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

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

[1]Kyoto University

[2]NTT Corporation

September 19, 2022

Contents

A Warmup of Schnorr Signature

To sign a message m:

- Choose a randomness k (usually called nonce).
- Let $r = g^k$ (g is the generator of some group) and $e = H(r||m)$.
- Signature $s = k - \alpha \cdot e$ (α is the signing key)
- Careless implementation easily leads to key recovery.

Scenario One: Nonce Reuse

- Never reuse the same nonce.
- $k = s_1 + \alpha \cdot e_1 = s_2 + \alpha \cdot e_2 \rightarrow \alpha = (s_1 - s_2)/(e_2 - e_1)$.

bbc.com/news/technology-12116051

iPhone hacker publishes secret Sony PlayStation 3 key

By Jonathan Fildes
Technology reporter, BBC News

6 January 2011

Scenario Two: Partial Nonce Leakage

- Small partial nonce leakage leads to key recovery: attacking 256-bit ECDSA with 5-bit (even less) nonce leakage is pretty easy.
- A lot of practical examples: Attacks on android phones, bitcoins.

Paper 2016/231

Side-Channel Analysis of Weierstrass and Koblitz Curve ECDSA on Android Smartphones

Pierre Belgarric, Pierre-Alain Fouque, Gilles Macario-Rat, and Mehdi Tibouchi

Paper 2019/023

Biased Nonce Sense: Lattice Attacks against Weak ECDSA Signatures in Cryptocurrencies

Joachim Breitner and Nadia Heninger

Two Types of Attacks

Two ways to attack biased nonce (EC)DSA (same as Schnorr signature):

- Original approach: Bleichenbacher attack [Ble00] rely on Fourier analysis techniques.
- Lattice reduction (our main focus in this work) also provides an attack technique.

Lattice Attacks on (EC)DSA

- Attacking (EC)DSA with nonce leakage reduces to a lattice problem [HGS01, NS02].
- When nonce leakage is small, lattice attacks on (EC)DSA become more difficult.
- In 2013, Liu and Nguyen Attacked 160-bit DSA with 2-bit leakage, using BKZ 2.0 with block size 90 [LN13].
- At Eurocypt 2021, Albrecht and Heninger [AH21] use G6K together with idea of predicate sieving to break new records.

Comparison of Bleichenbacher Attack and Lattice Attack

	leakage			
modulus	4-bit	3-bit	2-bit	1-bit
160-bit	✓	✓	[LN13,AH21]	×
256-bit	✓	[AH21]	×	×
384-bit	[AH21]	×	×	×

Table: Reported results of lattice attacks on (EC)DSA

	leakage		
modulus	3-bit	2-bit	1-bit
160-bit	✓	✓	[AFG+14b,ANT+20]
256-bit	✓	[TTA18]	×

Table: Reported results of Bleichenbacher attacks on (EC)DSA

Comparison of Bleichenbacher Attack and Lattice Attack

- Bleichenbacher attack can deal with small nonce leakage at the cost of using more signatures. For typical parameters, Bleichenbacher attack uses 2^{27} signatures and lattice attacks use around 100 signatures.
- At each iteration of Bleichenbacher attack, some bits of the signing key is recovered. With more known bits of the signing key, the attack becomes easier and easier. Lattice attacks are generally all-or-nothing.

Our Work

- [LN13] and [AH21] are improving lattice reduction algorithm itself.
- In this work, the starting point is slightly different: to understand some relation between Bleichenbacher attacks and lattice attacks [Tib17, Hen20].
- Concretely, we can ask several questions:
  - Lattice attacks on (EC)DSA are in general all-or-nothing. If some bits of the signing key is known, does it make the attack easier?
  - Can we exploit the special structure of the lattice?
  - By comparison with Bleichenbacher attack, if given many more signatures, can we improve lattice attacks?
- We give positive answers and do practical experiments following [JSSS20].

Contents

Lattice Attacks on (EC)DSA

Lattice Attacks on baised (EC)DSA amounts to the problem:

- Given d pairs of signatures $(r_i, s_i)(i = 1, \cdots, d)$ signed with biased nonce, we construct (t_i, u_i) s.t. $0 < \alpha t_i - u_i + c_i q < q/2^l$ where α is the signing key and l is the number of leakage.Then construct the lattice:

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

- $\mathbf{u} = (2^{l+1}u_1, 2^{l+1}u_2, \cdots 2^{l+1}u_d, 0)$ is the target point, and the hidden close lattice point is $\mathbf{v} = (2^{l+1}t_1\alpha + c_1 2^{l+1}q, \cdots, 2^{l+1}t_d\alpha + c_d 2^{l+1}q, \alpha)$ thus we can solve this by nearest plane algorithm or Kannan embedding method.

Contents

Intuitive Ideas to Improve the Attack

- Consider the case of BDD (bounded distance decoding) or CVP, denote λ_1 as the length of shortest vector in the original lattice and e as the distance between the target vector and the lattice.
- Intuitively, in order to improve the attack, increase the ratio $\frac{\lambda_1}{e}$.
- Equivalently, if viewed in the embedded lattice (by Kannan embedding), it becomes a unique-SVP instance and the goal is to increase the gap betwwen $\lambda_2^{'}$ and $\lambda_1^{'}$.
- To increase λ_1, according to Gaussian heuristic, $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d}{2\pi e}}\text{vol}(\mathcal{L})^{1/d}$, the goal is to increase the volume of the lattice.

Attempt 1: Increase the Volume

Recall that the lattice we construct is:

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

The target vector is $(2^{l+1}u_1 + q, 2^{l+1}u_2 + q, \cdots, 2^{l+1}u_d + q, 0)$, and the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \cdots \alpha 2^{l+1}t_d + c_d 2^{l+1}q, \alpha)$. Denote the difference vector between them as $\mathbf{e}$.

- *Each coordinate of* $\mathbf{e}$ *is upper-bounded by* q.

Attempt 1: Increase the Volume

We could modify the lattice as

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

- Increase the volume by 2^{100}.
- Problem is that the hidden lattice vector will not be close to the target vector anymore, because the hidden lattice vector is $(\alpha 2^{l+1}t_1 + c_1 2^{l+1}q, \alpha 2^{l+1}t_2 + c_2 2^{l+1}q, \cdots \alpha 2^{l+1}t_d + c_d 2^{l+1}q, 2^{100}\alpha)$.

Attempt 2: Enumerating Bits of Signing Key

- Denote $\alpha = \alpha_1 \cdot 2^c + \alpha_2$ $(0 \leq \alpha_2 < 2^c)$.
- Substitute α with $\alpha_1 \cdot 2^c + \alpha_2$ in $|\alpha \cdot t_i - u_i|_q < q/2^l$ $(i = 1, 2, \cdots d)$.
- $|\alpha_1 \cdot 2^c \cdot t_i + \alpha_2 \cdot t_i - u_i|_q < q/2^l$ $(i = 1, 2, \cdots d)$.
- Then set

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

so we have new instance for α_1:

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

- However, the remaining instance seems to be as hard as the original one.

Combine Attempt 1 and 2

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

- Increase the volume by 2^{100}.
- If signing key α only has 60 bits, $2^{100}\alpha$ is upperbounded by $2^{160} \approx q$.

Combine Attempt 1 and 2

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

- Enumerate α by c bits.
- Increase the volume by a factor of 2^c.
- The length of difference vector $\mathbf{e}$ does not change, thus improving the attack.

Guessing Bits of Nonce

- Variants of the idea lead to different attacks: Enumerate addtional bits of the nonces.
- Example: suppose that we are in the context of 160-bit group with 2-bit nonce leakage and we try to enumerate the third LSB of the nonce.

Guessing Bits of Nonce

- Enumerating all the bits: huge computation.
- We could do it for part of the signatures, thus constructing a hybrid lattice.
- For instance, we can guess one more bit for 20 out of the 100 signatures and keep the other 80 signatures unchanged:

$$B = \begin{pmatrix} 16q & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ & \vdots & 16q & 0 & & \vdots & \vdots \\ 0 & \cdots & 0 & 8q & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \vdots & \cdots & \vdots & \vdots \\ 16t'_1 & \cdots & 16t'_{20} & 8t_{21} & \cdots & 8t_{100} & 1 \end{pmatrix}$$

Then we perform the lattice attacks on the new matrix at the cost of 2^{20} operations for guessing bits.

Advantage of the Guessing Bits Attacks

- Very easy to simulate: assume we guessed the correct bits for self-generated instances, saving a lot of computation and making estimate of cost easy.
- Very easy to parallelize.
- Batch CVP (SVP) and CVP with pre-processing techniques.
- Compatible with existing techinques: [LN13, JSSS20, AH21]. Straightfoward to combine with those techniques.

Additional Contributions

- One more variant of the attacks: utilize many more signatures to improve lattice attacks.
- An explanation of gap between CVP and SVP approaches: difference between nearest plane and Kannan embedding (observed in [JSSS20]).
- Discussion on choosing the Kannan embedding factor.
- A theoretical analysis of lattice attacks on (EC)DSA.

Experimental Result

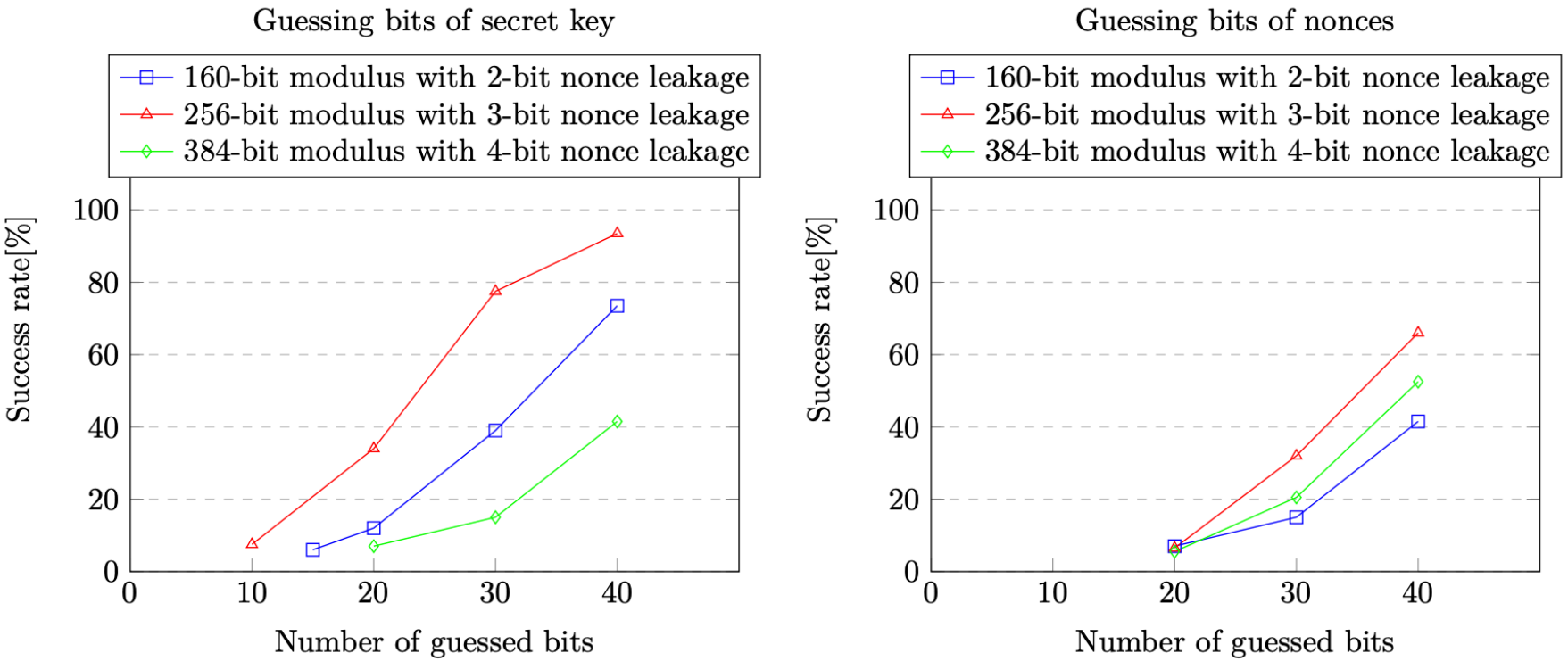

Figure 2: Experimental results: guessing bits of the secret key vs. of the nonces.

Experimental Result

- Experiments on TPM-Fail dataset [MSEH20, JSSS20].
- With only 800 signatures (40000 in [MSEH20], 900 in [JSSS20]), we are able to recover the signing key.

Contents

Summary

- We propose new ways to improve lattice attacks on (EC)DSA: guessing bits of the secret key or nonces.
- This solves difficult cases for lattice attacks.
- Easy to simulate and parallelize.
- Experimental results validated our ideas.

Thank you for your attention!