LadderLeak: Breaking ECDSA With Less Than One Bit Of Leakage

Srushti Singh
EPFL
srushti.singh@epfl.ch

Jonathan Poveda Colominas
EPFL
jonathan.povedacolominas@epfl.ch

***Abstract*—This report introduces LadderLeak[1], a sophisticated attack on ECDSA which allows a potential attacker to recover a secret key from less than one bit of nonce leakage. In particular, it suffices that the attacker is able to leak the most significant bit of the nonce with a probability ≤ 1 on several signatures for them to recover the secret key.**

I. Introduction

As one of the most popular signature schemes, ECDSA has gathered attention from researchers who have tried to find weak points in its real-life implementation. One popular attack vector has been to find weaknesses around the nonce k, which turns out to be quite sensitive: the successful leakage of this value would allow attackers to compute the secret key sk. Based on this fact, some research has focused on the idea of recovering the secret key using only part of the nonce. As of the end of 2019, some methods were developed that allowed an attacker to leak a 160-bits secret key from 1 bit of nonce leakage[2] from 2^{33} signatures.

Ladderleak builds upon past research[2], [3] and methods to create a novel attack which relies on solving the Hidden Number Problem (HNP) using a Fourier analysis-based method[4] on DSA and ECDSA. This results in an attack which works even when the most significant bit of the nonce is leaked inconsistently (with error probability between 0 and 1/2).

II. Background

A. Elliptic curve cryptography

An elliptic curve $E(\mathbb{F})$ defined over a finite field $\mathbb{F}$ is a set of points $(x, y) \in \mathbb{F}^2$ which satisfy a curve equation along with a point at infinity O. A group can be defined using an elliptic curve and a group law $+$ (in additive notation) for which the point O is the identity element. On elliptic curves defined over real numbers, the point $P + Q$ can be found by drawing the line that passes through P and Q and taking the symmetry of axis x of the intersection between the line and the curve.

From this group law, given a point $P \in E(\mathbb{F})$ and a scalar $k \in \mathbb{Z}$, we define the scalar multiplication $[k]P := P + \cdots + P$ where P is added to itself $k-1$ times. Cryptographic protocols based on elliptic curves rely on the Elliptic Curve Discrete Logarithm Problem (ECDLP): given some pair $(P, [k]P)$, an adversary would need to recover the scalar k. It turns out that this problem is hard (on classical computers) for certain curves defined as standards.

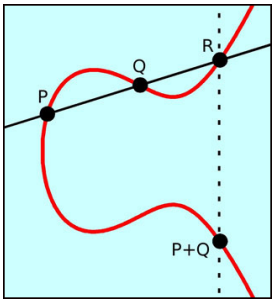

Fig. 1: Group law on an elliptic curve over $\mathbb{R}$

B. ECDSA and its nonce

ECDSA is a signature scheme that relies on elliptic curves and, most specifically, the hardness of the ECDLP to ensure that a message has been sent by the owner of a public key. On an elliptic curve $E(\mathbb{F})$ of prime order q with generator $G \in E(\mathbb{F})$, a sender who has the key pair $(sk, pk) = (sk, [sk]G)$ signs a message m in the following way:

1) Generation of a random nonce $k \in \mathbb{Z}_q^*$
2) Computation of $r = ([k]G)_x \mod q$ where $(P)_x$ is the x coordinate of P
3) Computation of $s = k^{-1}(H(m) + sk \cdot r)$ where k^{-1} is the multiplicative inverse of k in $\mathbb{Z}_q$ and H is a hash function
4) Sender sends $\sigma = (r, s)$ to receiver

The receiver checks the signature's validity by checking if the following equality holds:

$$r \stackrel{?}{=} (\frac{H(m)}{s}G + \frac{r}{s}Q) \mod q$$

One major potential flaw in the implementation of this scheme is the problem of nonce leakage. If implementation flaws lead to an attacker leaking the nonce k, the secret key can be computed easily with the known elements:

$$sk = \frac{k \cdot s - H(m)}{r} \mod q$$

Thus, the nonce is also part of the sensitive values that need to be protected from leakage and has naturally become a potential target for attacks on ECDSA.

III. Theoretical *LadderLeak* attack

LadderLeak is built upon some methods developed in previous works which allowed an attacker to retrieve the secret key using leaks of as few as 1 bit of the nonce. It improves one specific method to create an attack that can work with leaks of 1 bit with error probability between 0 and 1/2.

A. Hidden number problem and ECDSA

Extracting the secret key from nonce leakages in ECDSA signatures is equivalent to solving the Hidden Number Problem (HNP). The article mentions a variant of this problem which takes into account the probability that the leaked bits are not correct called *Hidden Number Problem with Erroneous Input*.

Definition 1: Let q be a prime number and $sk \in \mathbb{Z}_q$ be a secret. Let h_i, k_i be uniformly random elements of $\mathbb{Z}_q$ for each $i = 1, \ldots, M$ and define $z_i = k_i - h_i \cdot sk \mod q$. Suppose a fixed distribution χ_b on $\{0,1\}^b$ for $b > 0$ and define a probabilistic algorithm $\texttt{EMSB}_{\chi_b}(x) = MSB_b(x) \oplus e$ which returns the b most significant bits of x with e as an error string sampled from χ_b. The HNP with error distribution χ_b asks one to find sk given (h_i, z_i) and $\texttt{EMSB}_{\chi_b}(k_i)$ for $i = 1, \ldots, M$.

In the concrete attack, the authors focus on the case where $b = 1$ and χ_b is the Bernoulli distribution for some error parameter $\varepsilon \in [0, \frac{1}{2}]$ ($\Pr(e = 1) = \varepsilon = 1 - \Pr(e = 0)$). In other words, $\texttt{EMSB}_{\chi_b}(x)$ returns the negation of the most significant bit of x with probability ε.

By rearranging terms on the calculation of the s in ECDSA, we quickly see that a set of signatures with leaky nonces and computed using the same secret key result in an instance of the HNP. More precisely, we get:

$$\frac{H(m_i)}{s_i} = k_i - \frac{r_i}{s_i} sk$$

By letting $z_i = H(m_i)/s_i \mod q$ and $h_i = r_i/s_i \mod q$, we obtain the structure seen in Definition 1 for an instance of the HNP if the most significant bit of k_i is leaked with some probability.

B. Bias function

The method used by the authors of *LadderLeak* is based on an attack briefly presented by Bleichenbacher in 2000[4]. This attack relies on a bias function which can be used to quantify the modular bias of the nonce. If $K = \{k_i\}_{i=1}^M$ is the set of collected nonces, the function should be defined in such a way that $B_q(K) \approx 0$ when the nonces are uniform in $\mathbb{Z}_q$ and $B_q(K) \approx 1$ when they are biased.

The function proposed by Bleichenbacher uses the (magnitude of the) inverse discrete Fourier transform:

$$B_q(K) = \frac{1}{M} \sum_{i=1}^{M} e^{(2\pi k_i/q)i}$$

Intuitively, we see that the requirements on the values of the function given the bias of the nonces are somewhat respected. The bias sums complex values that lay on the unit circle of the complex plane. If those values are uniformly distributed, we expect almost every point to have an opposite on the unit circle and their sum would lead to canceling both terms. Thus, $|B_q(K)| \approx 0$. However, if the nonces are biased, they will be more numerous on a certain part of the unit circle with few opposite points to cancel them. Therefore, the sum leads to a vector of magnitude $\approx M$ and thus, $B_q(K) \approx 1$.

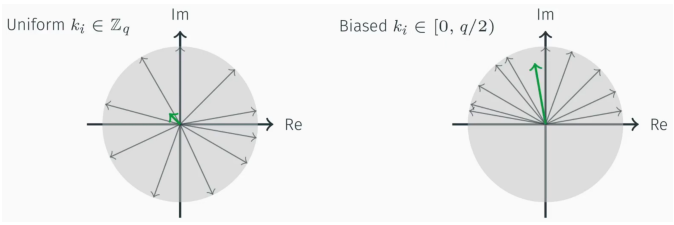

Fig. 2: Visualization of uniform and biased distribution on $B_q(K)$

When the first l MSBs of each nonce in K are fixed to some constant and the rest is uniform modulo q, it has been shown that $|B_q(K)|$ converges to $2^l/\pi \cdot \sin(\pi/2^l)$ for a large q [3]. For instance, for a single fixed MSB, the bias is estimated to $|B_q(K)| \approx 2/\pi \approx 0.637$. Moreover, for uniformly distributed k_i's over $\mathbb{Z}_q$, the mean of the norm of the bias can be approximated by $1/\sqrt{M}$.

This definition of bias function applies in the ideal situation where the samples are uniformly biased, that is, where the first l bits are fixed and the rest is uniform modulo q. The authors are however focused on using this method with some error $\varepsilon \in [0, 1/2]$ in the input. Note that $\varepsilon = 1/2$ means that the samples are not biased (equivalent to random guess of the MSB) and thus, the bias function degenerates to 0. Therefore, the definition of bias must be adapted to the following:

Lemma 1: For $b \in \{0,1\}$, any error $\varepsilon \in [0, 1/2]$ and even integer $q > 0$, the following holds.

Let $\mathbf{K}$ be a random variable following the weighted uniform distribution over $\mathbb{Z}_q$ below

$$\Pr(MSB(\mathbf{K}) = 0) = (1-b) \cdot \frac{1-\varepsilon}{q/2} + b\frac{\varepsilon}{q/2}$$

$$\Pr(MSB(\mathbf{K}) = 1) = b\frac{1-\varepsilon}{q/2} + (1-b) \cdot \frac{\varepsilon}{q/2}$$

Then, the modular bias of $\mathbf{K}$ is

$$B_q(\mathbf{K}) = (1 - 2\varepsilon) B_q(\mathbf{K}_b)$$

where $\mathbf{K}_b$ is uniformly distributed over $[bq/2, (b+1)q/2)$.

Note that the lemma above holds for odd q with an additive error of order $1/q$, which is negligible in practice.

C. Bleichenblacher's Attack Framework

Given such a bias function $|B_q(K)|$, a naive approach to find sk given an instance of the HNP would be to compute the set of candidate nonces $K_w = \{z_i + h_i w \mod q\}_{i=1}^M$ for all $w \in \mathbb{Z}_q$. The authors claim that for $w \neq sk$, the bias $|B_q(K_w)|$ is close to $1/\sqrt{M}$. However, if $w = sk$, $|B_q(K_w)|$

will be closer to 1. Therefore, by looking for a peak value on the sampled bias, the secret key could be found.

However, this method is not better than a naive exhaustive search over $\mathbb{Z}_q$. To circumvent this problem, the *collision search* of input samples is required as a preliminary step. This step is based on the following observation: by taking linear combinations of $\{(h_i, z_i)\}_{i=1}^{M}$ to generate M' new samples $\{(h'_j, z'_j)\}_{j=1}^{M'}$ such that $h'_j < L_{FFT}$, the peak's width broadens to approximately q/L_{FFT}. Thus, after computing these new samples, the number of candidate points reduces to L_{FFT} and since the Fast Fourier Transform (FFT) can be computed in $O(L_{FFT} \log L_{FFT})$ time and takes $O(L_{FFT})$ space, the value should be tweaked in order to find linear combinations that are small enough to make the FFT practically computable given the available resources.

However, this comes with a downside: in exchange for a broader peak width, the peak height gets reduced exponentially. Concretely, if the coefficients of the linear combinations are restricted to $\omega_{i,j} \in \{-1, 0, 1\}$ such that $\{(h'_j, z'_j)\}_{j=1}^{M'} = \{(\sum_i \omega_{i,j} h_i, \sum_i \omega_{i,j} z_i)\}_{j=1}^{M'}$ as the authors chose, the height of the bias decreases exponentially with the L_1-norm of the coefficient vector. Therefore, the new height is $|B_q(K)|^{\Omega_j}$ where $\Omega_j = \sum_i |\omega_{i,j}|$. This adds an extra constraint to the linear combinations: they must be sparse enough for the diminished peak height to be distinguishable from the noise floor, which is $1/\sqrt{M'}$ in average. The method used to find such linear combinations are explained in the following chapters.

Once the *collision search* step is done, the attack consists on computing the bias using the newly formed linear combinations with different candidate keys $w_n = nq/L_{FFT}$. The sampled bias after the collision search is

$$B_q(K_{w_n}) = \frac{1}{M'} \sum_{j=0}^{M'-1} e^{2\pi i (z'_i + h'_i \cdot sk)/q}$$

$$= \sum_{t=0}^{M'-1} \left(\frac{1}{M'} \sum_{\{j | h'_j = t\}} e^{2\pi i z'_j/q}\right) e^{2\pi i t n / M'}$$

Thus, by taking the term between parentheses and constructing a vector $Z = (Z_0, \ldots, Z_{L_{FFT}-1})$, the sampled bias can be computed by doing a FFT on Z. The algorithm then finds the value w_n where $|B_q(K_{w_n})|$ is the largest and outputs the most significant $\log L_{FFT}$ bits of w_n.

Once the top l' MSBs of sk have been recovered, the recovery of the remaining bits consists in running the same method again but removing the known part of the secret key from the hidden number in the HNP. Let $sk = sk_{hi} \cdot 2^{l-l'} + sk_{lo}$ (sk_{hi} is the l' MSBs of sk), the new instance of the HNP can be written as $k = z + h \cdot sk = (z + k \cdot sk_{hi} \cdot 2^{l-l'}) + h \cdot sk_{lo}$. This can be done until the number of remaining bits is low enough that an exhaustive search makes more sense.

D. $\mathcal{K}$-list sum problem

The $\mathcal{K}$-list sum problem, which is a subproblem of the Generalized Birthday Problem, has been used by the authors in the Bleichenbacher's attack where the value of $\mathcal{K}$ is set to 4 (4-list sum problem). The general definition of this problem is in *Definition 2.3* of the paper as follows:

Definition 2: Given $\mathcal{K}$ sorted lists $\mathcal{L}_1, \ldots, \mathcal{L}_{\mathcal{K}}$, each of which consists of 2^a uniformly random l-bit integers, the $\mathcal{K}$-list sum problem consists of finding a non-empty list $\mathcal{L}'$ consisting of $x' = \sum_{i=1}^{\mathcal{K}} \omega_i x_i$, where $\mathcal{K}$-tuples $(x_1, ..., x_{\mathcal{K}}) \in \mathcal{L}_1 \times ... \times \mathcal{L}_{\mathcal{K}}$ and $(\omega_1, ..., \omega_{\mathcal{K}}) \in \{-1, 0, 1\}^{\mathcal{K}}$ satisfy $MSB_n(x') = 0$ for some target parameter $n \leq l$

For example, considering the 4-list algorithm based on Howgrave-Graham-Joux (HGJ) which has been used in this paper, there are 4 sorted lists with a cardinality of 2^a. For each $c \in [0, 2^v)$, where v is a parameter of the algorithm, we form two sorted lists $\mathcal{L}'_1$ and $\mathcal{L}'_2$ respectively from $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, $\mathcal{L}_4$ s.t. $(x, y) \in \mathcal{L}_1 \times \mathcal{L}_2$ (resp. $\mathcal{L}_3 \times \mathcal{L}_4$) have their $MSB_{\mathbf{a}}(x+y) = \mathrm{c}$. Now, we look for tuples $(x', y') \in \mathcal{L}'_1 \times \mathcal{L}'_2$ s.t. $MSB_{\mathbf{n}}(|x' - y'|)$ cancel out. Output the absolute difference of the tuple values in the final output list $\mathcal{L}'_f$.

The 4-list sum algorithm is an efficient manner by which we can reduce the size of the candidate nonce samples in order to avoid brute forcing all the possible nonce values over $\mathbb{Z}_q$. Thus, the algorithm is used for collision search to generate M' samples which consists in finding pairs of linear combinations such that enough top bits are nullified to fulfill the smallness condition of all h'_j. Furthermore, each linear combination has only 4 terms and thus, $\Omega_j = 4$ for all j, which respects the sparsity condition if M' is large enough.

Algorithm 3 Bleichenbacher's attack framework

Require:
$\{(h_i, z_i)\}_{i=1}^{M}$ - HNP samples over $\mathbb{Z}_q$.
M' - Number of linear combinations to be found.
L_{FFT} - FFT table size.
Ensure: Most significant bits of sk
1: **Collision search**
2: Generate M' samples $\{(h'_j, z'_j)\}_{j=1}^{M'}$, where $(h'_j, z'_j) = \left(\sum_i \omega_{i,j} h_i, \sum_i \omega_{i,j} z_i\right)$ is a pair of linear combinations with the coefficients $\omega_{i,j} \in \{-1, 0, 1\}$, such that for $j \in [1, M']$
(1) *Small*: $0 \leq h'_j < L_{\mathrm{FFT}}$ and
(2) *Sparse*: $\left|B_q(\boldsymbol{K})\right|^{\Omega_j} \gg 1/\sqrt{M'}$ for all $j \in [1, M']$, where $\Omega_j := \sum_i |\omega_{i,j}|$.
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: $\quad Z_{h'_j} \leftarrow Z_{h'_j} + e^{(2\pi z'_j/q)\mathrm{i}}$
7: **end for**
8: $\left\{B_q(K_{w_i})\right\}_{i=0}^{L_{\mathrm{FFT}}-1} \leftarrow \mathrm{FFT}(Z)$, where $w_i = iq/L_{\mathrm{FFT}}$.
9: Find the value i such that $\left|B_q(K_{w_i})\right|$ is maximal.
10: Output most significant $\log L_{\mathrm{FFT}}$ bits of w_i.

E. Unified time-space-data tradeoffs

1) Tradeoffs for Parameterized 4-list Sum algorithm: The time-space tradeoffs analyzed in the previous works such as the HGJ approach made two artificial assumptions (1. cardinality of input, output samples and FFT table size is

equal in Bleichenbach's attack framework: $M = M' = L_{FFT}$; 2. The number of collided bits is a fixed constant in the 4-list sum algorithm) which greatly reduced flexibility for the optimization of the attacks.

The authors' motivation for introducing a third parameter of data complexity was followed by a practical approach where the attacker may want to trade "online" side-channel detection costs (data complexity) with "offline" computational costs (time and space complexity). Thus, they introduce a generalization of Dinur's tradeoff formula [5] for the parametrized 4-list sum algorithm in order to strike the optimal balance between the time, memory, and input data complexities:

$$2^4 M' N = TM^2 \Leftrightarrow m' = 3a + v - n$$

where $N = 2^n$, n being the number of top bits to be nullified; $M = 2^m = 4 \times 2^a$ is the number of input samples, 2^a being the length of each sublist; $M' = 2^{m'} \leq 2^{2a}$ is the number of output samples such that the top n bits are 0; $v \in [0, a]$ is a parameter deciding how many iterations of the collision search to be executed; $T = 2^t = 2^{a+v}$ is the time complexity.

This tradeoff formula is advantageous than its previous counterparts by giving more flexibility to the sample amplification, which is an important aspect to Bleichenbacher's framework as controlling the output samples in a way such that the signal stands above the noise floor indeed proves to be very useful for the attack framework.

2) Integration with Bleichenbacher and Linear Programming: Next, we integrate the tradeoff formula along with the two previously discussed important constraints of the Bleichenbach framework, i.e. small and sparse linear combinations. Thus, a linear programming problem is constructed from all the constraints whose goal is to optimize time, space and data complexities given a certain set of constraints. Given below the linear programming table for the 4-list sum algorithm used in the paper: we see that the first row gives the objective for each column whereas the rest of them denote the constraints imposed. For example, we see that in order to optimize the data complexity, the objective is to minimize the logarithm of number of input samples, m_{in} given $t_{max}, m_{max}, l_{FFT}$, slack parameter α, and estimated bias $B_q(\mathbf{K})$ as fixed constraint parameters. One can even iteratively solve the linear programming over various choices of r to find the optimal number of rounds leading to the best result, with at most 5 rounds needed for biases under 4 bits, making the process efficient. Fig.3 & 4 in the paper present the optimal time and data complexities for attacking a 1-bit biased HNP with varying FFT sizes and maximum memory bounds.

IV. Attack experiments and results

The authors have implemented the attack on a modified version of OpenSSL 1.0.2u (in a way that the MSB of the nonce is leaked with the same error rate as their experiments with side-channel attacks) for the P-192 and sect163r1 curves.

	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	$m_{i+1} = 3a_i + v_i - n_i$		$i \in [0, r-1]$
	$t_i = a_i + v_i$		$i \in [0, r-1]$
	$v_i \leq a_i$		$i \in [0, r-1]$
	$m_i = a_i + 2$		$i \in [0, r-1]$
	$m_{i+1} \leq 2a_i$		$i \in [0, r-1]$
	$m_{\text{in}} = m_0 + f$		
	$\ell \leq \ell_{\text{FFT}} + f + \sum_{i=0}^{r-1} n_i$		
	$m_r = 2(\log \alpha - 4^r \log(\lvert B_q(\boldsymbol{K})\rvert))$		

Fig. 3: Linear programming problems based on the 4-list sum problem

For the P-192 curve, they used 24 `r5.24xlarge` instances of AWS EC2 R5 with 96 vCPUs each to carry out the collision search step. Then, they used 2 `x1e.32xlarge` instances of AWS EC2 with $\approx$ 4TB of RAM each to store FFT tables of size $L_{FFT} = 2^{38}$ and conduct the search for the peak. The experiments were conducted with error rates of $\varepsilon = 0$ and $\varepsilon = 0.01$. The latter error rate was estimated through experiments from the authors on a Flush+Reload side channel attack on OpenSSL. The main results are summarized in the table below.

ε	Input	Output	L_{FFT}	Total time	Recovered MSBs
0	2^{29}	2^{27}	2^{38}	113.5h	39
0.01	2^{35}	2^{30}	2^{37}	64h	39

For the sect163r1 curve, the authors used a cluster of 16 $\times$ 16 core nodes with a total of 128GB of RAM to attack an HNP instance with $\varepsilon = 0$ and a single 48-core workstation with 512GB of RAM to attack an HNP instance with $\varepsilon = 0.027$. The main results are summarized in the table below.

ε	Input	Output	L_{FFT}	Total time	Recovered MSBs
0	2^{23}	2^{27}	2^{35}	8h	36
0.027	2^{24}	2^{29}	2^{34}	43h	35

They observed that for both experiments, the expected number of most significant key bits ($\log L_{FFT}$) were recovered and the detected peak sizes matched the estimates.

References

[1] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom, "Ladderleak: Breaking ecdsa with less than one bit of nonce leakage," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 225–242.

[2] D. F. Aranha, P.-A. Fouque, B. Gérard, J.-G. Kammerer, M. Tibouchi, and J.-C. Zapalowicz, "Glv/gls decomposition, power analysis, and attacks on ecdsa signatures with single-bit nonce bias," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 262–281.

[3] A. Takahashi, M. Tibouchi, and M. Abe, "New bleichenbacher records: Fault attacks on qdsa signatures," *Cryptology ePrint Archive*, 2018.

[4] D. Bleichenbacher, "On the generation of one-time keys in dl signature schemes," in *Presentation at IEEE P1363 working group meeting*, 2000, pp. 81–83.

[5] I. Dinur, "An algorithmic framework for the generalized birthday problem," *Designs, Codes and Cryptography*, vol. 87, pp. 1897–1926, 2019.