MUNI
FI

CROCS

Centre for Research on
Cryptography and Security

CHES
2020

Minerva:
The curse of ECDSA nonces

Jan Jancar, Vladimir Sedlacek,
Petr Svenda, Marek Sys

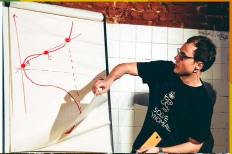
MUNI
FI

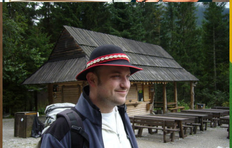
CROCS
Centre for Research on
Cryptography and Security

CHES
2020

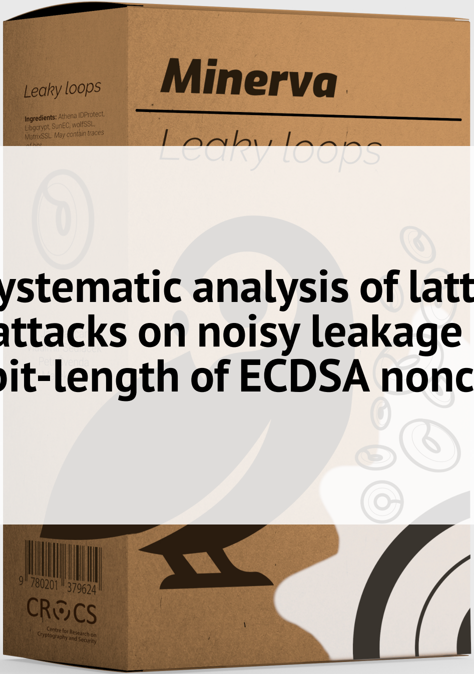
Minerva:
The curse of ECDSA nonces

Jan Jancar, Vladimir Sedlacek,
Petr Svenda, Marek Sys



Leaky loops
Ingredients: Athena IDProtect, Libgcrypt, SunEC, wolfSSL, MatrixSSL. May contain traces of bits.
Ján Jančár
Vladimír Sedláček
Petr Švenda
Marek Sýs
9 780201 379624
CROCS
Centre for Research on Cryptography and Security
Minerva
Leaky loops



Systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces

Discovery

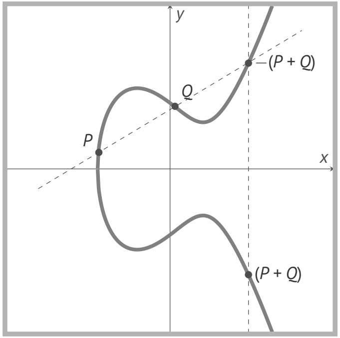
- Tool for testing black-box ECC implementations
  - JavaCards
  - Software libraries (15 supported)
- **Idea:** Independently verify implementations are well-behaved and do not contain bugs
- 12 test suites
- **crocs-muni/ECTester**

Discovery

ECDSA

Sign(message m, private key x)

1. $k \xleftarrow{\$} \mathbb{Z}_n$ (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`



$$y^2 \equiv x^3 + ax + b \quad \text{over } \mathbb{F}_p$$

$$G \in E(\mathbb{F}_p), |G| = n \quad \text{(prime)}$$

Discovery

ECDSA tests

- ASN.1 parsing ✅

Discovery

ECDSA tests

- ASN.1 parsing ✅
- Signature malleability ✅

Discovery

ECDSA tests

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

Discovery

ECDSA tests

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

Discovery

ECDSA tests

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

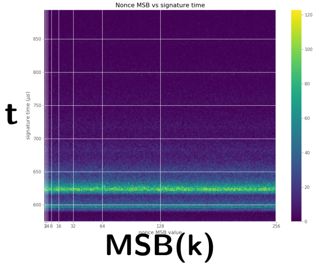
Let's test timing as well!

Discovery

ECDSA tests

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

Let's test timing as well!

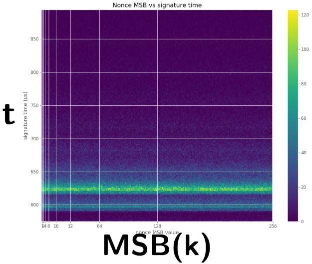


Discovery

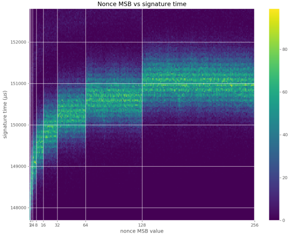
ECDSA tests

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

Let's test timing as well!



Athena IDProtect



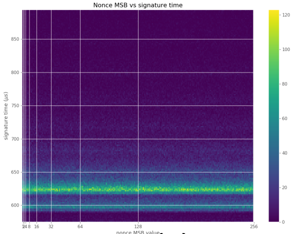
Discovery

ECDSA tests

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

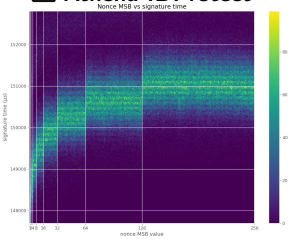
Let's test timing as well!

t

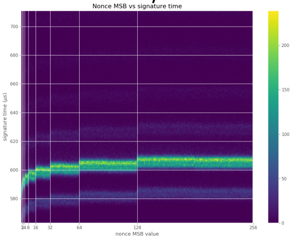


MSB(k)

Athena IDProtect



SunEC/Java

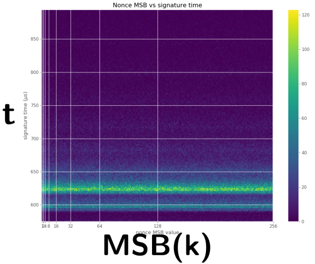


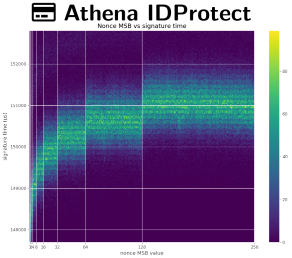
Discovery

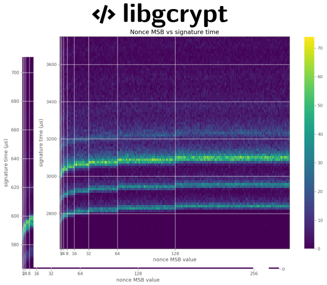
ECDSA tests

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

Let's test timing as well!





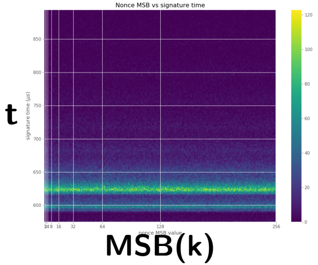


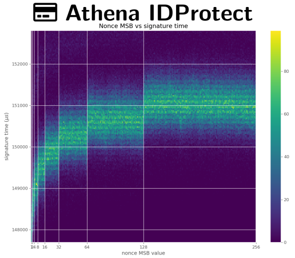
Discovery

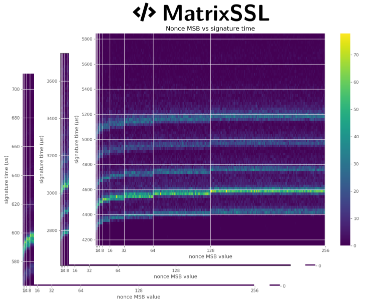
ECDSA tests

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

Let’s test timing as well!





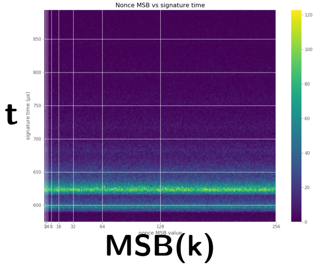


Discovery

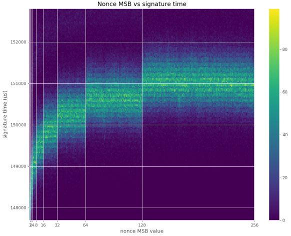
ECDSA tests

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

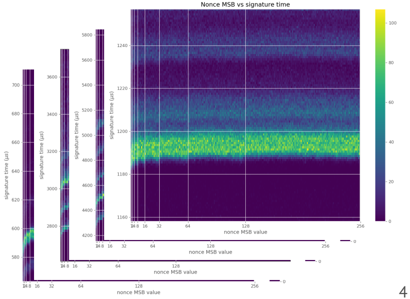
Let's test timing as well!



Athena IDProtect



WolfSSL

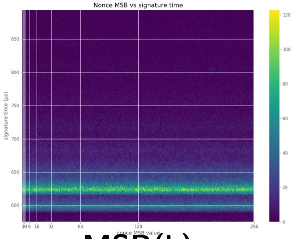


Discovery

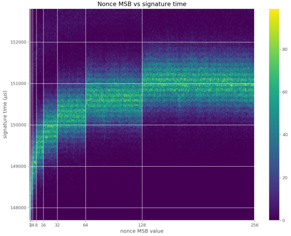
ECDSA tests

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

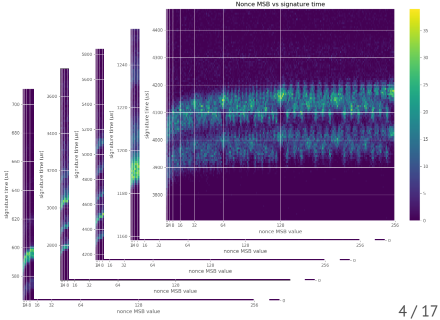
Let's test timing as well!



Athena IDProtect



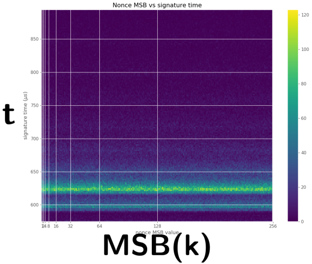
Crypto++

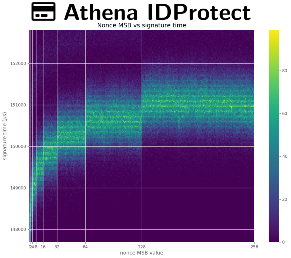


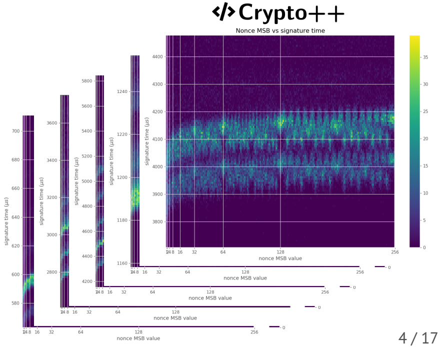
Discovery

ECDSA tests

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





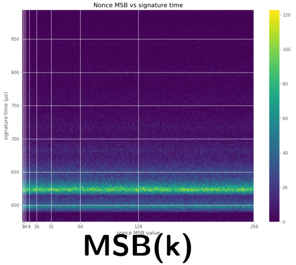


Discovery

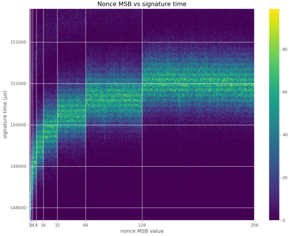
ECDSA tests

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

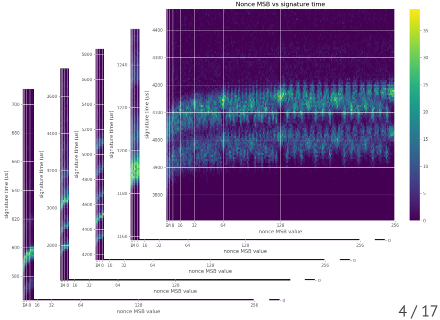
Minerva $\frac{1}{5}$



Athena IDProtect



Crypto++



Discovery

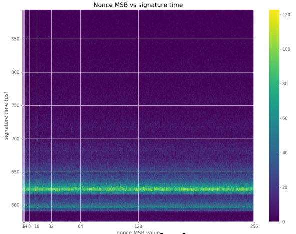
ECDSA tests

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

Minerva 1 / 5

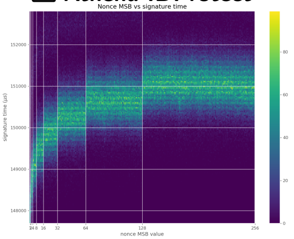
TPM-FAIL 2

t

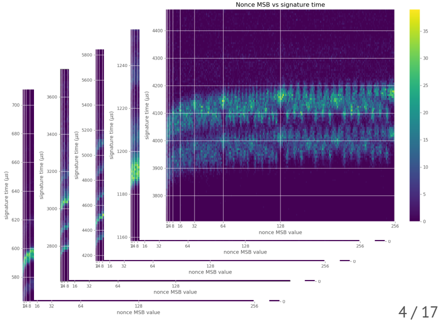


MSB(k)

Athena IDProtect



</> Crypto++



Discovery

ECDSA tests

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

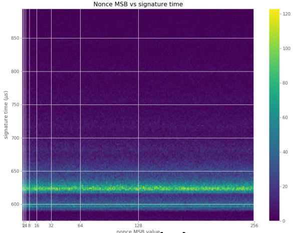
Minerva 15

Déjà Vu 1

TPM-FAIL 2

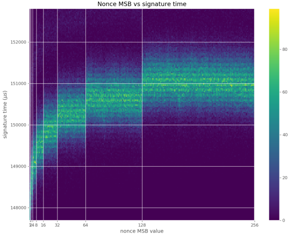
...

t

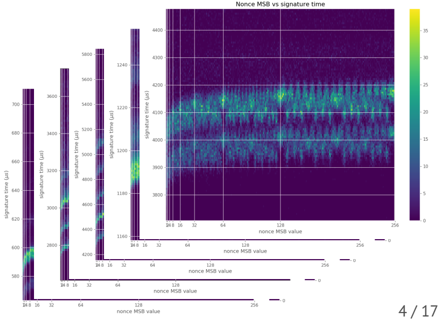


MSB(k)

Athena IDProtect



Crypto++



Discovery

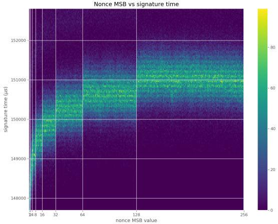
Tested

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

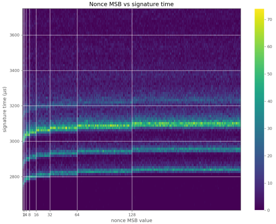
Discovery

Leak

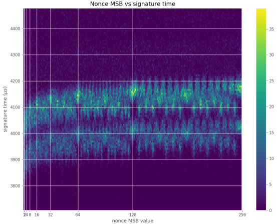
Athena IDProtect



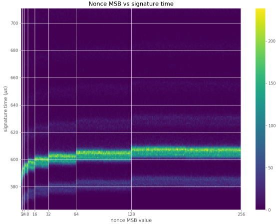
libgcrypt



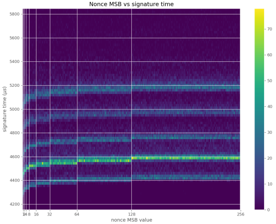
Crypto++



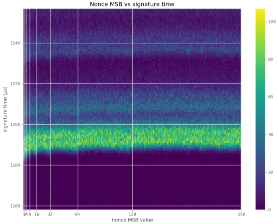
SunEC/Java



MatrixSSL



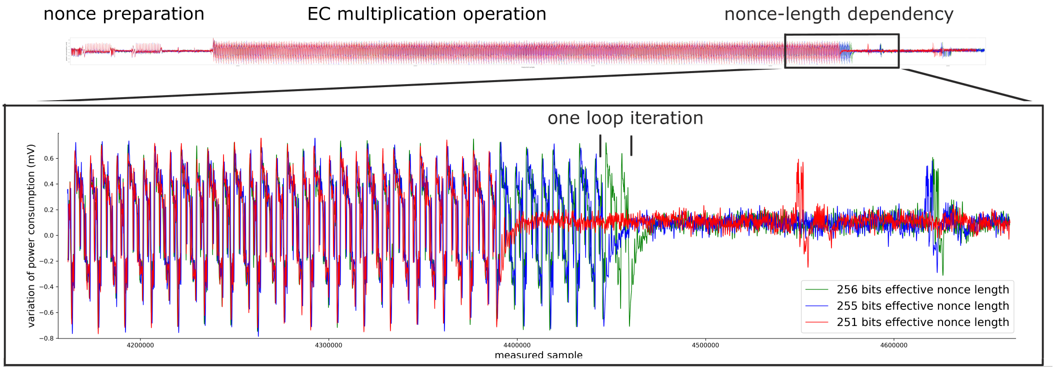
WolfSSL



Discovery

Leak

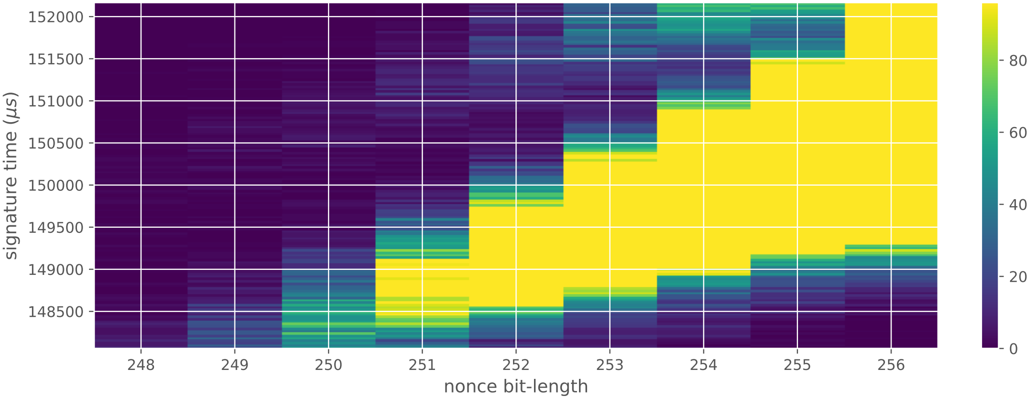
[k]G



Discovery

Leak

[k]G



Discovery

Leak

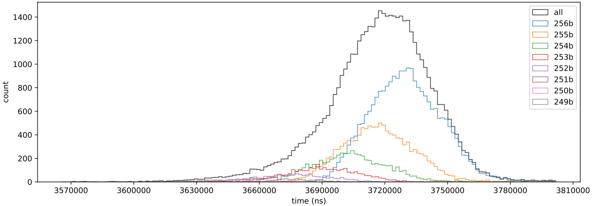
base
iter_time
sdev

$$\mathbf{L} = base + iter\_time \cdot \mathbf{B} + \mathbf{N}$$

$$\mathbf{B} \sim \mathbf{Geom}(p = 1/2, (256, 255, \ldots, 0))$$

// `secp256r1` curve

$$\mathbf{N} \sim \mathbf{Norm}(0, sdev^2)$$



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[1]
Princeton University

RAMARATHNAM VENKATESAN[2]
Bellcore

Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hidden Number Problem (HNP) [1]

Given an oracle computing:

$$\mathcal{O}_{b,t}() = \mathrm{MSB}_l(at + b \mod n)$$

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:

$$\mathcal{O}_{r,s}() = \mathrm{MSB}_l(k \mod n)$$

Exploitation

Hidden Number Problem

- Average 1 LZB per signature
- There is noise

Hidden Number Problem (HNP) [1]

Given an oracle computing:

$$\mathcal{O}_{r,s}() = \mathrm{MSB}_l(xs^{-1}r + H(m)s^{-1} \mod n)$$

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: $|k_i| = |xt_i - u_i| = |xs_i^{-1}r_i + H(m_i)s_i^{-1}| < n/2^{l_i}$

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 **B** and reduce it:

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

Exploitation

Basic attack [2]

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

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

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

Exploitation

Basic attack [2]

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

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

- Construct a target $\mathbf{u} = (2^{l_1}u_1, \dots, 2^{l_d}u_d, 0)$
- Solve CVP($\mathbf{B}, \mathbf{u}$). The closest lattice point is often: $\mathbf{v} = (2^{l_1}t_1x, \dots, 2^{l_d}t_dx, x)$

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 **B** and reduce it:

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

- Construct a target $\mathbf{u} = (2^{l_1}u_1, \dots, 2^{l_d}u_d, 0)$
- Solve CVP($\mathbf{B}, \mathbf{u}$). The closest lattice point is often: $\mathbf{v} = (2^{l_1}t_1x, \dots, 2^{l_d}t_dx, x)$
- 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 $\beta \in \{15, 20, \dots, 55\}$

Dataset	*base* (µs)	*iter_time* (µs)	*sdev* (µs)
`sim`	0	1	0
`sw`	453.4	12.7	17.2
`tpm`	27047.3	236.1	211.3
`card`	43578.4	371.5	451.3

Analysis

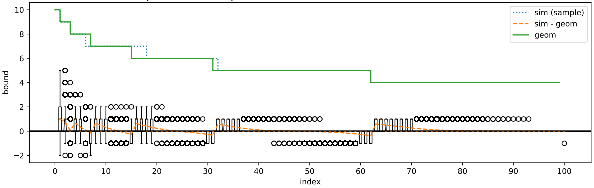
Bounds l_i

- 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

Analysis

Bounds l_i

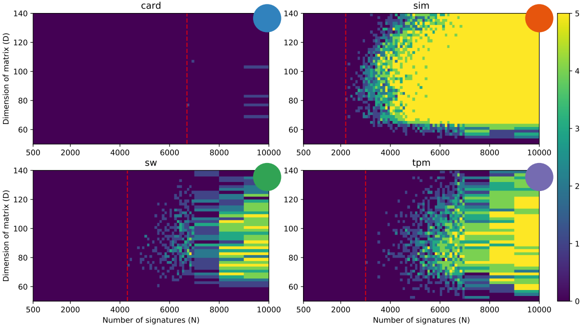
- How to assign bound l_i for the i-th fastest signature?
  - Constant ($l_i = c$ for $c \in \{1, 2, 3, 4\}$) based on d
  - Geometric based on N (✹ **new**)



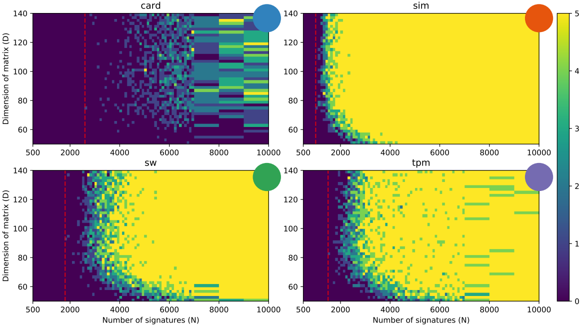
Analysis

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 $c = 3$

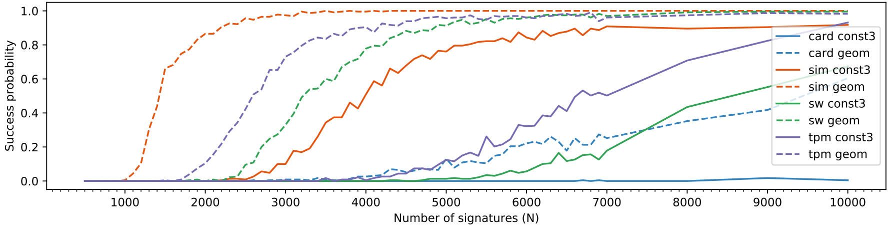


geometric

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**)

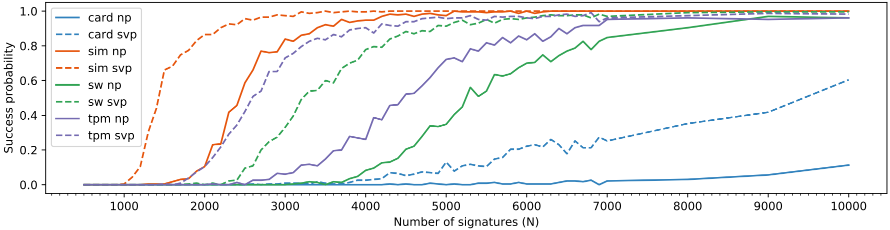


Analysis

CVP vs SVP

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

$$\mathbf{C} = \begin{pmatrix} \mathbf{B} & 0 \\ \mathbf{u} & n \end{pmatrix}$$

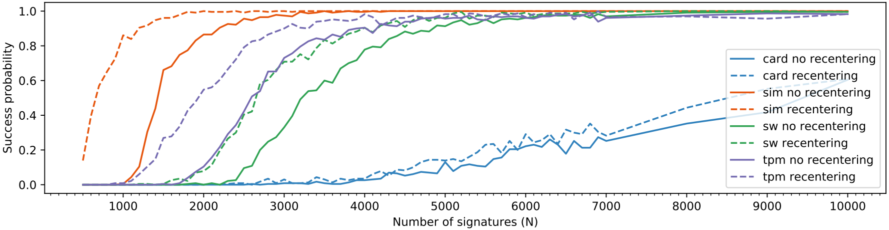


Analysis

Recentering

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

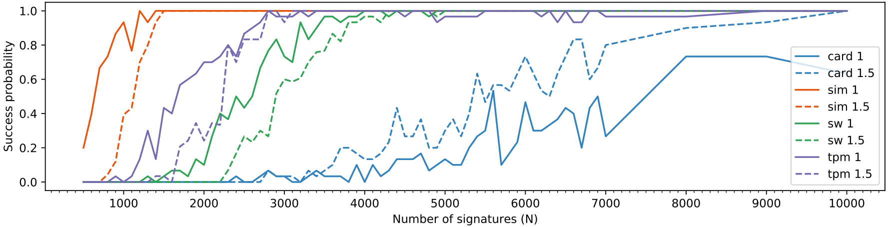
$$\left|k_i - n/2^{l_i+1}\right| < n/2^{l_i+1}$$



Analysis

Random subsets

- 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

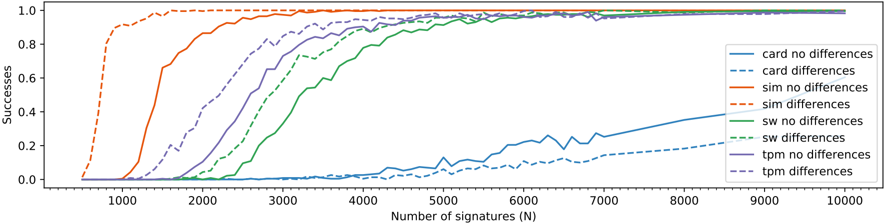


Analysis

Nonce differences

- 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

$$|k_i - k_j| < n/2^{\min(l_i, l_j)}$$



Analysis

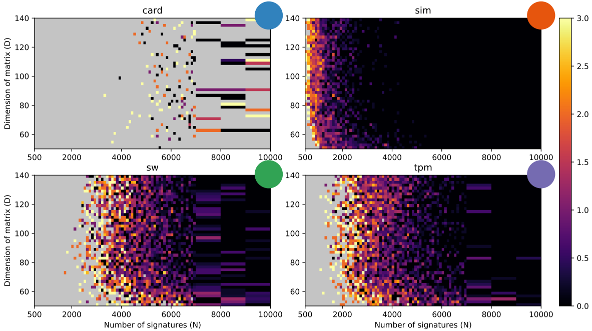
u-bitflips

- 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 $\mathbf{u}$
- (✹ **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**)

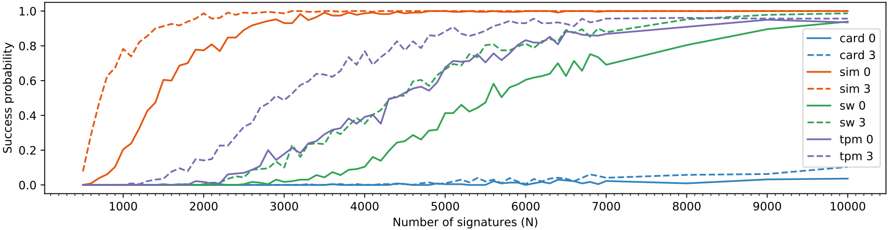


errors fixed for success

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 **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**