pyecsca
[pietska]

Reverse engineering black-box elliptic curve cryptography via side-channel analysis

Jan Jancar[1], Vojtech Suchanek[1], Petr Svenda[1], Vladimir Sedlacek[2], Łukasz Chmielewski[1]

CROCS [1]
RUTGERS UNIVERSITY [2]

CHES 2024
pyecsca.org

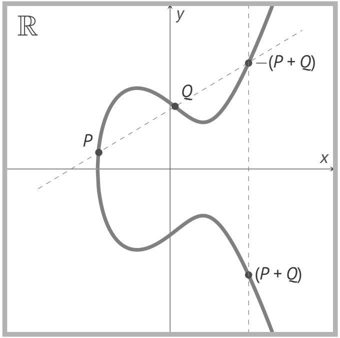
Outline

- Why?
  - Elliptic Curve Cryptography
  - Side-Channel Attacks
- RQ1: Real-world ECC implementations
- RQ2: Space of possible ECC implementations
- RQ3: Reverse-engineering ECC implementations
- Conclusions

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**: $y^2 \equiv x^3 + ax + b$
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$

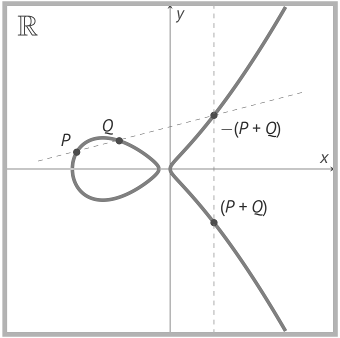


Short Weierstrass

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**: $by^2 \equiv x^3 + ax^2 + x$
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$

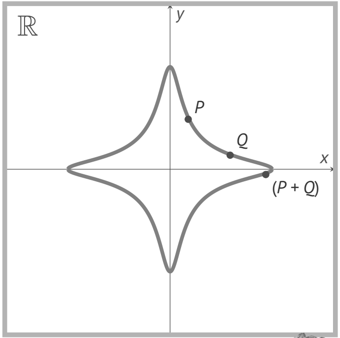


Montgomery

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**: $x^2 + y^2 \equiv c^2(1 + dx^2y^2)$
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$

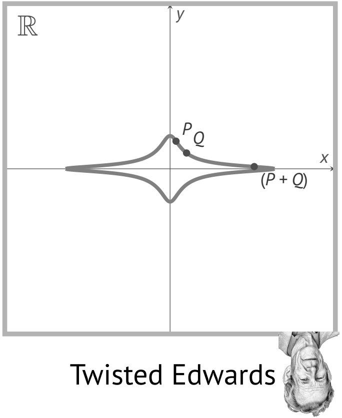


Edwards

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**: $ax^2 + y^2 \equiv 1 + dx^2y^2$
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$



Twisted Edwards

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**:
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$
- **ECDH**: Diffie-Hellman on $E(\mathbb{F}_p)$
  - Scalar multiplication + hash -> Shared secret

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**:
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$
- **ECDH**: Diffie-Hellman on $E(\mathbb{F}_p)$
  - Scalar multiplication + hash -> Shared secret
- **ECDSA**: Digital Signature Algorithm on $E(\mathbb{F}_p)$
  - Random sample + scalar multiplication + hash + mod. arithmetic -> Signature

Why?

Elliptic Curve Cryptography

- **Elliptic Curve**:
  - Points $(x, y) \in E(\mathbb{K})$ form an abelian group
  - Scalar multiplication
    $[n] : E(\mathbb{K}) \to E(\mathbb{K})$
    $P \mapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ times}}$
  - ECDLP: Find x given $[x]G$ and $G \in E(\mathbb{F}_p)$
    Generally hard when $\mathbb{K} = \mathbb{F}_p$
- **ECDH**: Diffie-Hellman on $E(\mathbb{F}_p)$
  - Scalar multiplication + hash -> Shared secret
- **ECDSA**: Digital Signature Algorithm on $E(\mathbb{F}_p)$
  - Random sample + scalar multiplication + hash + mod. arithmetic -> Signature
- **XDH**, **EdDSA**, **...**

Why?

Elliptic Curve Cryptography

- Many implementation possibilities

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
    - $y^2 \equiv x^3 + ax + b$

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
    - (X, Y)
    - (X, Y, Z)
    - $(X, Y, Z, ZZ) \ldots$

Why?

Elliptic Curve Cryptography

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

Why?

Elliptic Curve Cryptography

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

```
Y1Z2 = Y1*Z2
X1Z2 = X1*Z2
Z1Z2 = Z1*Z2
u = Y2*Z1-Y1Z2
uu = u2
v = X2*Z1-X1Z2
vv = v2
vvv = v*vv
R = vv*X1Z2
A = uu*Z1Z2-vvv-2*R
X3 = v*A
Y3 = u*(R-A)-vvv*Y1Z2
Z3 = vvv*Z1Z2
```

Why?

Elliptic Curve Cryptography

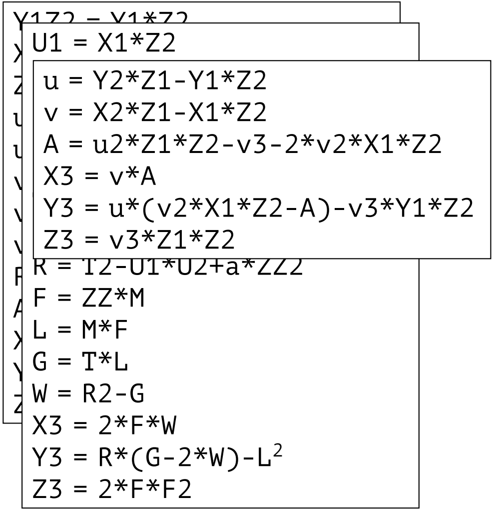
- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas

```
U1 = X1*Z2
U2 = X2*Z1
S1 = Y1*Z2
S2 = Y2*Z1
ZZ = Z1*Z2
T = U1+U2
M = S1+S2
R = T2-U1*U2+a*ZZ2
F = ZZ*M
L = M*F
G = T*L
W = R2-G
X3 = 2*F*W
Y3 = R*(G-2*W)-L^2
Z3 = 2*F*F2
```

Why?

Elliptic Curve Cryptography

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



Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
    - *fixed-base*, *variale-base*, *multi-scalar*

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
    - *fixed-base*, *variale-base*, *multi-scalar*

Algorithm Left-to-right double-and-add

```
function LTR(G, k = (k_l, ..., k_0)_2)
    R = O
    for i = l downto 0 do
        R = dbl(R)
        if k_i = 1 then
            R = add(R, G)
    return R
```

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
    - *fixed-base*, *variale-base*, *multi-scalar*

Algorithm Fixed-window scalar multiplier

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

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
    - Multiplication: *Toom-Cook, Karatsuba, …*

Why?

Elliptic Curve Cryptography

- Many implementation possibilities
  - Curve model
  - Coordinates
  - Addition formulas
  - Scalar multiplier
  - Finite field operations
    - Multiplication: *Toom-Cook, Karatsuba, …*
    - Squaring: *Toom-Cook, Karatsuba, …*

Why?

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, …*

Why?

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*

Why?

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
- ...

Why?

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
- ...

Why?

Assumptions

Why?

Assumptions

$\mathbb{F}_p$ with $p \neq \{2, 3\}$. The algorithm used for the hardware modular multiplication is assumed to be known to the attacker. Moreover, to simplify the attack

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

Why?

Assumptions

is assumed to be known

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

Why?

Assumptions

is assumed to be known

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

Why?

Assumptions

input to s (“fix class”). (This assumes a white-box evaluator that has access to implementation internals.)

is assumed to be known

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

Why?

Assumptions

assumes a white-box evaluator

is assumed to be known

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

Why?

Assumptions

assumes a white-box evaluator

e 6.1 abstractly depicts a side-channel measurement of such an ex-
tion. For the sake of simplicity, I assume it is a binary exponen-

is assumed to be known

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

Why?

Assumptions

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

Why?

Assumptions

assumes a white-box evaluator

assume it is a binary exp

is assumed to be known

values may be manipulated when working with points P and $2P$. However this idea only works when using the downward routine.

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

Why?

Assumptions

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

Why?

Assumptions

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

Why?

Assumptions

assumes a white-box evaluator

assume it is a binary exp

a doubling operation from an addition one. This technique, which allows to eventually recover the secret scalar, 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

Why?

Assumptions

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

Why?

Assumptions

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

Why?

Assumptions

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

full knowledge of all algorithms,

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

Why?

Assumptions

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

full knowledge of all algorithms,

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

Why?

Assumptions

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

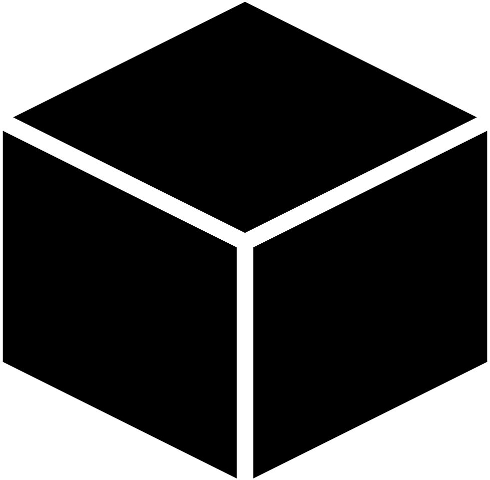
full knowledge of all algorithms,

only works when using the downward routine.

Lots of assumptions you've got there!

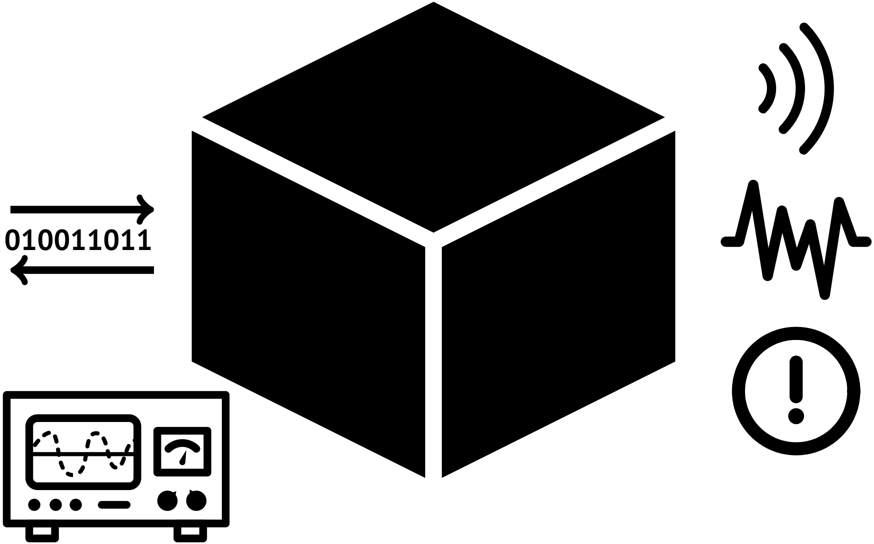
Why?

Reality



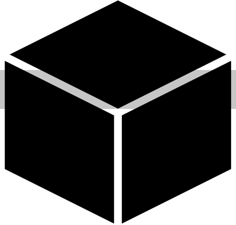
Why?

Reality



Why?

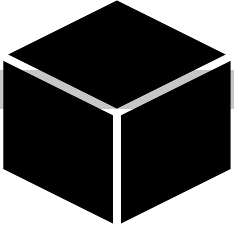
Reality



- Real-world cryptographic hardware is usually a black-box
  - TPMs, HSMs, smartcards, ...
- Why?
  - Security by obscurity
  - Certifications encourage information hiding (CC, JIL-SCA)
- Contrast to cryptographic theory space
  - Kerckhoffs's principle
  - Open design, open discussion

Why?

Reality

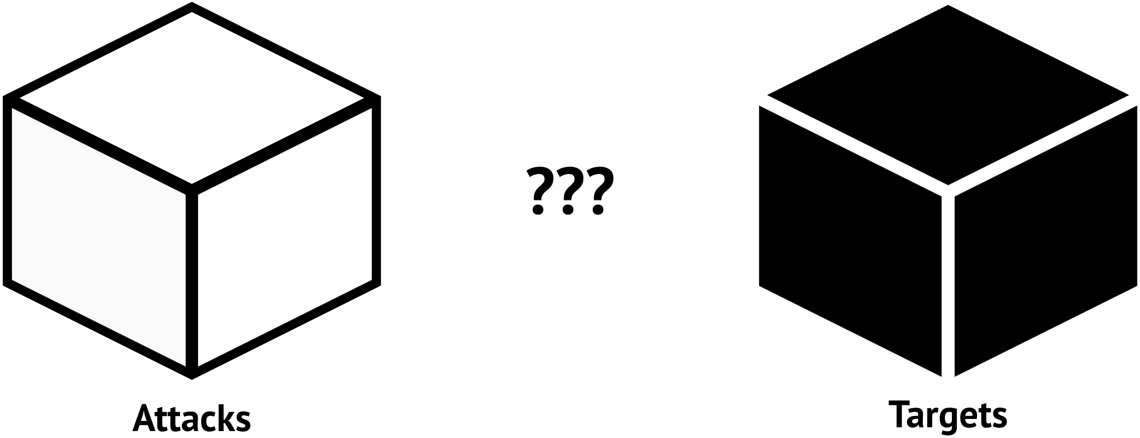


- Real-world cryptographic hardware is usually a black-box
  - TPMs, HSMs, smartcards, …
- Why?
  - Security by obscurity
  - Certifications encourage information hiding (CC, JIL-SCA)
- Contrast to cryptographic theory space
  - Kerckhoffs's principle
  - Open design, open discussion

Auguste Kerckhoffs

Why?

Gap



RQ1:
What implementation choices are used in real-world open-source ECC libraries?

RQ2:
How large is the space of all possible ECC implementations?

RQ3:
Is it possible to automatically reverse-engineer black-box ECC implementations?

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- Analyzed 18 open-source ECC libraries ( )
- *BearSSL, BoringSSL, Botan, BouncyCastle, fastecdsa, Go crypto, Intel IPP cryptography, libgcrypt, LibreSSL, libsecp256k1, libtomcrypt, mbedTLS, micro-ecc, Nettle, NSS, OpenSSL, SunEC,* and *Microsoft SymCrypt*

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- Analyzed 18 open-source ECC libraries (📕)
- *BearSSL, BoringSSL, Botan, BouncyCastle, fastecdsa, Go crypto, Intel IPP cryptography, libgcrypt, LibreSSL, libsecp256k1, libtomcrypt, mbedTLS, micro-ecc, Nettle, NSS, OpenSSL, SunEC,* and *Microsoft SymCrypt*
- Source-code analysis of **ECDH**, **ECDSA**, **X25519**, and **Ed25519**

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- Analyzed 18 open-source ECC libraries ( )
- *BearSSL, BoringSSL, Botan, BouncyCastle, fastecdsa, Go crypto, Intel IPP cryptography, libgcrypt, LibreSSL, libsecp256k1, libtomcrypt, mbedTLS, micro-ecc, Nettle, NSS, OpenSSL, SunEC,* and *Microsoft SymCrypt*
- Source-code analysis of **`ECDH`**, **`ECDSA`**, **`X25519`**, and **`Ed25519`**
- Curve model, Scalar multiplier, Coordinate system, Addition formulas

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- Analyzed 18 open-source ECC libraries ( )
- *BearSSL, BoringSSL, Botan, BouncyCastle, fastecdsa, Go crypto, Intel IPP cryptography, libgcrypt, LibreSSL, libsecp256k1, libtomcrypt, mbedTLS, micro-ecc, Nettle, NSS, OpenSSL, SunEC,* and *Microsoft SymCrypt*
- Source-code analysis of **ECDH**, **ECDSA**, **X25519**, and **Ed25519**
- Curve model, Scalar multiplier, Coordinate system, Addition formulas
- Full report: https://pyecsca.org/libraries.html

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10 📕)
  - e.g. $a = -3$ or special prime arithmetic

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10)
  - e.g. $a = -3$ or special prime arithmetic
- **Curve models**
  - Usually outside = inside
  - Montgomery outside, Twisted-Edwards inside (4)

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10)
  - e.g. $a = -3$ or special prime arithmetic
- **Curve models**
  - Usually outside = inside
  - Montgomery outside, Twisted-Edwards inside (4)
- **Scalar multipliers**
  - *fixed-base* + *variable-base* + *multi-scalar*
  - Comb, fixed-window, wNAF, GLV, …
  - 4 to 7 bit widths

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10 📘)
  - e.g. $a = -3$ or special prime arithmetic
- **Curve models**
  - Usually outside = inside
  - Montgomery outside, Twisted-Edwards inside (4 📘)
- **Scalar multipliers**
  - *fixed-base* + *variable-base* + *multi-scalar*
  - Comb, fixed-window, wNAF, GLV, …
  - 4 to 7 bit widths

- **Coordinate systems**
  - Usually Jacobian, also homogenous or xz

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10 📖)
  - e.g. $a = -3$ or special prime arithmetic
- **Curve models**
  - Usually outside = inside
  - Montgomery outside, Twisted-Edwards inside (4 📖)
- **Scalar multipliers**
  - *fixed-base* + *variable-base* + *multi-scalar*
  - Comb, fixed-window, wNAF, GLV, …
  - 4 to 7 bit widths
- **Coordinate systems**
  - Usually Jacobian, also homogenous or xz
- **Addition formulas**
  - 112 formula implementations
  - 50 “*standard*” (EFD)
  - 23 out-of-scope
  - 39 “*non-standard*”
  - → Expanded standard formulas from ~200 to ~20000

RQ1

What implementation choices are used in real-world open-source ECC libraries?

- **Specific implementations**
  - Curve or architecture-based (10)
  - e.g. $a = -3$ or special prime arithmetic
- **Curve models**
  - Usually outside = inside
  - Twisted-Edwards inside (4)
- **Scalar multipliers**
  - *fixed-base* + *variable-base* + *multi-scalar*
  - Comb, fixed-window, wNAF, GLV, …
  - 4 to 7 bit widths
- **Coordinate systems**
  - Usually Jacobian, also homogenous or xz
- **Addition formulas**
  - 112 formula implementations
  - 50 "*standard*" (EFD)
  - 39 "*non-standard*"
  - Expanded standard formulas from ~200 to ~20000

Wide range of implementation choices in real-world implementations.

↓

Expect also in black-box implementations.

RQ2

How large is the space of all possible ECC implementations?

- We can enumerate:
  - Curve model
  - Coordinate system
  - Addition formulas
  - Scalar multiplier
  - Misc. options

RQ2

How large is the space of all possible ECC implementations?

- We can enumerate:
  - Curve model
  - Coordinate system
  - Addition formulas
  - Scalar multiplier
  - Misc. options
- Total: 139 489

Curve	Coords	#	Total
$\mathcal{E}_{\mathrm{SW}}$	jacobian	17 136	113 502
	jacobian-0	22 848	
	jacobian-3	28 560	
	modified	2 856	
	projective	9 520	
	projective-1	10 710	
	projective-3	16 660	
	w12-0	476	
	xyzz	1 428	
	xyzz-3	2 856	
	xz	452	
$\mathcal{E}_{\mathrm{M}}$	xz	132	132
$\mathcal{E}_{\mathrm{E}}$	inverted	2 856	14 431
	projective	11 424	
	yz	99	
	yzsquared	52	
$\mathcal{E}_{\mathrm{TE}}$	extended	2 856	11 424
	extended-1	5 712	
	inverted	1 428	
	projective	1 428	

RQ2

How large is the space of all possible ECC implementations?

- We can enumerate:
  - Curve model
  - Coordinate system
  - Addition formulas
  - Scalar multiplier
  - Misc. options
- Total: 139 489

Scalar multiplier	**#**
LTR	9 328
RTL	9 328
Coron	1 166
Ladder	407
SimpleLadder	2 332
DiffLadder	328
BinaryNAF	4 664
WindowNAF	18 656
WindowBooth	18 656
Window	9 328
SlidingWindow	18 656
FullPrecomp	18 656
Comb	9 328
BGMW	18 656

RQ2

How large is the space of all possible ECC implementations?

- We can enumerate:
  - Curve model
  - Coordinate system
  - Addition formulas
  - Scalar multiplier
  - Misc. options
- Total: 139 489

Considerable number of implementation configurations: 139 489.

↓

Worth reverse engineering.

RQ3

Is it possible to automatically reverse-engineer black-box ECC implementations?

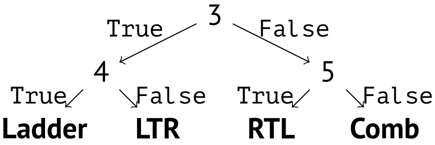
- **Idea:** Use side-channel attacks and turn them around
  - ~~Assume knowledge of the impl. and target the key~~
  - Assume knowledge of the key and target the impl.
- Concretely *"special-point-based"* attacks: **RPA, ZVP, EPA**
- Can recognize when a special point appears in scalar multiplication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

RQ3

Is it possible to automatically reverse-engineer black-box ECC implementations?

- Simulate behavior of implementations under the oracle (attack)
  - **RPA**: Is $[r]P$ computed during $[k]P$ computation by the target?
- Build a decision table with the answers
- Build a decision tree, recursively picking the best split

$\mathcal{I}_{\mathsf{RPA}}$:	$[2^{-1}]P_0$	$[3^{-1}]P_0$	$[4^{-1}]P_0$	$[5^{-1}]P_0$
LTR	True	True	False	False
RTL	True	False	True	True
Comb	True	False	True	False
Ladder	True	True	True	False
…		…		



RQ3

Is it possible to automatically reverse-engineer black-box ECC implementations?

- Simulate behavior of implementations under the oracle (attack)
  - **RPA**: Is $[r]P$ computed during $[k]P$ computation by the target?
- Build a decision table with the answers
- Build a decision tree, recursively picking the best split

Method	Curve	Coordinates	Formulas	Multiplier	Scalar	Input point
RPA-RE	chosen	any	any	**target**	known	chosen
ZVP-RE	chosen	**target**	**target**	known	known	chosen
EPA-RE	chosen	**target**	**target**	known	known	chosen

RQ3

Is it possible to automatically reverse-engineer black-box ECC implementations?

- Implemented in the **pyecsca** toolkit
- It works!

				Expected		**Random**	
Method	**Oracle**	$\lvert\mathcal{C}\rvert$	# 🍃	|🍃|	🍃↕	|🍃|	🍃↕
RPA-RE	binary	34	34	1.0	5.0	1.0	5.0
ZVP-RE	binary	214	74	8.7	5.1	5.0	4.0
ZVP-RE	count	214	134	2.4	4.0	1.3	2.5
ZVP-RE	position	214	196	1.2	2.1	1.1	1.8

RQ3

Is it possible to automatically reverse-engineer black-box ECC implementations?

- Implemented in the **pyecsca** toolkit
- It works!

Method	Oracle	$\lvert\mathcal{C}\rvert$	# 🍃	Expected $\lvert$🍃$\rvert$	Expected 🍃↕	Random $\lvert$🍃$\rvert$	Random 🍃↕
RPA-RE	binary	34	34	1.0	5.0	1.0	5.0
ZVP-RE	binary	214	74	8.7	5.1	5.0	4.0
ZVP-RE	count	214	134	2.4	4.0	1.3	2.5
ZVP-RE	position	214	196	1.2	2.1	1.1	1.8

Yes, it is possible.

. . . and much more

pyecsca

pyecsca.org

Python **E**lliptic **C**urve **S**ide **C**hannel **A**nalysis toolkit

- Can enumerate configurations
- Can simulate computation given any configuration
- Can generate C implementations of ECC for micro-processors
- Can perform power and EM-tracing
- Can process collected traces and visualize them
- Can perform known attacks against ECC
- **Can be used to reverse engineer ECC**

Conclusions

- Documented large variety of implementation choices in 18 open-source ECC libraries
  - **Expect similar variety in black-box devices**
- Explored the space of possible implementation choices of ECC
  - **Considerable number of choices, necessary knowledge for an attack**
- Presented several novel attack-based reverse-engineering methods for ECC
  - **Demonstrated effectivenes on two simulation levels**

- Explore our tutorial:

github.com/J08nY/pyecsca-tutorial-ches2024 pyecsca

[pietska]

Reverse engineering black-box elliptic curve cryptography via side-channel analysis

Jan Jancar, Vojtech Suchanek, Petr Svenda, Vladimir Sedlacek, Łukasz Chmielewski

Questions?

J08nY
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

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

- https://hyperelliptic.org/EFD/
- Icons from **Noun Project** & **Font Awesome**

References (cont.)

ECC attack and countermeasure surveys

- Junfeng Fan, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel & Ingrid Verbauwhede; State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures
- Junfeng Fan & Ingrid Verbauwhede; An updated survey on secure ECC implementations: Attacks, countermeasures and cost
- Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica & David Naccache; A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards
- Rodrigo Abarzúa, Claudio Valencia Cordero & Julio Cesar López-Hernández; Survey on performance and security problems of countermeasures for passive side-channel attacks on ECC

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

References (cont.)

Side-channel-based disassembly

- Jean-Jacques Quisquater & David Samyde;
  Automatic code recognition for smart cards using a Kohonen neural network
- Dennis Vermoen, Marc F. Witteman & Georgi Gaydadjiev;
  Reverse engineering Java Card applets using power analysis
- Thomas Eisenbarth, Christof Paar & Björn Weghenkel;
  Building a side channel based disassembler
- ...and much more (see the paper)

References (cont.)

Side-channel-based reverse engineering

- Christophe Clavier;
  Side channel analysis for reverse engineering (SCARE) – an improved attack against a secret A3/A8 GSM algorithm
- Rémy Daudigny, Hervé Ledig, Frédéric Muller & Frédéric Valette;
  SCARE of the DES
- Manuel San Pedro, Mate Soos & Sylvain Guilley;
  FIRE: Fault injection for reverse engineering
- Frederic Amiel, Benoit Feix & Karine Villegas;
  Power analysis for secret recovering and reverse engineering of public key algorithms
- ...and some more (see the paper)

References (cont.)

Manual reverse engineering

- Thomas Roche, Victor Lomné, Camille Mutschler & Laurent Imbert;
  A Side Journey to Titan
- Thomas Roche;
  EUCLEAK: Side-Channel Attack on the YubiKey 5 Series