furthermore, we find that the error rate and size of the bias do not affect the number of signatures required, whereas the number of rounds is varied. Note that the values for $l = 1$ are completely consistent with the evaluation of Aranha et al.

To understand Equation (17), we can break it down into three separate parts and analyze each one individually.

Third term, represented by $2^{\mathcal{A}}$, remains constant regardless of any changes to l or ε. By utilizing Equations (17) and (18), we can determine that the number of required signatures for an attack is solely dependent on r and n_is, provided that l and ε remain unchanged. These values are utilized in the calculation of $\mathcal{A}$. Moreover, if r is fixed, it depends only on n_is. Therefore, the number of signatures required for the attack depends on the value of n_i. Here, n_i is represented by constraints such as $m_{i+1} = 3a_i + v_i - n_i$ and $\lambda - \ell_{\text{FFT}} - f \leq \sum_{r=0}^{r-1} n_i$ as given in Table 1. M is minimized if the equality holds. It is also multiplied by the square of the inverse of the modular bias. Considering each value in Table 2, we find that the number of signatures required increases significantly for high error rates.

The initial component of Equation (17) is referred to as the *penalty* term, which is always greater than 1, except when all ε_j values are zero. Moreover, as the value of ε_j increases, this term also increases, ultimately leading to an increase in M. This aligns with our natural intuitions.

The value of the second term is determined solely by the parameter l. As l increases, this term gradually decreases and approaches 1, but it always remains greater than 1. As l increases, the required signatures decrease, which intuitively makes sense. The penalty term prevents the second term from reducing M, and its significance increases with an increase in ε_j. However, it does not completely eliminate the possibility of the second term reducing M.

Combing Equation (17) with Equation (13), we can estiamte the contribution j-th MSB leakage. We can see that as l becomes larger, M will decrease, but its rate of decrease will be negligibly small.

4 Experimental results

4.1 Extension to multiple bit leakage with errors

Aranha et al. [3] have posted a script on GitHub [10] for solving linear programming problems based on Table 1. In this script, ε is freely changeable. On the other hand, the number l of nonce bits to leak is fixed to $l = 1$. In a Fourier analysis-based attack, the leakage bit length and error rate affect only $|B_q(\boldsymbol{K})|$ in the constraints of Table 1. Therefore, we can easily obtain the script for multiple bits leakage by replacing the $|B_q(\boldsymbol{K})|$ evaluation equation for the [10] script with Equation (10).

We first naively optimize the number of signatures for multiple bit leakage with errors using a script with only $|B_q(\boldsymbol{K})|$ modifications. Figure 5 shows the optimal number of signatures for each ε and l. Here, $\lambda = 162$, $m_{\max} = 40$, $\ell_{\text{FFT}} = 40$, $t_{\max} = 80$, $r = 2$. In addition, α depends only on the value of ε because L_{FFT} is fixed.

4.2 Attack experiment

For 131-bit ECDSA, we recover the secret key when nonces have 1 bit leakage without error and when 2 bits leakage, each with an error rate of 0.1. The computer used in the experiments has Intel Xeon Silver 4214R CPU $\times 2$ and 256GB of DDR4 RAM. The parameters for the $l = 1, \varepsilon = 0$ and $l = 2, \varepsilon = 0.1$ cases are shown in Table 4. Table 5 shows the obtained $M' = 2^{m'} = 2^{m_2}$, mean value of bias and peak bias as a result of range reduction. In both cases,

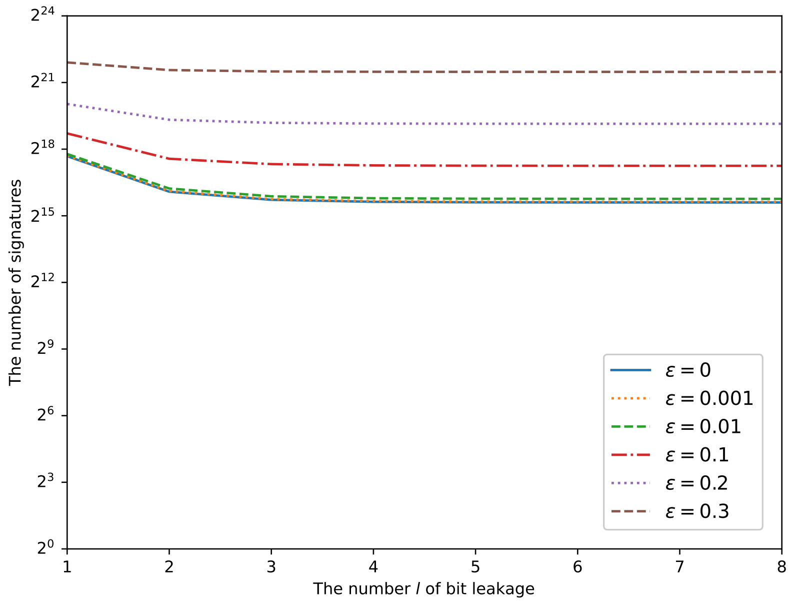


Fig. 5. The number of signatures required for the extended [3] script.

the top 29 bits were successfully recovered. The experimental results show that $l = 2, \varepsilon = 0.1$ successfully recovers the secret key with a smaller bias value. Note that the error rate of 2 bits is 0.19, since the error rate of each bit is 0.1.

Table 4. Paramenters of attack experiment.

	a_0	a_1	v_0	v_1	n_0	n_1
$(l, \varepsilon) = (1, 0)$	22	24	18	18	48	55
$(l, \varepsilon) = (2, 0.1)$	22	24	18	16	48	55

From the experimental results, when $l = 2, \varepsilon = 0.1$, the secret key was successfully recovered with about 1/16 of the number of samples after linear combination than when $l = 1, \varepsilon = 0$. Furthermore, the time required for range reduction is about 0.26 times smaller. Although the value of M' is changed by the parameter v in this case, the number of signatures required can be changed by changing other parameters, and it can be inferred that the 2 bits leakage requires a smaller number of signatures.

Next, the parameters in Table 4 were changed to $a_0 = 20, a_1 = 23$, and $v_1 = 18$ to confirm the experiment in the case with errors. As a result, $M' = 2^{23.0}$,

Table 5. Experiment result.

	M'	Average noise	Peak bias	Range reduction time (sec)
$(l, \varepsilon) = (1, 0)$	$2^{28.1}$	1.5×10^{-5}	1.5×10^{-4}	5957
$(l, \varepsilon) = (2, 0.1)$	$2^{22.1}$	2.0×10^{-6}	1.7×10^{-5}	1555

the time was 1984 seconds, and the secret key is successfully recovered. This shows that $l = 2, \varepsilon = 0.1$ can be recovered with fewer signatures and in less time than without errors.

5 Conclusion

We first evaluated the number of signatures by finding the formula of the modular bias for multiple bit leakage in the nonce. The modular bias as indicated by De Mulder et al. [8] and Aranha et al. [3] was extended to the case in which the MSBs of the nonce were leaked with multiple errors. We then proved Theorem 1. As the modular bias can now be calculated for any l, ε, we can now estimate the required number of signatures using a linear combination algorithm. In addition, the absolute value of the modular bias was given by Corollary 1. This corollary indicates that the error rate of the first MSB of the nonce has a greater effect on the modular bias than the error rates of the other bits. We then provided an estimate of the number of signatures required for various error rates.

We evaluated the number of signatures and computation time by obtaining the parameters of the 4-list sum algorithm. Then, we performed an attack on 131-bit ECDSA with $l = 2, \varepsilon = 0.1$, and succeeded in recovering the secret key with fewer signatures with $l = 1, \varepsilon = 0$.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR2113 and JSPS KAKENHI Grant Number 21H03440.

References

1. Albrecht, M.R., Heninger, N.: On bounded distance decoding with predicate: Breaking the "lattice barrier" for the hidden number problem. In: EUROCRYPT 2021. LNCS, vol. 12696, pp. 528–558. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_19
2. Aranha, D.F., Fouque, P., Gérard, B., Kammerer, J., Tibouchi, M., Zapalowicz, J.: GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In: ASIACRYPT 2014. LNCS, vol. 8873, pp. 262–281. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_14
3. Aranha, D.F., Novaes, F.R., Takahashi, A., Tibouchi, M., Yarom, Y.: Ladderleak: Breaking ECDSA with less than one bit of nonce leakage. In: ACM CCS '20:. pp. 225–242 (2020). https://doi.org/10.1145/3372297.3417268
4. Bleichenbacher, D.: On the generation of one-time keys in dl signature schemes. In: Presentation at IEEE P1363 working group meeting. p. 81 (2000)

5. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In: CRYPTO '96. LNCS, vol. 1109, pp. 129–142. Springer (1996). https://doi.org/10.1007/3-540-68697-5_11
6. Dinur, I.: An algorithmic framework for the generalized birthday problem. Des. Codes Cryptogr. **87**(8), 1897–1926 (2019). https://doi.org/10.1007/s10623-018-00594-6
7. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer (2013). https://doi.org/10.1007/978-3-642-36095-4_19
8. Mulder, E.D., Hutter, M., Marson, M.E., Pearson, P.: Using bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In: CHES 2013. LNCS, vol. 8086, pp. 435–452 (2013). https://doi.org/10.1007/978-3-642-40349-1_25
9. Sun, C., Espitau, T., Tibouchi, M., Abe, M.: Guessing bits: Improved lattice attacks on (EC)DSA with nonce leakage. IACR TCHES **2022**(1), 391–413 (2022). https://doi.org/10.46586/tches.v2022.i1.391-413
10. Takahashi, A.: ladderleak attack ecdsa (2020), `https://github.com/akiratk0355/ladderleak-attack-ecdsa/blob/master/tradeoffs-submit.ipynb`
11. Takahashi, A., Tibouchi, M., Abe, M.: New bleichenbacher records: Fault attacks on qdsa signatures. IACR TCHES **2018**(3), 331–371 (2018). https://doi.org/10.13154/tches.v2018.i3.331-371
12. Wagner, D.A.: A generalized birthday problem. In: CRYPTO 2002. LNCS, vol. 2442, pp. 288–303 (2002). https://doi.org/10.1007/3-540-45708-9_19

A Proof of Lemma 2

The proof for $b = 0$ is as follows. Note that for simplicity, we denote $\exp(2\pi \mathrm{i} k_i/q)$ by $\mathcal{E}_q(k_i)$.

$$\begin{aligned}
B_q(\boldsymbol{K}) = \boldsymbol{E}[\exp(2\pi\mathrm{i}\boldsymbol{K}/q)] &= \sum_{k_i\in\mathbb{Z}_q} \mathcal{E}_q(k_i)\cdot \Pr[\boldsymbol{K}=k_i] \\
&= \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4}\sum_{k_i\in[0,q/4)}\mathcal{E}_q(k_i) + \frac{(1-\varepsilon_1)\varepsilon_2}{q/4}\sum_{k_i\in[q/4,q/2)}\mathcal{E}_q(k_i) \\
&+ \frac{\varepsilon_1(1-\varepsilon_2)}{q/4}\sum_{k_i\in[q/2,3q/4)}\mathcal{E}_q(k_i) + \frac{\varepsilon_1\varepsilon_2}{q/4}\sum_{k_i\in[3q/4,q)}\mathcal{E}_q(k_i) \\
&= \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4}\sum_{k_i\in[0,q/4)}\mathcal{E}_q(k_i) + \frac{(1-\varepsilon_1)\varepsilon_2}{q/4}\sum_{k_i^{(1)}\in[0,q/4)}\mathcal{E}_q\left(k_i^{(1)}+q/4\right) \\
&+ \frac{\varepsilon_1(1-\varepsilon_2)}{q/4}\sum_{k_i^{(2)}\in[0,q/4)}\mathcal{E}_q\left(k_i^{(2)}+q/2\right) + \frac{\varepsilon_1\varepsilon_2}{q/4}\sum_{k_i^{(3)}\in[0,q/4)}\mathcal{E}_q\left(k_i^{(3)}+3q/4\right) \\
&= \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{q/4}\sum_{k_i\in[0,q/4)}\mathcal{E}_q(k_i) + \mathrm{i}\frac{(1-\varepsilon_1)\varepsilon_2}{q/4}\sum_{k_i^{(1)}\in[0,q/4)}\mathcal{E}_q\left(k_i^{(1)}\right)
\end{aligned}$$

$$
\begin{aligned}
&- \frac{\varepsilon_1 (1-\varepsilon_2)}{q/4} \sum_{k_i^{(2)} \in [0, q/4)} \mathcal{E}_q\left(k_i^{(2)}\right) - \mathrm{i}\frac{\varepsilon_1 \varepsilon_2}{q/4} \sum_{k_i^{(3)} \in [0, q/4)} \mathcal{E}_q\left(k_i^{(3)}\right) \\
&= \frac{(1-2\varepsilon_1)(1-\varepsilon_2)}{q/4} \sum_{k_i \in [0, q/4)} \mathcal{E}_q(k_i) + \mathrm{i}\frac{(1-2\varepsilon_1)\varepsilon_2}{q/4} \sum_{k_i \in [0, q/4)} \mathcal{E}_q(k_i) \\
&= \{(1-2\varepsilon_1)(1-\varepsilon_2) + \mathrm{i}(1-2\varepsilon_1)\varepsilon_2\} B_q(K_b)
\end{aligned}
$$

B Visual explanation of the bias function for multi-bit leakage

In this appendix, Equation (9) is represented graphically. In Equation (9), at each j, $(1-\varepsilon) + \exp\left(2\pi\mathrm{i}/2^j\right)$ can be understood as a point in the complex plane with 1 and $\exp\left(2\pi\mathrm{i}/2^j\right)$ endowed by $1-\varepsilon : \varepsilon$. The endpoints in the complex plane at $j = 1, 2$ and $j = 3, 4$ in Figures 6 and 7, respectively, are indicated by red dots. ε and $1-\varepsilon$ in the figures represent ratios. When $j = 1$, the two points 1 and -1 are endowed by $1-\varepsilon : \varepsilon$, and the red point is in the complex plane at coordinates $1-2\varepsilon$. When $j = 2$, we endow 1 and i, and when $j = 3$, we endow 1 and $\exp(\pi\mathrm{i}/4)$ with $1-\varepsilon : \varepsilon$.

As j increases, $\exp\left(2\pi\mathrm{i}/2^j\right)$ approximates 1. Therefore, $\exp\left(2\pi\mathrm{i}/2^j\right)$ and the interior point of 1 also approximates 1. Figures 6 and 7 also show that the interior point approximates 1 in the complex plane. The absolute value also approximates 1.

Table 2 shows that as l increases, the value does not readily increase. As explained in Section 3.3, this is because $\sin^2\left(\pi/2^j\right)$ is closer to 0. This can also be observed in Figures 6 and 7.

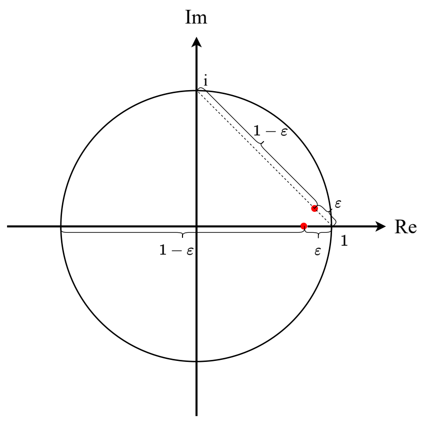


Fig. 6. When the first and second bits leak.

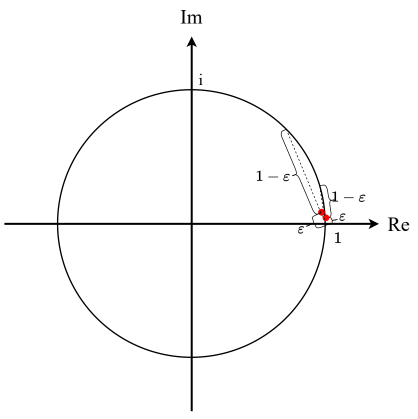


Fig. 7. When the third and fourth bits leak.