Minerva: The curse of ECDSA nonces

Jan Jancar, Vladimir Sedlacek, Petr Svenda, Marek Sys
Minerva: The curse of ECDSA nonces
Jan Jancar, Vladimir Sedlacek, Petr Svenda, Marek Sys
Systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces
Discovery

**EC Tester**

- Tool for testing black-box ECC implementations
  - JavaCards
  - Software libraries (15 supported)

- **Idea:** Independently verify implementations are well-behaved and do not contain bugs

- 12 test suites

- [crocs-muni/ECTester](https://github.com/crocs-muni/ECTester)
**Discovery**

**ECDSA**

\[ y^2 \equiv x^3 + ax + b \mod n \]

\[ G \in E(\mathbb{F}_p), \ |G| = n \] (prime)

\[ \text{Sign}(\text{message } m, \text{ private key } x) \]

1. \( k \leftarrow \mathbb{Z}_n \) (nonce)
2. \( r \equiv ([k]G)_x \mod n \)
3. \( s \equiv k^{-1}(H(m) + rx) \mod n \)
4. Output \((r, s)\) as ASN.1 DER SEQUENCE
Discovery

ECDSA tests

- ASN.1 parsing ✓
ASD1 parsing ✓
Signature malleability ✓
Discovery

ECDSA tests

- ASN.1 parsing  
- Signature malleability  
- Test-vectors  

Let's test timing as well!

Minerva: The curse of ECDSA nonces
<table>
<thead>
<tr>
<th>ECDSA tests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN.1 parsing</td>
<td>✓</td>
</tr>
<tr>
<td>Signature malleability</td>
<td>✓</td>
</tr>
<tr>
<td>Test-vectors</td>
<td>🛑</td>
</tr>
<tr>
<td>Nonce randomness</td>
<td>✓</td>
</tr>
</tbody>
</table>

Let’s test timing as well!

Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✔
- Signature malleability ✔
- Test-vectors 🔴
- Nonce randomness ✔

Let’s test timing as well!
Discovery

ECDSA tests

- ASN.1 parsing ✓
- Signature malleability ✓
- Test-vectors ~
- Nonce randomness ✓

Let’s test timing as well!
Discovery

ECDSA tests

- ASN.1 parsing
- Signature malleability
- Test-vectors
- Nonce randomness

Let’s test timing as well!
Discovery

ECDSA tests

- ASN.1 parsing ✔
- Signature malleability ✔
- Test-vectors ✗
- Nonce randomness ✔

Let's test timing as well!

Jan Jancar
Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✓
- Signature malleability ✓
- Test-vectors ↳
- Nonce randomness ✓

Let’s test timing as well!

Jan Jancar
Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✔
- Signature malleability ✔
- Test-vectors ~
- Nonce randomness ✔

Let’s test timing as well!
ECDSA tests

- ASN.1 parsing ✓
- Signature malleability ✓
- Test-vectors 🔴
- Nonce randomness ✓

Let’s test timing as well!
Discovery

ECDSA tests

- ASN.1 parsing ✓
- Signature malleability ✓
- Test-vectors
- Nonce randomness ✓

Let’s test timing as well!
Discovery

ECDSA tests

- ASN.1 parsing  ✓
- Signature malleability  ✓
- Test-vectors  ❌
- Nonce randomness  ✓
- Timing  ❌

Let's test timing as well!

Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✓
- Signature malleability ✓
- Test-vectors
- Nonce randomness ✓
- Timing ×

Minerva

Let's test timing as well!

Athena IDProtect

Crypto++

Jan Jancar

Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✔
- Signature malleability ✔
- Test-vectors 🕰
- Nonce randomness ✔
- Timing 🕒

Let's test timing as well!

Minerva

Minerva

1

TPM-FAIL

2

Crypto++

Athena IDProtect

Jan Jancar

Minerva: The curse of ECDSA nonces
Discovery

ECDSA tests

- ASN.1 parsing ✔
- Signature malleability ✔
- Test-vectors ❄
- Nonce randomness ✔
- Timing ❌

Let's test timing as well!

Jan Jancar

Minerva: The curse of ECDSA nonces
## Discovery

### Tested

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Version/Model</th>
<th>Scalar multiplier</th>
<th>Leakage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>OpenSSL</td>
<td>1.1.1d</td>
<td>Montgomery ladder</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>BouncyCastle</td>
<td>1.58</td>
<td>Comb method</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>SunEC</td>
<td>JDK 7 - JDK 12</td>
<td>Window-NAF</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>WolfSSL</td>
<td>4.0.0</td>
<td>Sliding window</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>BoringSSL</td>
<td>974f4dddf</td>
<td>Window method</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>libtomcrypt</td>
<td>v1.18.2</td>
<td>Sliding window</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>libgcrypt</td>
<td>1.8.4</td>
<td>Double-and-add</td>
<td>yes*</td>
</tr>
<tr>
<td></td>
<td>Botan</td>
<td>2.11.0</td>
<td>Window method</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Microsoft CNG</td>
<td>10.0.17134.0</td>
<td>Window method</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>mbedTLS</td>
<td>2.16.0</td>
<td>Comb method</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>MatrixSSL</td>
<td>4.2.1</td>
<td>Sliding window</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Intel PP Crypto</td>
<td>2020</td>
<td>Window-NAF</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Crypto++</td>
<td>8.2</td>
<td>unknown</td>
<td>yes</td>
</tr>
<tr>
<td>Card</td>
<td>Athena IDProtect</td>
<td>010b.0352.0005</td>
<td>unknown</td>
<td>yes*</td>
</tr>
<tr>
<td></td>
<td>NXP JCOP3</td>
<td>J2A081, J2D081, J3H145</td>
<td>unknown</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Infineon JTOP</td>
<td>52GLA080AL, SLE78</td>
<td>unknown</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>G+D SmartCafe</td>
<td>v6, v7</td>
<td>unknown</td>
<td>no</td>
</tr>
</tbody>
</table>
Discover

Leak

Athena IDProtect

libgcrypt

Crypto++

SunEC/Java

MatrixSSL

WolfSSL

Jan Jancar

Minerva: The curse of ECDSA nonces
**Discovery**

**Leak**

\[ [k]G \]

- Nonce preparation
- EC multiplication operation
- Nonce-length dependency

One loop iteration

256 bits effective nonce length
255 bits effective nonce length
251 bits effective nonce length
Discovery

Leak

\[ [k]G \]

Jan Jancar

Minerva: The curse of ECDSA nonces
\[ L = \text{base} + \text{iter \_ time} \cdot B + N \]

\[ B \sim \text{Geom}(p = 1/2, (256, 255, \ldots, 0)) \]

\[ N \sim \text{Norm}(0, sdev^2) \]

// secp256r1 curve
Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise
Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes
(Extended Abstract) [1]

Dan Boneh\textsuperscript{1}
Princeton University

Ramarathnam Venkatesan\textsuperscript{2}
Bellcore
Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hidden Number Problem (HNP) [1]

Given an oracle computing:

\[ O_{b,t}(a) = \text{MSB}_l(at + b \mod n) \]

with \( t \) u.i.d. in \( \mathbb{Z}_n^* \), find \( a \).
Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hidden Number Problem (HNP) [1]

Given an oracle computing:

\[ O_{r,s}(r) = \text{MSB}_l(k \mod n) \]
Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hidden Number Problem (HNP) [1]

Given an oracle computing:

\[ O_{r,s}(r) = \text{MSB}_l(xs^{-1}r + H(m)s^{-1} \mod n) \]

find \( x \).
Exploitation

Basic attack [2]

- Collect $N$ signatures, take $d$ of the fastest
Exploitation

Basic attack [2]

- Collect $N$ signatures, take $d$ of the fastest
- Assume some bounds $l_i$: $|k_i| = |x_{t_i} - u_i| = |x_s^{-1}r_i + H(m_i)s^{-1}| < n/2^{l_i}$
Exploitation

Basic attack [2]

- Collect $N$ signatures, take $d$ of the fastest
- Assume some bounds $l_i$: $|k_i| = |x t_i - u_i| = |x s_i^{-1} r_i + H(m_i) s_i^{-1}| < n/2^{l_i}$
- Construct a lattice with basis $B$ and reduce it:

$$B = \begin{pmatrix} 2^{l_1} n & 0 & 0 & \ldots & 0 & 0 \\ 0 & 2^{l_2} n & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 2^{l_d} n & 0 \\ 2^{l_1} t_1 & 2^{l_2} t_2 & 2^{l_3} t_3 & \ldots & 2^{l_d} t_d & 1 \end{pmatrix}$$
Exploitation

Basic attack [2]  

- Collect $N$ signatures, take $d$ of the fastest
- Assume some bounds $l_i$: $|k_i| = |x t_i - u_i| = |x s_i^{-1} r_i + H(m_i) s_i^{-1}| < n/2^{l_i}$
- Construct a lattice with basis $B$ and reduce it:

$$B = \begin{pmatrix}
2^{l_1} n & 0 & 0 & \ldots & 0 & 0 \\
0 & 2^{l_2} n & 0 & \ldots & 0 & 0 \\
& & \vdots & & & \\
0 & 0 & 0 & \ldots & 2^{l_d} n & 0 \\
2^{l_1} t_1 & 2^{l_2} t_2 & 2^{l_3} t_3 & \ldots & 2^{l_d} t_d & 1
\end{pmatrix}$$

- Construct a target $u = (2^{l_1} u_1, \ldots, 2^{l_d} u_d, 0)$
Exploitation

Basic attack [2]

- Collect $N$ signatures, take $d$ of the fastest
- Assume some bounds $l_i$: $|k_i| = |x t_i - u_i| = |x s_i^{-1} r_i + H(m_i) s_i^{-1}| < n/2^{l_i}$
- Construct a lattice with basis $B$ and reduce it:

$$
B = \begin{pmatrix}
2^{l_1} n & 0 & 0 & \ldots & 0 & 0 \\
0 & 2^{l_2} n & 0 & \ldots & 0 & 0 \\
0 & 0 & 2^{l_3} n & \ldots & 0 & 0 \\
0 & 0 & 0 & \ldots & 2^{l_d} n & 0 \\
2^{l_1} t_1 & 2^{l_2} t_2 & 2^{l_3} t_3 & \ldots & 2^{l_d} t_d & 1
\end{pmatrix}
$$

- Construct a target $u = (2^{l_1} u_1, \ldots, 2^{l_d} u_d, 0)$
- Solve CVP($B$, $u$). The closest lattice point is often: $v = (2^{l_1} t_1 x, \ldots, 2^{l_d} t_d x, x)$
Exploitation

Basic attack [2]

- Collect $N$ signatures, take $d$ of the fastest
- Assume some bounds $l_i$: $|k_i| = |xt_i - u_i| = |xs_i^{-1}r_i + H(m_i)s_i^{-1}| < n/2^{l_i}$
- Construct a lattice with basis $\mathbf{B}$ and reduce it:

$$
\mathbf{B} = \begin{pmatrix}
2^{l_1}n & 0 & 0 & \ldots & 0 & 0 \\
0 & 2^{l_2}n & 0 & \ldots & 0 & 0 \\
0 & 0 & 2^{l_3}n & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 2^{l_d}n & 0 \\
2^{l_1}t_1 & 2^{l_2}t_2 & 2^{l_3}t_3 & \ldots & 2^{l_d}t_d & 1
\end{pmatrix}
$$

- Construct a target $\mathbf{u} = (2^{l_1}u_1, \ldots, 2^{l_d}u_d, 0)$
- Solve CVP($\mathbf{B}, \mathbf{u}$). The closest lattice point is often: $\mathbf{v} = (2^{l_1}t_1x, \ldots, 2^{l_d}t_dx, x)$
- Because $\forall i$: $(xt_i - u_i) \mod n$ is small
Analysis

- Can we improve the attack?
- How to choose $l_i$, $d$ and minimize $N$?
- 4 datasets of signatures, varying noise, secp256r1 curve
- Run attack 5 times,
  - for each $N$ from 500 to 10,000 (steps 100) and
  - $d$ from 50 to 140 (steps 2)
- Solve via SVP (search reduced basis vectors), after BKZ reduction with $\beta \in \{15, 20, \ldots, 55\}$

<table>
<thead>
<tr>
<th>Dataset</th>
<th>base (µs)</th>
<th>iter_time (µs)</th>
<th>sdev (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>453.4</td>
<td>12.7</td>
<td>17.2</td>
</tr>
<tr>
<td>tpm</td>
<td>27047.3</td>
<td>236.1</td>
<td>211.3</td>
</tr>
<tr>
<td>card</td>
<td>43578.4</td>
<td>371.5</td>
<td>451.3</td>
</tr>
</tbody>
</table>
How to assign bound $l_i$ for the $i$-th fastest signature?
- Constant ($l_i = c$ for $c \in \{1, 2, 3, 4\}$) based on $d$
Analysis

Bounds $l_i$

- How to assign bound $l_i$ for the $i$-th fastest signature?
  - Constant ($l_i = c$ for $c \in \{1, 2, 3, 4\}$) based on $d$
  - Geometric based on $N$ (**new**)
Analysis

Bounds $l_i$

- How to assign bound $l_i$ for the $i$-th fastest signature?
  - Constant ($l_i = c$ for $c \in \{1, 2, 3, 4\}$) based on $d$
  - Geometric based on $N$ (new)

constant $c = 3$

geometric
How to assign bound $l_i$ for the $i$-th fastest signature?

- Constant ($l_i = c$ for $c \in \{1, 2, 3, 4\}$) based on $d$
- Geometric based on $N$ (ﷺ new)
Analysis

CVP vs SVP

- Solve the HNP via the natural CVP(\(B, u\)) or transform into SVP(\(C\))?
- CVP solved via Babai's Nearest Plane algorithm after reduction
- SVP solved via search of the reduced basis vectors

\[
C = \begin{pmatrix} B & 0 \\ u & n \end{pmatrix}
\]

![Graph showing success probability vs number of signatures]
Recentering

- Nonces are non-negative, we bound the absolute value
- Gain 1 bit by recentering! [5] [3]

\[|k_i - n/2^{l_i+1}| < n/2^{l_i+1}\]
Avoid errors by using a random subset of signatures! [2]

Sample $d$ random signatures out of the $1.5d$ fastest signatures, repeat 100 times

Time consuming
Instead of bounding $|k_i|$, we can bound $|k_i - k_j|$! [5] [6]

Strive for $l_i = l_j$, information is lost otherwise

Errors might cancel out

Cannot use recentering, difference might be negative

$$|k_i - k_j| < n/2^{\min(l_i,l_j)}$$
Lattice reduction is the costly part of the attack

In solving via CVP, the $u$ values are not part of the lattice

Reduce the lattice once, then try Babai’s Nearest Plane with many $u$

(new)
Lattice reduction is the costly part of the attack
In solving via CVP, the $u$ values are not part of the lattice
Reduce the lattice once, then try Babai’s Nearest Plane with many $u$
(new)
Analysis

**u-bitflips**

- Lattice reduction is the costly part of the attack
- In solving via CVP, the $u$ values are not part of the lattice
- Reduce the lattice once, then try Babai’s Nearest Plane with many $u$
- (new)
Conclusions

- Geometric assignment of bounds lowers $\min(N)$ for attack success
- SVP solving of HNP outperforms CVP solving via Nearest Plane algorithm
- Recentering improves the attack’s success rate
- Correcting errors via $u$-bitflips is promising, but pays the cost of using Nearest Plane algorithm
- Demonstrated an attack on data from the TPM-FAIL paper [3], with only 900 signatures, instead of 40 000
Thanks!

✉️ jan@neuromancer.sk

The paper: minerva.crocs.fi.muni.cz

Icons from ● X ■ Noun Project & ⚖️ Font Awesome

Photos from 📷 Unsplash
References

1. Dan Boneh, Ramarathnam Venkatesan; **Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes**
2. Billy Bob Brumley, Nicola Tuveri; **Remote Timing Attacks are Still Practical**
3. Daniel Moghimi, Berk Sunar, Thomas Eisenbarth, Nadia Heninger; **TPM-FAIL: TPM meets Timing and Lattice Attacks**
4. Sohaib ul Hassan, Iaroslav Gridin, Ignacio M. Delgado-Lozano, Cesar Pereida García, Jesús-Javier Chi-Domínguez, Alejandro Cabrera Aldaya, Billy Bob Brumley; **Déjà Vu: Side-Channel Analysis of Mozilla’s NSS**
5. Joachim Breitner, Nadia Heninger; **Biased Nonce Sense: Lattice Attacks against Weak ECDSA Signatures in Cryptocurrencies**
6. Jean-Charles Faugere, Christopher Goyet, Guénaël Renault; **Attacking (EC)DSA given only an implicit hint**