Bias from Uniform Nonce: Revised Fourier Analysis-based Attack on ECDSA

Shunsuke Osaki[1] and Noboru Kunihiro[2]

[1]NEC Corporation, Japan
[2]University of Tsukuba, Japan

2024/8/29

Outline

ECDSA (Elliptic Cureve Digital Signiture Algorithm)

- Used for SSH, SSL/TLS, Bitcoin, etc.

ECDSA key recovery

- Solving ECDSA from only public key is reduced to solve the discrete logarithm problem known as ECDLP
- It is believed that exponential time is required to solve.

- By using a part of the secret information called nonce (Number used only ONCE) and a number of ECDSA signatures, the secret key is recovered.

ECDSA signature generation algorithm

Algorithm 1 ECDSA signature generation

Input: prime number q, secret key $\mathrm{sk} \in \mathbb{Z}_q$, message msg $\in \{0,1\}^*$, base point G, and hash function $H : \{0,1\}^* \to \mathbb{Z}_q$

Output: valid signature (r, s)

1: $k =_{\$} \mathbb{Z}_q$

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)

- If the fully nonce is leaked or reused, the secret key is recovered.
- If a part of the nonce is leaked, it is known that the secret key can be recovered by solving HNP.
- Consider a situation where the top l bits of nonces k are leaked with an error (error rate ε) due to a side-channel attack

Previous Studies and Research Goals

- Several security evaluations have been performed assuming partial leakage of the nonce
- By reducing this leakage to the Hidden Number Problem (HNP), the secret key can be recovered using lattice-based attacks or Bleichenbacher's Fourier analysis-based attacks
- Fourier analysis-based attacks can recover the secret key even when the nonce error rate is high or the length of the leaked bits is short
- In the previous studies [Ble00] [MHMP13] [AFGKTZ14] [TTA18] [ANTTY20] [OK23], if the leaked MSBs are uniform, they collect nonces which top bits are same to get biased nonces.

Research Goals

Reduce the number of signatures to recover the secret key by using all signatures.
To reduce, we generate biased samples from uniform samples.

Summary of our contributions

Contribution 1

- Correct the estimate the number of samples which are outputs of 4-list sum algorithm.

Contribution 2

- Reduce the number of signatures to recover the secret key
- Successfully recovered secret keys with fewer signatures and the same runtime and computational resources as previous studies
  - 50% reduction with 1 bit leakage
  - 75% reduction with more than 2 bits leakage

Translation to Hidden Number Problem (HNP)

Consider the situation where the most significant bits of the nonces are leaked

- Function $\mathrm{MSB}_n(x)$ returns the top n bits of x for a $x \in \mathbb{N}$
- Let $k_i = z_i + h_i \cdot \mathrm{sk} \bmod q$, for each $i = 1, \ldots, M$.
- HNP is the problem of finding sk for $i = 1, \ldots, M$, given $\{h_i, z_i, \mathrm{MSB}_n(k_i)\}$

Transforming the equation for signature generation yields

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

Let $z := H(\text{msg})/s \bmod q, \;\; h := r/s \bmod q$, then

$$k = z + h \cdot \mathrm{sk} \bmod q$$

If MSBs of k is leaked, we get a sample of HNP

How to solve HNP

Two methods for solving are known:

Lattice-based attack

+ Dozens of signatures
+ Laptop
+ Less than an hour
- The nonces do not contain high errors

Fourier analysis-based attack

- Hundred of millions signatures
- Workstation
- A few days or a week
+ The nonces can contain high errors

Lattice for errors [GWHH24]

- Recover secret key with hundred of millions signatures
- They show that recovery is possible with an error rate up to 0.1.
- But the number of signatures required is higher than with the Fourier analysis-based attack

[GWHH24]Gao et al., "Attacking ECDSA with Nonce Leakage by Lattice Sieving: Bridging the Gap with Fourier Analysis-based Attacks", ePrint 2024

Bias function

Definition 1

Sample bias for the set $K = \{k_j \in \mathbb{Z}_q\}_{j=1}^{M}$ is given by

$$B_q(K) := \frac{1}{M}\sum_{j=1}^{M}\exp\left(\frac{2\pi k_j}{q}\mathrm{i}\right)$$

- We can compute the function by Fast Fourier Transformation.
- If each k_i is random, the aboslute value is $1/\sqrt{M}$.

From [TTA18] the absolute value of the sample bias is:

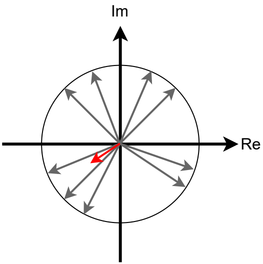
$$\lim_{q\to\infty}|B_q(K)| \to \frac{2^l}{\pi}\cdot\sin\left(\frac{\pi}{2^l}\right).$$

when the top l bits of all k_i are fixed to a constant.

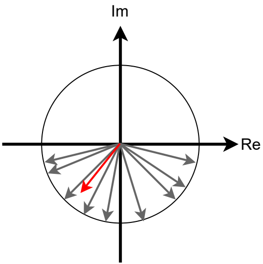
- If $l = 1$, the value is $0.637\,(= 2/\pi)$; if $l = 2$, the value is $0.900\,\left(= 2\sqrt{2}\pi\right)$.

Image of bias function

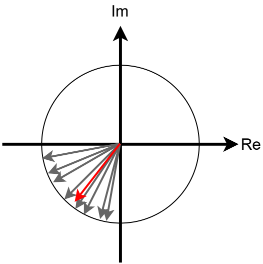
- Average of vectors on the unit circumference of the complex plane
- The more biased the nonces, the larger the absolute value of the bias
- Use the fact that the computed bias is larger for the correct secret key as an attack



Bias in the random case



Bias in the case of $\mathrm{MSB}_1(k) = 1$



Bias in the case of $\mathrm{MSB}_2(k) = 10$

If top l bits of nonces leak with errors

From [OK23] when the top l bits of the nonces leak with errors, the absolute value of the bias function can be expressed as:

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

- If the error rate of each bit of nonces are same, we can use $\varepsilon_j = \varepsilon$.
- If $l = 1$, the result is equal to that of Aranha et al.
- Let α and β be error rates where $\alpha < \beta$. $|B_q(K)|$ for $\varepsilon_1 = \alpha, \varepsilon_2 = \beta$ is larger than $|B_q(K)|$ for $\varepsilon_1 = \beta, \varepsilon_2 = \alpha$

Naive key search method

Perform an exhaustive secret key search and obtain the w with the largest bias as the correct secret key

Algorithm 2 Naive method

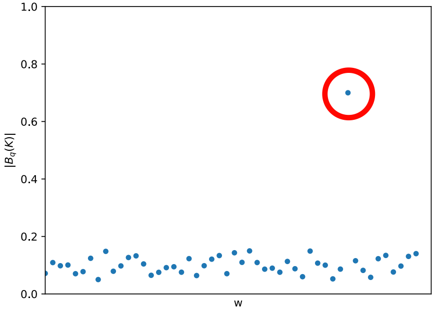
Input: $(h_i, z_i)_{i=1}^{M}$: Nonce biased HNP samples on $\mathbb{Z}_q$
Output: Correct secret key sk

1: **for** $w = 1$ to $q - 1$ **do**
2: Compute the set $K_w = \{z_i + h_i w \bmod q\}_{i=1}^{M}$
3: Compute $|B_q(K_w)|$
4: **end for**
5: **return** w that maximizes $|B_q(K_w)|$

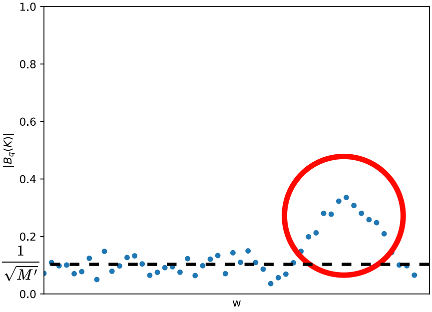
The naive method is inefficient because it performs an exhaustive secret key search

After taking linear combinations of the samples, efficiency is improved by computing the bias

Peak bias using linear combinations



Peak bias before linear combinations



Peak bias after linear combinations

- Before linear combinations, the bias is large only for the secret key
- After linear combinations, the bias is large near the secret key. However, the peak goes down.

Reduce the search range using linear combinations

De Mulder et al. and Aranha et al. proposed a method to avoid the full search for the secret key using linear combinations of samples

Attack strategy (linear combinations)

- M':Number of samples after linear combination, $L_{\mathrm{FFT}}\ (< q)$: FFT table size
- Take linear combinations of the input samples $\{(h_i, z_i)\}_{i=1}^{M}$ and new samples $\left\{\left(h'_j, z'_j\right) = (\sum_i \omega_{i,j} h_i, \sum_i \omega_{i,j} z_i)\right\}_{j=1}^{M'}$ with $h'_j < L_{\mathrm{FFT}}$ are generated, where $\omega_{i,j} \in \{-1, 0, 1\}$, $\Omega_j := \sum_i |\omega_{i,j}|$
- The peak width extends from 1 to about q/L_{FFT}. Candidate secret key to be examined decreases from q to L_{FFT}.

Constraints on linear combinations

Sparse linear combinations

- Distinguishable if the value of the bias corresponding to the correct secret key is much larger than the average of the noise $1/\sqrt{M'}$
- By taking many linear combinations, it is easy to make small h'_j
- However, by taking many linear combinations, the aboslute value of the bias corresponding to the correct secret key decreases exponentially, as in $|B_q(K)|^{\Omega_j}$

- To find M' that is $|B_q(K)|^{\Omega_j} \gg 1/\sqrt{M'}$, it is sufficient to estimate $|B_q(K)|$ exactly
- It is important to compute the bias function rigorously to find parameters such as the number of signatures needed to perform Fourier analysis-based attack

How to take linear combinations

- [ANTTY20] takes linear combinations by using 4-list sum algorithm.
- 4-list sum algorithm can be used to increase the number of samples while decreasing the value by taking a linear combination
- They make linear programming problem to estimate signatures.

Constraints on linear programming problem

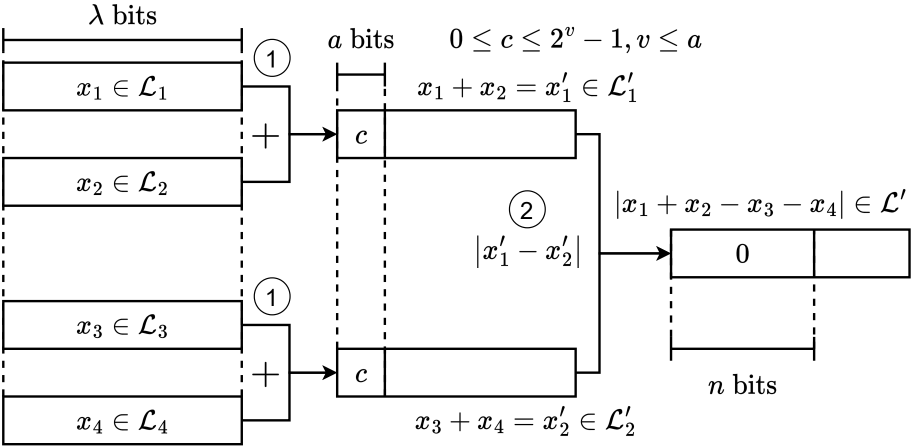
Table: Linear programming problem based on the Iterative HGJ 4-list sum algorithm. Each column is a constraint to optimize [ANNTY20]

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

Outline

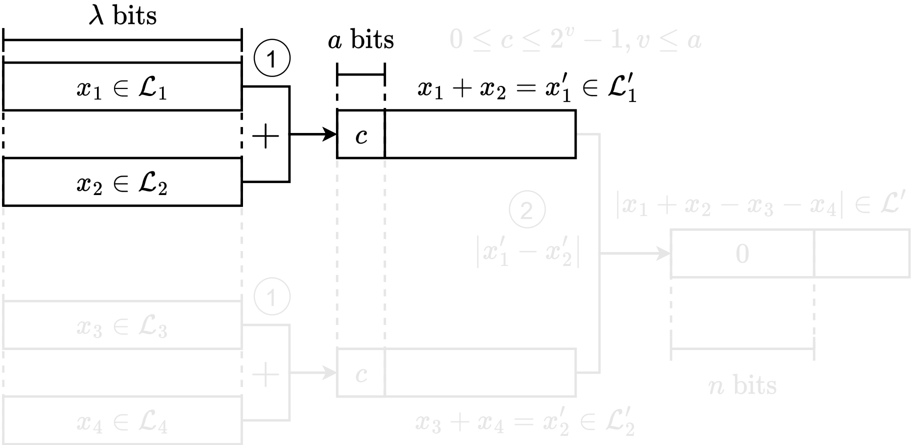
4-list sum algorithm [ANTTY20]

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \leq a, n$
- Output: $|\mathcal{L}'| = 2^{a+a-(n-a)+v} = 2^{3a+v-n}$
- $|\mathcal{L}'_1| = |\mathcal{L}'_2| = 2^{a+a-a} = 2^a$



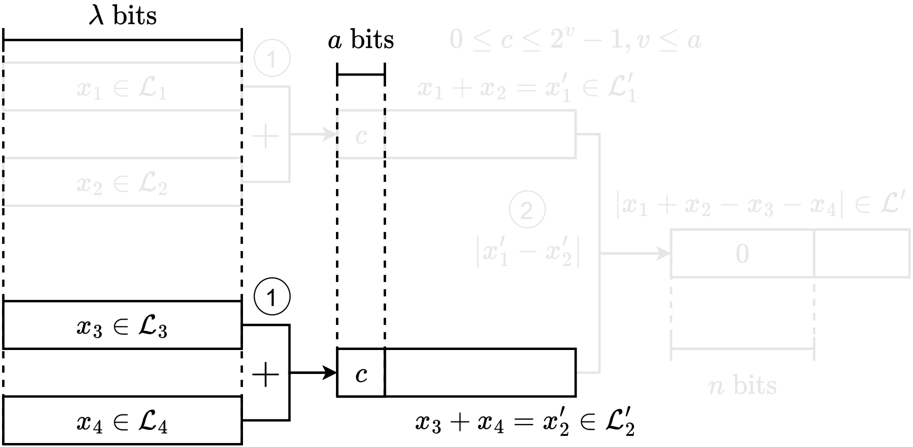
4-list sum algorithm [ANTTY20]

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \le a, n$
- Output: $|\mathcal{L}'| = 2^{a+a-(n-a)+v} = 2^{3a+v-n}$
- $|\mathcal{L}'_1| = |\mathcal{L}'_2| = 2^{a+a-a} = 2^a$



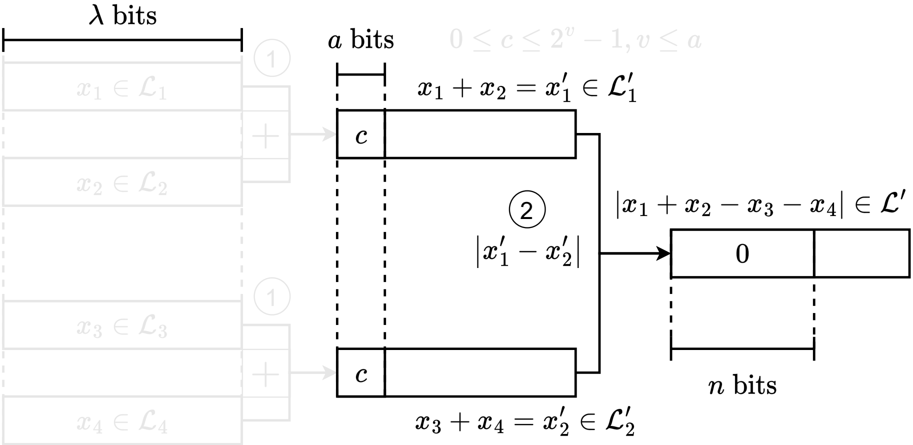
4-list sum algorithm [ANTTY20]

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \leq a, n$
- Output: $|\mathcal{L}'| = 2^{a+a-(n-a)+v} = 2^{3a+v-n}$
- $|\mathcal{L}'_1| = |\mathcal{L}'_2| = 2^{a+a-a} = 2^a$



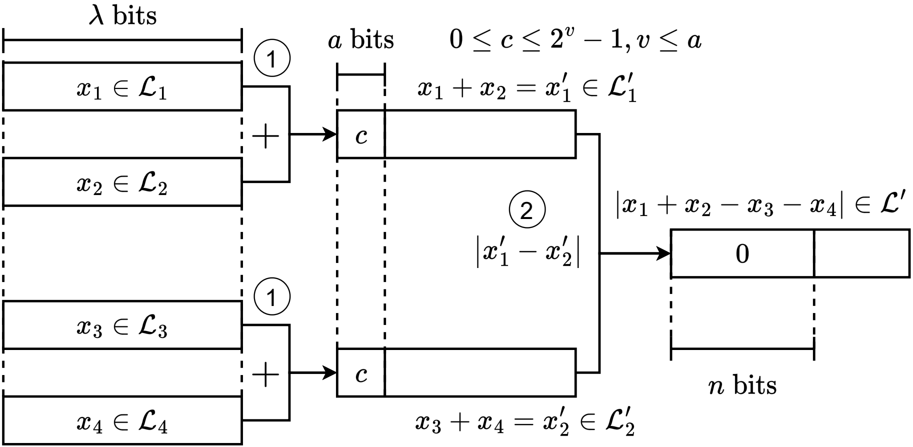
4-list sum algorithm [ANTTY20]

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \leq a, n$
- Output:
  $|\mathcal{L}'| = 2^{a+a-(n-a)+v} = 2^{3a+v-n}$
- $|\mathcal{L}'_1| = |\mathcal{L}'_2| = 2^{a+a-a} = 2^a$



4-list sum algorithm [ANTTY20]

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \leq a, n$
- Output:
  $|\mathcal{L}'| = 2^{a+a-(n-a)+v} = 2^{3a+v-n}$
- $|\mathcal{L}'_1| = |\mathcal{L}'_2| = 2^{a+a-a} = 2^a$



Issue 1: 4-list sum algorithm of [ANTTY20]

Issue 1: Carry is not considered

Let $\lambda = 5, n = 4, a = 2$ and then let
$x_1 = 17\,(1\,0001)\,, x_2 = 18\,(1\,0010)\,, x_3 = 15\,(1111)\,, x_4 = 17\,(1\,0001)$

- $x'_1 = 35\,(10\,0011)\,, x'_2 = 32\,(10\,0000)$ then $\mathrm{MSB}_2\,(x'_1) = \mathrm{MSB}_2\,(x'_2) = 2\,(10)$
- $\mathrm{MSB}_4\,(|x'_1 - x'_2|) = 0$ then $|x'_1 - x'_2| = 3\,(11)$

Since $\lambda - n = 1$, the output result is expected to be less than 1 bit, but it is 2 bits.

The carry that occurs with a probability of $1/2$ is not considered.

- $|\mathcal{L}'| = 2^{3a+n}$ should be modified to $|\mathcal{L}'| = 2^{3a+n-2}$
- $M' = 2^{3a+v+n-2}$. Previous study estimated more than 4 times

Issue 2: 4-list sum algorithm of [ANTTY20]

Issue 2: The assumption about the distribution is not appropriate.

Estimation of [ANTTY20] is uniform distribution, but the actual biased.

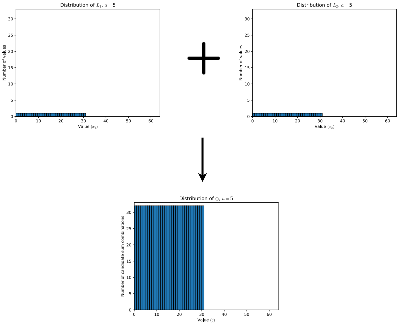


Figure: Distribution assumed in [ANTTY20]

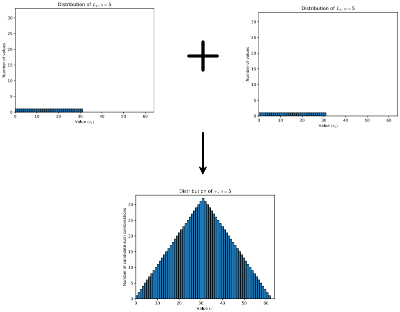
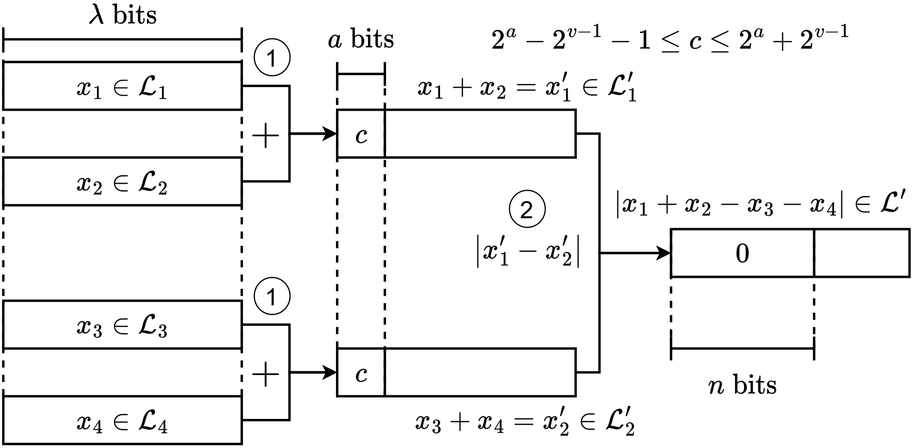


Figure: Real distribution

Our 4-list sum algorithm

- Input: $|\mathcal{L}_1| = \cdots = |\mathcal{L}_4| = 2^a, v \leq a+1, n$
- Output: $|\mathcal{L}'| = \left(2^{2a+v} - 2^{a+2v-1} + \frac{2^{3v-2}}{3} - 2^{2v-2} + \frac{7\cdot 2^v}{6}\right) 2^{-(n-a)}$



Attack experiment

- 60-bit ECDSA
- To check the distribution, it is not necessary to recover the key
- It is sufficient to confirm that the number of samples output does not depend on a

Table: Parameters and results of the experiment

Parameter	a_0	v_0	n_0	a_1	v_1	n_1	Original M'	Our M'
$l = 1, \varepsilon = 0$	8	5	14	14	2	16	0	$2^{29.43}$
$l = 2, \varepsilon = 0.1$	8	5	15	14	2	15	0	$2^{27.34}$

- Original algorithm cannot recover the secret key
- Our algorithm recovers the secret key

Outline

1. Preliminaries and previous studies
2. Modifying of 4-list sum algorithm
3. Attack for uniform nonce

Proposed attack using bias due to linear combination

Previous studies issue

- In previous studies, attacks were conducted using only signatures corresponding to biased nonces
- When 1 bit was leaked, twice the number of signatures were needed for the attack; when 2 bits were leaked, 4 times were needed; and when l bits were leaked, 2^l times were needed.
- Out of the collected signatures, only $1/2^l$ were used, while the remaining $1 - 1/2^l$ were not used

Trick of our new attacks

- By taking linear combinations based on h_i from the set $\{(k_i, h_i, z_i)\}_{i=1}^{M}$, we obtain a new set $\left\{\left(k'_j, h'_j, z'_j\right)\right\}_{j=1}^{M'}$.
- Here, it is sufficient if $\left\{k'_j\right\}_{j=1}^{M'}$ are biased, because the bias calculation is performed after the linear combinations.

Bleichenbacher's attack framework

Algorithm Bleichenbacher's attack framework

Input: $(h_i, z_i)_{i=1}^{M}$: Samples of HNP over $\mathbb{Z}_q$, M': Number of linear combinations to find, L_{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' samples $\left\{\left(h'_j, z'_j\right)\right\}_{j=1}^{M'}$ that satisfies the following two conditions.

(1) Small: $0 \leq h'_j < L_{\mathrm{FFT}}$

(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: $Z := \left(Z_0, \ldots Z_{L_{\mathrm{FFT}}-1}\right) \leftarrow (0, \ldots, 0)$

5: **for** $j = 1$ to M' **do**

6: $\quad Z_{h'_j} \leftarrow Z_{h'_j} + \exp\left(2\pi \mathrm{i} z'_j / q\right)$

7: **end for**

8: Let $w_i = iq/L_{\mathrm{FFT}}$, $\{B_q(K_{w_i})\}_{i=0}^{L_{\mathrm{FFT}}-1} \leftarrow \mathrm{FFT}\,(Z)$

9: Find i that maximizes $|B_q(K_{w_i})|$

10: **return** $\mathrm{MSB}\,(w_i)_{\log L_{\mathrm{FFT}}}$ bits

Methods to reduce the number of collected signatures

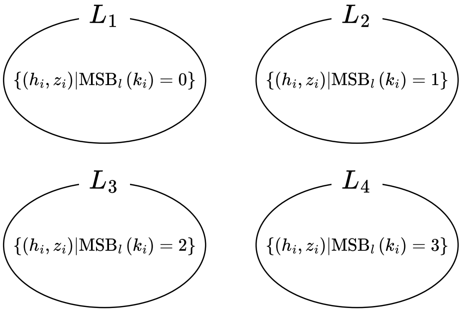
- It is sufficient that MSBs of $\left\{k'_j\right\}_{j=1}^{M'}$ are biased.
- It is sufficient to efficiently perform linear combinations while making bias

Approach

- Employ the 4-list sum algorithm
- Ensure that the top bits of the nonce corresponding to each element in the lists are biased according to the HNP samples.
- Taking linear combinations to the lists, $\left\{k'_j\right\}_{j=1}^{M'}$ be biased

Preprocessing for 2 bits leakage

- HNP samples are assigned to lists by MSBs value



- Apply the 4-list sum algorithm using the obtained set of lists $\{\mathcal{L}_i\}_{i=1}^{2^l}$
- When 1 bit is leaked, split the obtained 2 lists into 4 lists each
- When 3 or more bits are leaked, group the obtained lists into sets of 4 and run the 4-list sum algorithm on each set.

Distribution by linear combinations with 1 leakage

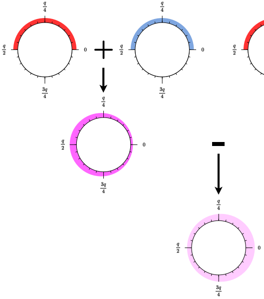


Figure: Biased distribution

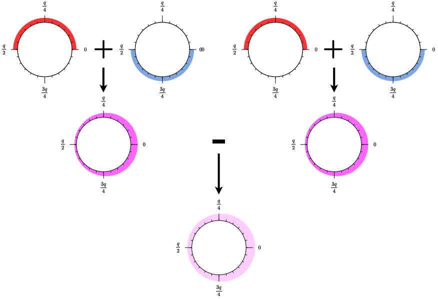


Figure: Uniform distribution

Distribution by linear combinations with 2 leakage

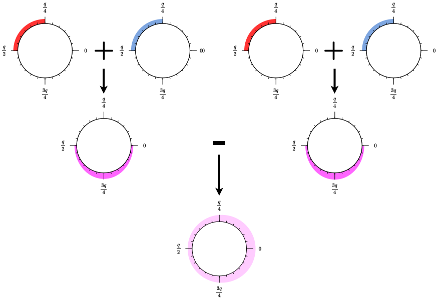


Figure: Biased distribution

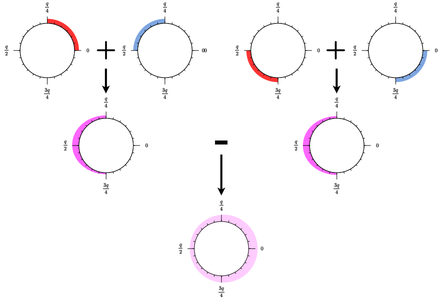


Figure: Uniform distribution

Distribution by linear combinations with 3 leakage; 8 lists

Perform 4-list sum algorithm for $\{0, 1, 2, 3\}$ and $\{4, 5, 6, 7\}$, then get same distribution

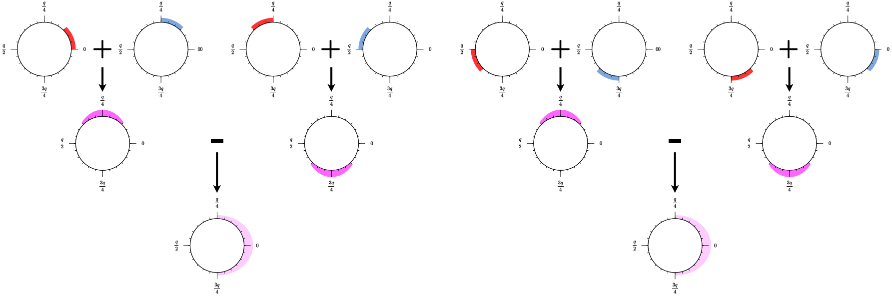


Figure: Uniform distribution

- After 2nd round, input is all output
- Using more time, decreasing the numnber of collected signatures to $1/2^{(l+6)/4}$

Experimental Overview

- We attacked 131-bit ECDSA and confirmed that the secret key can be recovered in a uniform case just as it can in a biased case
- Ubuntu 20.04 LTS, Intel Xeon Silver 4214R $\times 2$, total 24 cores and 48 threads, DDR4 256GB

Experimental Details

In each case, the experiment is as follows

1. The 1 bit contains no error
2. The 2 bits contain no error
3. The error rate for each of the 2 bits is about 0.11. [a]
4. The 3 bits contain no error
    - Using only 4 lists
    - Using all 8 lists

[a] 0.11 is the error rate at which 2 bits can be recovered with an equal number of signatures if 1 bits are leaked with no errors

Experimental Results

Table: Experimental results with bias

l	ε	Number of collected signatures	M'	Sec.	Recovered bits
1	0	2^{24}	$2^{26.90}$	1186	29
2	0	2^{25}	$2^{23.99}$	504	29
	0.11	2^{25}	$2^{26.89}$	1201	29
3	0	2^{20}	$2^{7.93}$	90	29

Table: Experimental results without bias

l	ε	Number of collected signatures	M'	Sec.	Recovered bits	Combinations of lists top l bits
1	0	2^{23}	$2^{26.90}$	1210	29	$\{0,0,1,1\}$
		2^{23}	$2^{26.90}$	1223	29	$\{1,0,1,0\}$
2	0	2^{23}	$2^{23.98}$	530	29	$\{00,01,10,11\}$
	0.11	2^{23}	$2^{26.89}$	1190	29	$\{00,01,10,11\}$
3	0	2^{18}	$2^{7.80}$	87	29	$\{000,010,101,001\}$
		2^{16}	$2^{7.77}$	829	29	$\{000,001,010,011,$ $100,101,110,111\}$

Conclusion

Modifying of 4-list sum algorithm

- Find and solve the issues about carry and distribution

Takeaways: Attack for uniform nonces

- In previous studies, the signatures which nonces are biased only used, so the others are discarded
- Decreasing the number of signatures to recover the secret key
  - 50% decrease with 1 bit, using the same time and computational resources
  - 75% decrease with more than 2 bits, using the same time and computational resources
  - $1/2^{(l+6)/4}$ decreases for more time if more than $l \geq 3$ bits leakage by using 2^l lists