Kerckhoff's Principle in Practice: Addressing Security by Obscurity in Secure Hardware

Jan Jancar[1*] Vojtech Suchanek[1*] Jan Kvapil[1]
Petr Svenda[1] Vladimir Sedlacek[2] Łukasz Chmielewski[1]
+ Thanks to Thomas Roche.

CROCS [1]
RUTGERS UNIVERSITY [2]

pyecsca.org

Secure hardware

- Smartcards, TPMs, HSMs, U2F tokens, secure elements, …
  - All share hardware/firmware components

Secure hardware

- Smartcards, TPMs, HSMs, U2F tokens, secure elements, …
  - All share hardware/firmware components
- Wide-spread use
  - Credit cards, eIDs, SIM cards, mobile phones, …

Secure hardware

- Smartcards, TPMs, HSMs, U2F tokens, secure elements, …
  - All share hardware/firmware components
- Wide-spread use
  - Credit cards, eIDs, SIM cards, mobile phones, …
- JavaCard platform
  - Programmable smart card
  - Subset of Java + access to hardware-accelerated crypto
  - Underlying platform for many devices

Secure hardware

- Smartcards, TPMs, HSMs, U2F tokens, secure elements, …
  - All share hardware/firmware components
- Wide-spread use
  - Credit cards, eIDs, SIM cards, mobile phones, …
- JavaCard platform
  - Programmable smart card
  - Subset of Java + access to hardware-accelerated crypto
  - Underlying platform for many devices
- **Very secretive**
  - NDAs, hard to obtain samples, no docs

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …
- Cryptographic theory (primitive & protocol)
  - Open design
  - Public review, competitions
  - Proofs (machine-checked!)

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …
- Cryptographic theory (primitive & protocol)
  - Open design
  - Public review, competitions
  - Proofs (machine-checked!)
- Open-source cryptographic implementations
  - Public review
  - Audits
  - Formal verification

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …
- Cryptographic theory (primitive & protocol)
  - Open design
  - Public review, competitions
  - Proofs (machine-checked!)
- Open-source cryptographic implementations
  - Public review
  - Audits
  - Formal verification
- Secure hardware
  - Be a big fish, sign NDAs, do your own analysis
  - Trust the vendor
  - Security certifications

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …

Cryptographic theory (primitive & protocol) • Open design • Public review, competitions • Proofs (machine-checked!)	**Rarely fails**
Open-source cryptographic implementations • Public review • Audits • Formal verification	**Vulnerabilities**
Secure hardware • Be a big fish, sign NDAs, do your own analysis • Trust the vendor • Security certifications	**Here be dragons**

Verifying security

- How can **you** verify security of secure hardware?
  - It's in the name, duh …

- Cryptographic theory (primitive & protocol)
  - Open design
  - Public review, competitions
  - Proofs (machine-checked!)
- Open-source cryptographic implementations
  - Public review
  - Audits
  - Formal verification
- Secure hardware
  - Be a big fish, sign NDAs, do your own analysis
  - Trust the vendor
  - Security certifications

Security certifications

- How to gain confidence in security?
  - **Common Criteria** (+EUCC), **FIPS 140**, **EMVCo**

Security certifications

- How to gain confidence in security?
  - **Common Criteria** (+EUCC), **FIPS 140**, **EMVCo**
- Mandated use of certified products
  - Government, healthcare, eID, …

Security certifications

- How to gain confidence in security?
  - **Common Criteria** (+EUCC), **FIPS 140**, **EMVCo**
- Mandated use of certified products
  - Government, healthcare, eID, …
- Certification that product adheres to some *security target*
  - Independent lab evaluates

Security certifications

- How to gain confidence in security?
  - **Common Criteria** (+EUCC), **FIPS 140**, **EMVCo**
- Mandated use of certified products
  - Government, healthcare, eID, …
- Certification that product adheres to some *security target*
  - Independent lab evaluates
- **Certifications reward secrecy**
  - *Attack Potential*
  - Non-public information required -> higher score
  - Samples required -> higher score

Vulnerabilities

- **ROCA** (2017)
  - Factorable RSA keys

Vulnerabilities

- **ROCA** (2017)
  - Factorable RSA keys
- **Minerva** (2019), **TPM-Fail** (2019), **TPMScan** (2024)
  - Key recovery from (timing) leakage in ECDSA signing

Vulnerabilities

- **ROCA** (2017)
  - Factorable RSA keys
- **Minerva** (2019), **TPM-Fail** (2019), **TPMScan** (2024)
  - Key recovery from (timing) leakage in ECDSA signing
- **Side journey to Titan** (2021)
  - Key recovery from EM leakage in ECDSA signing

Vulnerabilities

- **ROCA** (2017)
  - Factorable RSA keys
- **Minerva** (2019), **TPM-Fail** (2019), **TPMScan** (2024)
  - Key recovery from (timing) leakage in ECDSA signing
- **Side journey to Titan** (2021)
  - Key recovery from EM leakage in ECDSA signing
- **EUCLEAK** (2024)
  - Key recovery from EM leakage in ECDSA signing

Vulnerabilities

- **ROCA** (2017)
  - Factorable RSA keys
- **Minerva** (2019), **TPM-Fail** (2019), **TPMScan** (2024)
  - Key recovery from (timing) leakage in ECDSA signing
- **Side journey to Titan** (2021)
  - Key recovery from EM leakage in ECDSA signing
- **EUCLEAK** (2024)
  - Key recovery from EM leakage in ECDSA signing
- All affected certified devices
- All very hard to patch in practice
- Most required recovering non-public information

Side-channel attacks

- Simple Power Analysis
- Differential Power Analysis
- Correlation Power Analysis
- Mutual Information Analysis
- Refined Power Analysis, Zero-value Point Attack, Exceptional Procedure Attack
- Template attacks
- Leakage assessment
- Doubling attack, Collision attacks
- …

Side-channel attacks

is assumed to be known

[1] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard & Justine Wild: Horizontal Collision Correlation Attack on Elliptic Curves

Side-channel attacks

assumes a white-box evaluator

is assumed to be known

[2] Oscar Reparaz, Josep Balasch & Ingrid Verbauwhed: Dude, is my code constant time?

Side-channel attacks

assumes a white-box evaluator

assume it is a binary exp

is assumed to be known

[3] Johann Heyszl: Impact of Localized Electromagnetic Field Measurements on Implementations of Asymmetric Cryptography

Side-channel attacks

assumes a white-box evaluator

assume it is a binary exp

is assumed to be known

only works when using the downward routine.

[4] Pierre-Alain Fouque & Frederic Valette: The Doubling Attack – Why Upwards Is Better than Downwards

Side-channel attacks

assumes a white-box evaluator

assume it is a binary exp

is applied to three different atomic formulae on elliptic curves,

is assumed to be known

only works when using the downward routine.

[1] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard & Justine Wild: Horizontal Collision Correlation Attack on Elliptic Curves

Side-channel attacks

attack on the Montgomery-López-Dahab ladder algorithm

assumes a white-box evaluator

assume it is a binary exp

is applied to three different atomic formulae on elliptic curves,

is assumed to be known

only works when using the downward routine.

[5] Bo-Yeon Sim & Dong-Guk Han: Key Bit-Dependent Attack on Protected PKC Using a Single Trace

Side-channel attacks

attack on the Montgomery-López-Dahab ladder algorithm

assumes a white-box evaluator

assume it is a binary exp

knowledge of the ECSM and the elliptic

is applied to three different atomic formulae on elliptic curves,

is assumed to be known

only works when using the downward routine.

[6] Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica & David Naccache: A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards

Side-channel attacks

attack on the Montgomery-López-Dahab ladder algorithm

assumes a white-box evaluator

assume it is a binary exp

is applied to three different atomic formulae on elliptic curves,

knowledge of the ECSM and the elliptic

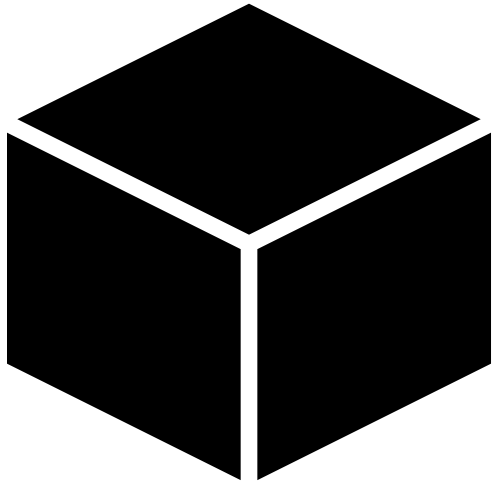
is assumed to be known

only works when using the downward routine.

Lots of assumptions!

Reality and the gap

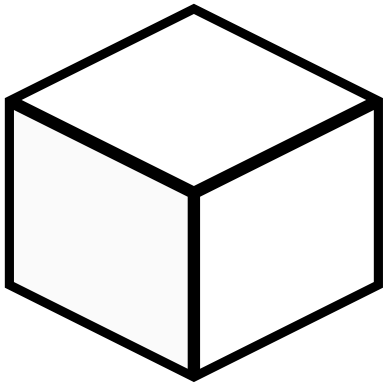
(Kool and the gang)



Reality and the gap

(Kool and the gang)

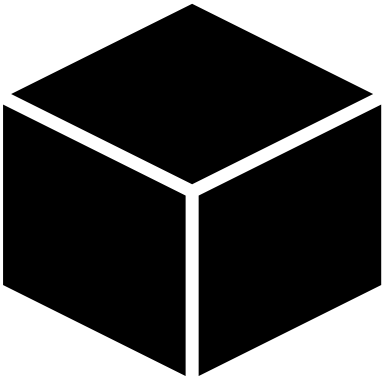
Reality and the gap (Kool and the gang)



Attacks

???

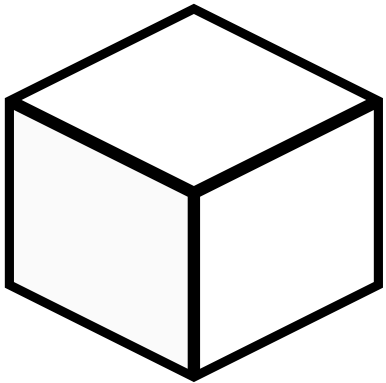
Security by obscurity



Targets

Reality and the gap

(Kool and the gang)

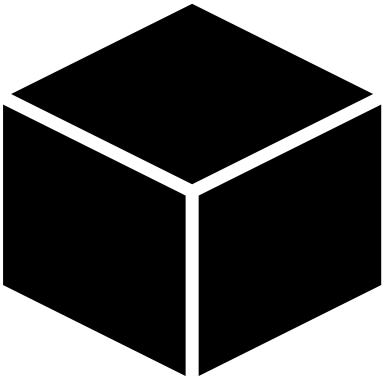


Attacks

???

Security by obscurity

How large is the gap?
How hard is it to cross?



Targets

Elliptic curve cryptography

- Many implementation possibilities …

Elliptic curve cryptography

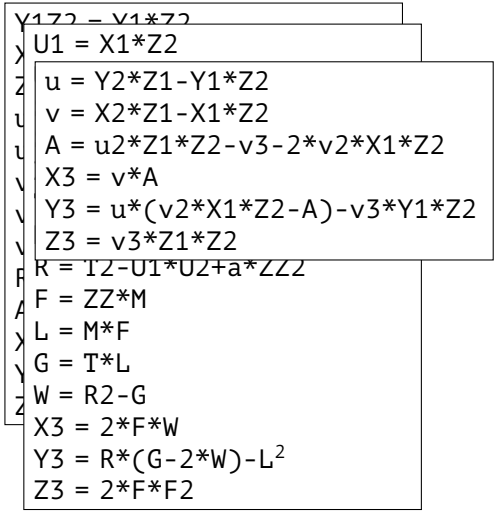
- Many implementation possibilities …
  - Curve model
    - $y^2 \equiv x^3 + ax + b$
    - $x^2 + y^2 \equiv c^2(1 + dx^2y^2)$
    - $ax^2 + y^2 \equiv 1 + dx^2y^2$
    - $by^2 \equiv x^3 + ax^2 + x$

Elliptic curve cryptography

- Many implementation possibilities …
  - Curve model
  - Coordinates
    - (X, Y)
    - (X, Y, Z)
    - (X, Y, Z, ZZ) …

Elliptic curve cryptography

- Many implementation possibilities …
  - Curve model
  - Coordinates
  - Addition formulas



Elliptic curve cryptography

- Many implementation possibilities …
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
    - *LTR*, *RTL*, *Window*, *NAF*, *Comb* …

Fixed-window scalar multiplier

function Window($G, k = (k_l, \ldots, k_0)_2$)
  $Table = [0G, 1G, \ldots, (2^w - 1)G]$
  $\hat{k}$ = recode k to w-bit windows
  $T = \mathcal{O}$
  for $i = 1$ **to** $|\hat{k}|$ **do**
    $T = 2^w T$
    $T = T + Table[\hat{k}_i]$
  return T

Elliptic curve cryptography

- Many implementation possibilities …
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
    - Multiplication: *Toom-Cook, Karatsuba,* …
    - Squaring: *Toom-Cook, Karatsuba,* …
    - Reduction: *Barret, Montgomery,* …
    - Inversion: *GCD, Euler*

Elliptic curve cryptography

- Many implementation possibilities …
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
  - Side-channel countermeasures

Elliptic curve cryptography

- Many implementation possibilities … -> **100k - 100m**
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
  - Side-channel countermeasures

Elliptic curve cryptography

- Many implementation possibilities … -> **100k - 100m**
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
  - Side-channel countermeasures
- …that are used in practice
  - Analyzed 18 open-source crypto libraries

Elliptic curve cryptography

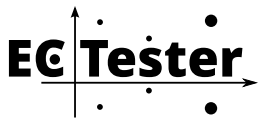
- Many implementation possibilities … -> **100k - 100m**
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
  - Side-channel countermeasures
- …that are used in practice
  - Analyzed 18 open-source crypto libraries
  - Mixed curve models
  - Jungle of scalar multipliers (Comb, fixed window, wNAF, GLV, …)
  - Various coordinate systems
  - Standard and custom addition formulas (more than 100)

Reverse engineering

- How hard is it to cross the gap?

pyecsca

Side-channels



Lying

Reverse engineering via side-channel attacks

- **Idea**: Use side-channel attacks and turn them around
  - ~~Assume knowledge of the impl. and target the key~~
  - Assume knowledge of the key and target the impl.
- ≠ Behavior of different implementations differs under these attacks

Reverse engineering via side-channel attacks

- **Idea**: Use side-channel attacks and turn them around
  - ~~Assume knowledge of the impl. and target the key~~
  - Assume knowledge of the key and target the impl.
- ≠ Behavior of different implementations differs under these attacks
- Concretely "*special-point-based*" attacks: **RPA, ZVP, EPA**
  - Can recognize when a special point appears in scalar multiplication
  - Introduce zeros adaptively
  - Zeros detectable via side-channel

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples
  - Simulate behavior of implementations under the oracle (attack)

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples
  - Simulate behavior of implementations under the oracle (attack)
  - Build a decision table with the answers

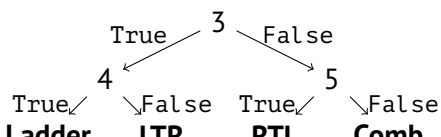
r =	2	3	4	5
LTR	True	True	False	False
RTL	True	False	True	True
Comb	True	False	True	False
Ladder	True	True	True	False
…	…			

RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples
  - Simulate behavior of implementations under the oracle (attack)
  - Build a decision table with the answers
  - Build a decision tree, recursively picking the best split

r =	2	3	4	5
LTR	True	True	False	False
RTL	True	False	True	True
Comb	True	False	True	False
Ladder	True	True	True	False
…		…		

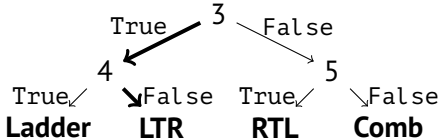


RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples
  - Simulate behavior of implementations under the oracle (attack)
  - Build a decision table with the answers
  - Build a decision tree, recursively picking the best split
  - Walk the tree

r =	2	3	4	5
LTR	True	True	False	False
RTL	True	False	True	True
Comb	True	False	True	False
Ladder	True	True	True	False
…		…		



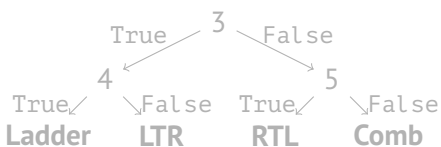
RPA-RE

Target: Static ECDH, $[k]P$ for some known k, can measure power/EM

- **RPA** oracle: Choose r, is $[r]P$ computed while doing $[k]P$?
- **RPA-RE**: Use **RPA** oracle
  - Diff. scalar multiplier -> diff. sequence of multiples
  - Simulate behavior of implementations under the oracle (attack)
  - Build a decision table with the answers
  - Build a decision tree, recursively picking the best split
  - Walk the tree

Works,
but what about countermeasures?

r =	2	3	4	5
LTR	True	True	False	False
RTL	True	False	True	True
Comb	True	False	True	False
Ladder	True	True	True	False
…	…			



Scalar randomization countermeasures

Group scalar randomization

function Mult(P, k)
  $r \xleftarrow{\$} \{0, 1, \ldots, 2^b\}$
  return $[k + rn]P$

Additive splitting

function Mult(P, k)
  $r \xleftarrow{\$} \mathbb{Z}_n^*$
  return $[k - r]P + [r]G$

Euclidean splitting

function Mult(P, k)
  $r \xleftarrow{\$} \{0, 1, \ldots, 2^{\lfloor \log_2(n)/2 \rfloor}\}$
  $S \leftarrow [r]P$
  $k_1 \leftarrow k \mod r$
  $k_2 \leftarrow \left\lfloor \frac{k}{r} \right\rfloor$
  return $[k_1]P + [k_2]S$

Multiplicative splitting

function Mult(P, k)
  $r \xleftarrow{\$} \{0, 1, \ldots, 2^b\}$
  $S \leftarrow [r]P$
  return $[kr^{-1} \mod n]S$

-> **RPA** will not work

Reverse engineering by lying

- Countermeasures are correct on well-defined curves, what about bad curves?
- JavaCard platform: Allows any short-Weierstrass curve
- **Idea:** Recover the countermeasures based on their (erroneous) behavior

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels + countermeasures

- Select curve with order $3q$

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

- Select curve with order $3q$
- Lie to the target, claim order $n = q$

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

- Select curve with order $3q$
- Lie to the target, claim order $n = q$
- Supply point of order $3q$

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

- Select curve with order $3q$
- Lie to the target, claim order $n = q$
- Supply point of order $3q$
- …
- Profit

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

- Select curve with order $3q$
- Lie to the target, claim order $n = q$
- Supply point of order $3q$
- …
- Profit

$[k]P \quad [k + n]P \quad [k + 2n]P$

Method $3q$

Target: Static ECDH, $[k]P$ for some known k, no side-channels
+ countermeasures

- Select curve with order $3q$
- Lie to the target, claim order $n = q$
- Supply point of order $3q$
- …
- Profit

	$[k]P$	$[k + n]P$	$[k + 2n]P$
Group scalar	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
Additive	$\frac{1}{2}$	$\frac{1}{2}$	0
Multiplicative	$\frac{5}{9}$	$\frac{2}{9}$	$\frac{2}{9}$
Euclidean	1	0	0

Results

Card	Target	Method $3q$			Countermeasure	r bits
NXP 1	Derive	0.34	0.33	0.32	Group scalar	✗
	Sign	0.31	0.31	0.38	Group scalar	160
	Keygen	0.32	0.33	0.35	Group scalar	160
NXP 4	Derive	0.56	0.22	0.22	Multiplicative	64
	Sign	0.54	0.23	0.23	Multiplicative	?
Infineon 2	Derive	0.31	0.36	0.33	Group scalar	✗
	Sign	0.32	0.32	0.36	Group scalar	64
	Keygen	0.32	0.32	0.36	Group scalar	32
⋯	Many more results in the paper ⋯					

Conclusions

- Security by obscurity does not work well

Conclusions

- Security by obscurity does not work well
- Obscurity is hard to measure
  - Side-channel attacks can be (and are) quantified
  - How do you quantify obscurity and reverse engineering effort?

Conclusions

- Security by obscurity does not work well
- Obscurity is hard to measure
  - Side-channel attacks can be (and are) quantified
  - How do you quantify obscurity and reverse engineering effort?
- Security certifications should update to reflect this
  - Secrecy of design should not be rewarded, it is easy to bypass
  - **sec-certs.org** project: Bringing transparency to security certifications

Conclusions

- Security by obscurity does not work well
- Obscurity is hard to measure
  - Side-channel attacks can be (and are) quantified
  - How do you quantify obscurity and reverse engineering effort?
- Security certifications should update to reflect this
  - Secrecy of design should not be rewarded, it is easy to bypass
  - **sec-certs.org** project: Bringing transparency to security certifications
- Consider transparent alternatives
  - opentitan
  - tropicsquare

Kerckhoff's Principle in Practice: Addressing Security by Obscurity in Secure Hardware

Questions?

pyecsca.org
jan@neuromancer.sk

References

- Jan Jancar, Vojtech Suchanek, Petr Svenda, Vladimir Sedlacek & Łukasz Chmielewski;
  pyecsca: Reverse-engineering black-box elliptic curve cryptography via side-channel analysis
- Vojtech Suchanek, Jan Jancar, Jan Kvapil, Petr Svenda, & Łukasz Chmielewski;
  ECTester: Reverse-engineering side-channel countermeasures of ECC implementations

References (cont.)

Attack assumptions

1. Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard & Justine Wild: Horizontal Collision Correlation Attack on Elliptic Curves
2. Oscar Reparaz, Josep Balasch & Ingrid Verbauwhed: Dude, is my code constant time?
3. Johann Heyszl: Impact of Localized Electromagnetic Field Measurements on Implementations of Asymmetric Cryptography
4. Pierre-Alain Fouque & Frederic Valette: The Doubling Attack – Why Upwards Is Better than Downwards
5. Bo-Yeon Sim & Dong-Guk Han: Key Bit-Dependent Attack on Protected PKC Using a Single Trace
6. Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica & David Naccache: A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards

Other

- Icons from **Font Awesome** and **Noun Project**

References (cont.)

Vulnerabilities

- Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec & Vashek Matyas;
  The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli
- Jan Jancar, Vladimir Sedlacek, Petr Svenda & Marek Sys;
  Minerva: The curse of ECDSA nonces
- Daniel Moghimi, Berk Sunar, Thomas Eisenbarth & Nadia Heninger;
  TPM-FAIL: TPM meets Timing and Lattice Attacks
- Petr Svenda, Antonin Dufka, Milan Broz, Roman Lacko, Tomas Jaros, Daniel Zatovic & Josef Pospisil;
  TPMScan: A wide-scale study of security-relevant properties of TPM 2.0 chips
- Thomas Roche, Victor Lomné, Camille Mutschler & Laurent Imbert;
  A Side Journey to Titan
- Thomas Roche;
  EUCLEAK: Side-Channel Attack on the YubiKey 5 Series

References (cont.)

Special-point-based attacks

- Louis Goubin; A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems
- Toru Akishita, Tsuyoshi Takagi; Zero-value point attacks on elliptic curve cryptosystem
- Tetsuya Izu, Tsuyoshi Takagi; Exceptional procedure attack on elliptic curve cryptosystems
- Vladimir Sedlacek, Jesús-Javier Chi-Domínguez, Jan Jancar, Billy Bob Brumley; A formula for disaster: a unified approach to elliptic curve special-point-based attacks