Combined Attack on ECC using Points of Low Order

Benedikt Gierlichs
joint work with Junfeng Fan and Frederik Vercauteren

COSIC, Katholieke Universiteit Leuven, Belgium

- ECC: Elliptic curve over finite field
  - A set of points P(x,y) and $\mathcal{O}$ at infinity
- $\mathcal{O}$ required to form an abelian group
- But in crypto you should never see $\mathcal{O}$
- $\mathcal{O}$ is not easy to deal with in implementation
- But it should never occur anyway

So how does /would
your implementation
deal with $\mathcal{O}$?

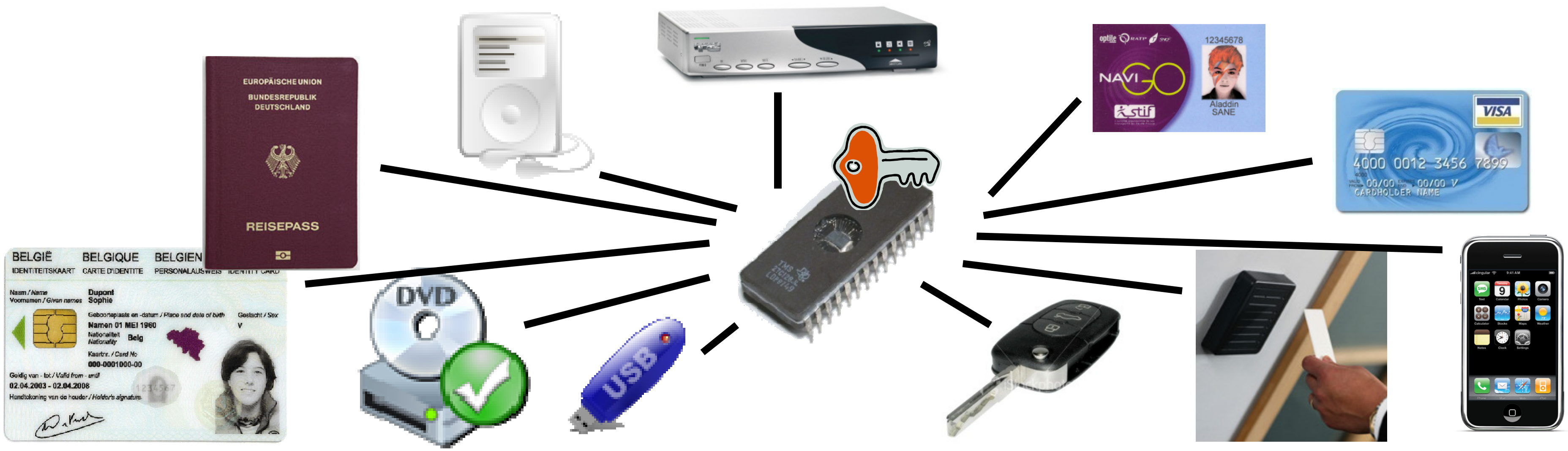
Outline

- Context: embedded security
- Background: elliptic curves and their use in cryptography
- Our attack: principle, toy examples, requirements
- Popular countermeasures for ECC implementations and our attack
- Conclusion

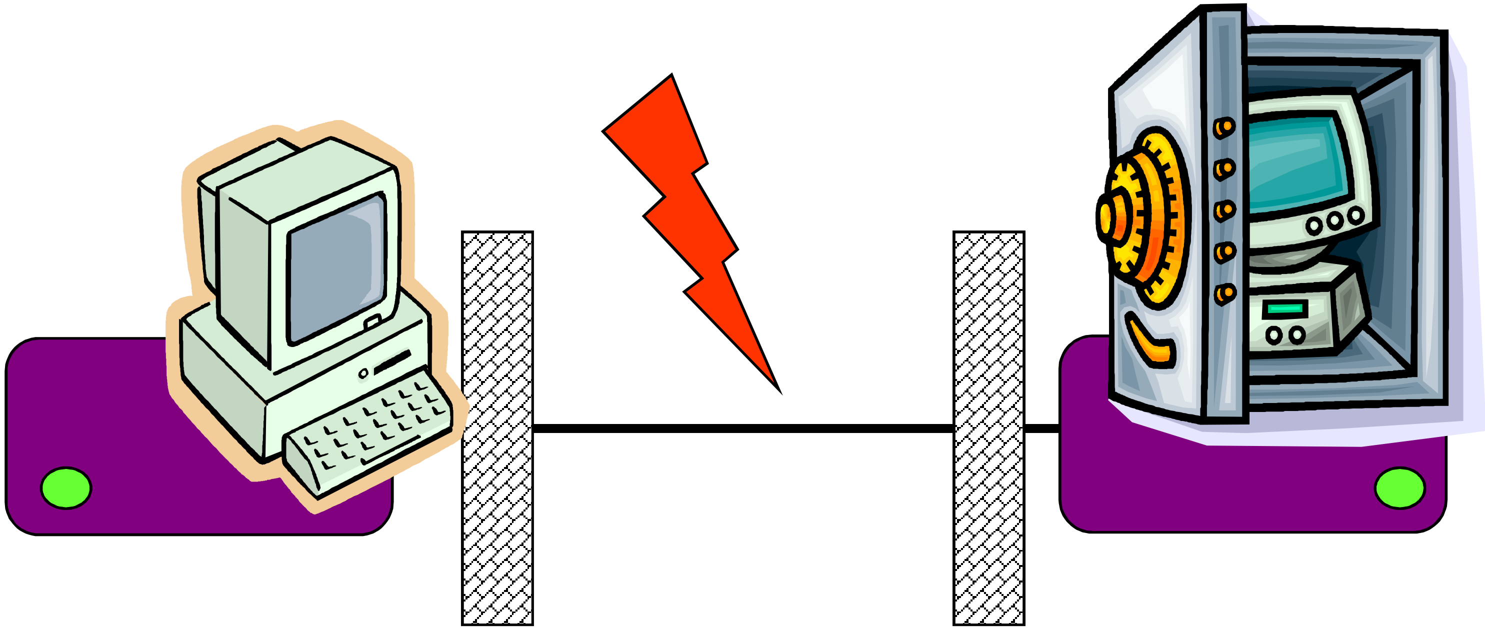
Context

Embedded cryptography

- 98% of processor market are embedded processors
  - In 2008: over 10 billion embedded devices
- Over 100 embedded processors in a single modern luxury car
- More and more applications with security context



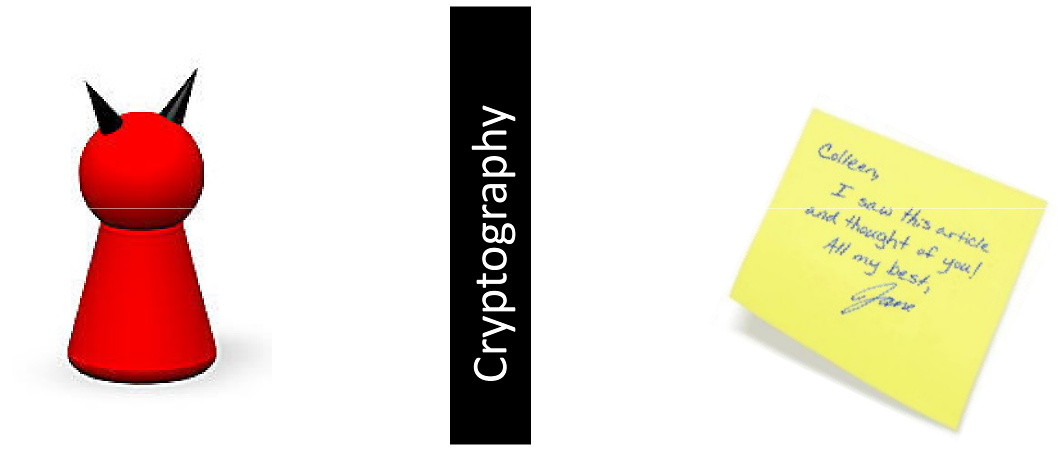
Classical security model (simplified)



- Encryption and cryptographic operations in ***black*** boxes
- Attack on channel ***between*** communicating parties
- Protection by strong mathematic algorithms and protocols

Embedded security

- Can cryptographic functions alone assure security?

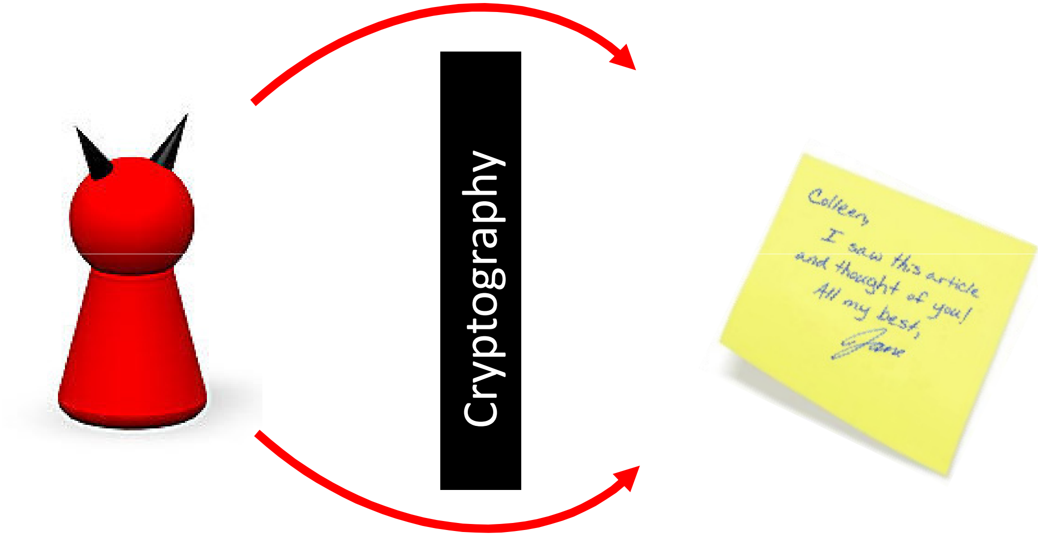


- Cryptographic functions put a barrier between what must be protected and an adversary

The system is as secure as its weakest link

Embedded security

- Can cryptographic functions alone assure security?



- Not if it is easy to bypass them

Embedded security

- Devices are not mathematical functions

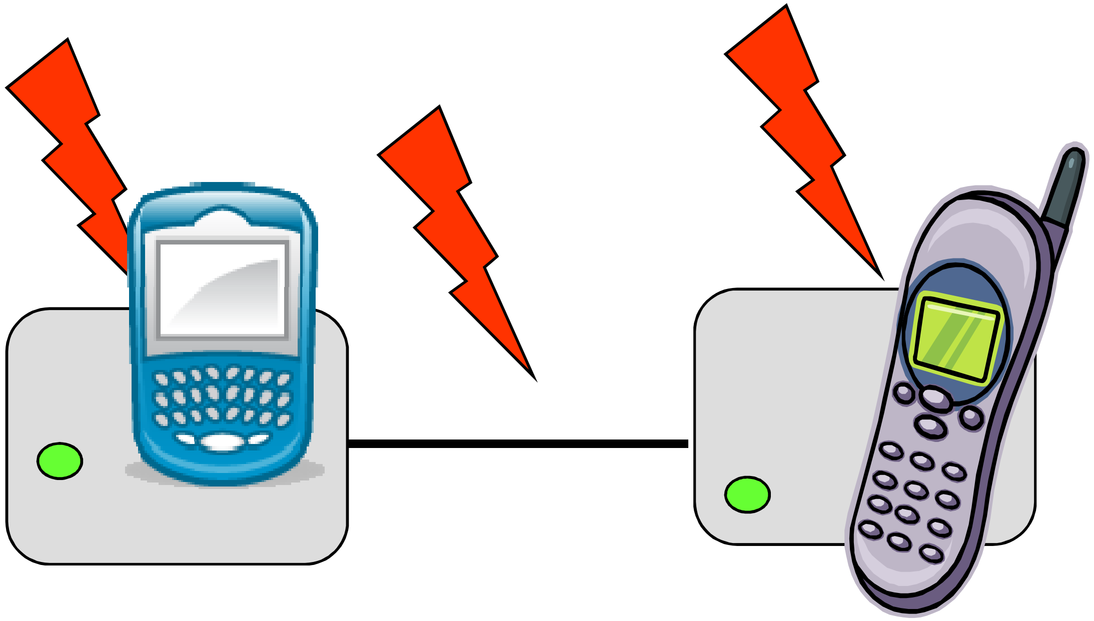
≠

Encryption method

- Subject to physics
  - Physical properties leak information about the secret key
  - Devices react to physical stimulation

- Device in possession of user
  - User can be malicious (or device stolen)
- Device under physical control of adversary
  - No time constraints

New security model (simplified)

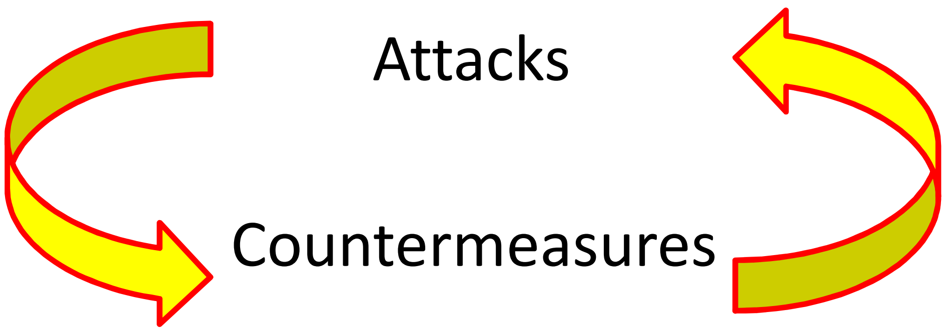


- Attack channel ***and*** endpoints
- Encryption and cryptographic operations in ***gray*** boxes
- Need ***both***:
  - Protection by strong mathematic algorithms and protocols
  - Protection by secure implementation

Why research on attacks?

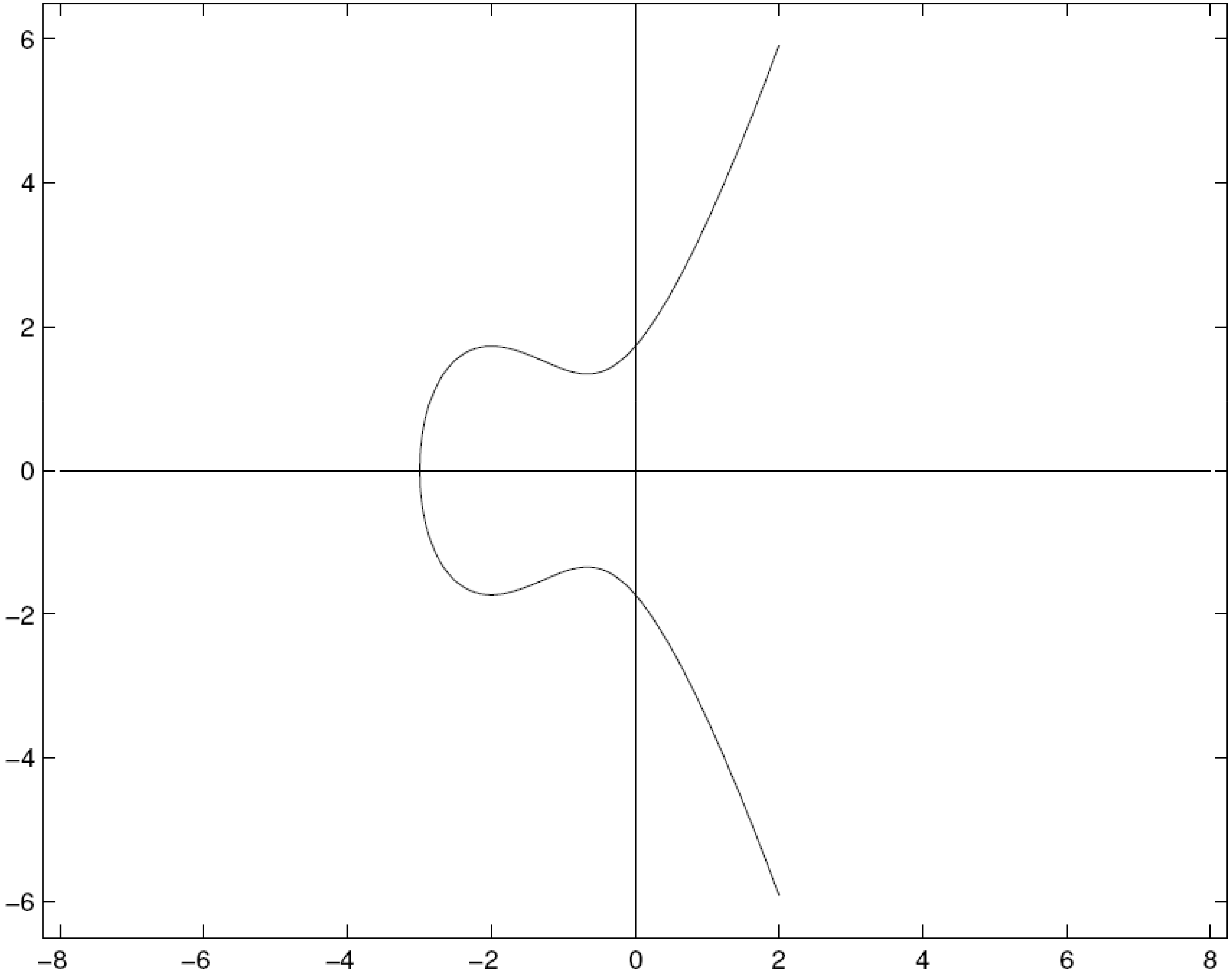
- We have no way to prove that implementation is secure
- Instead, test if implementation resists all known attacks
  - In research and in standardized real-world evaluations
  - Problem similar to symmetric cryptography
- Problem: one can not know all attacks

- Attacks lead to new countermeasures and vice versa

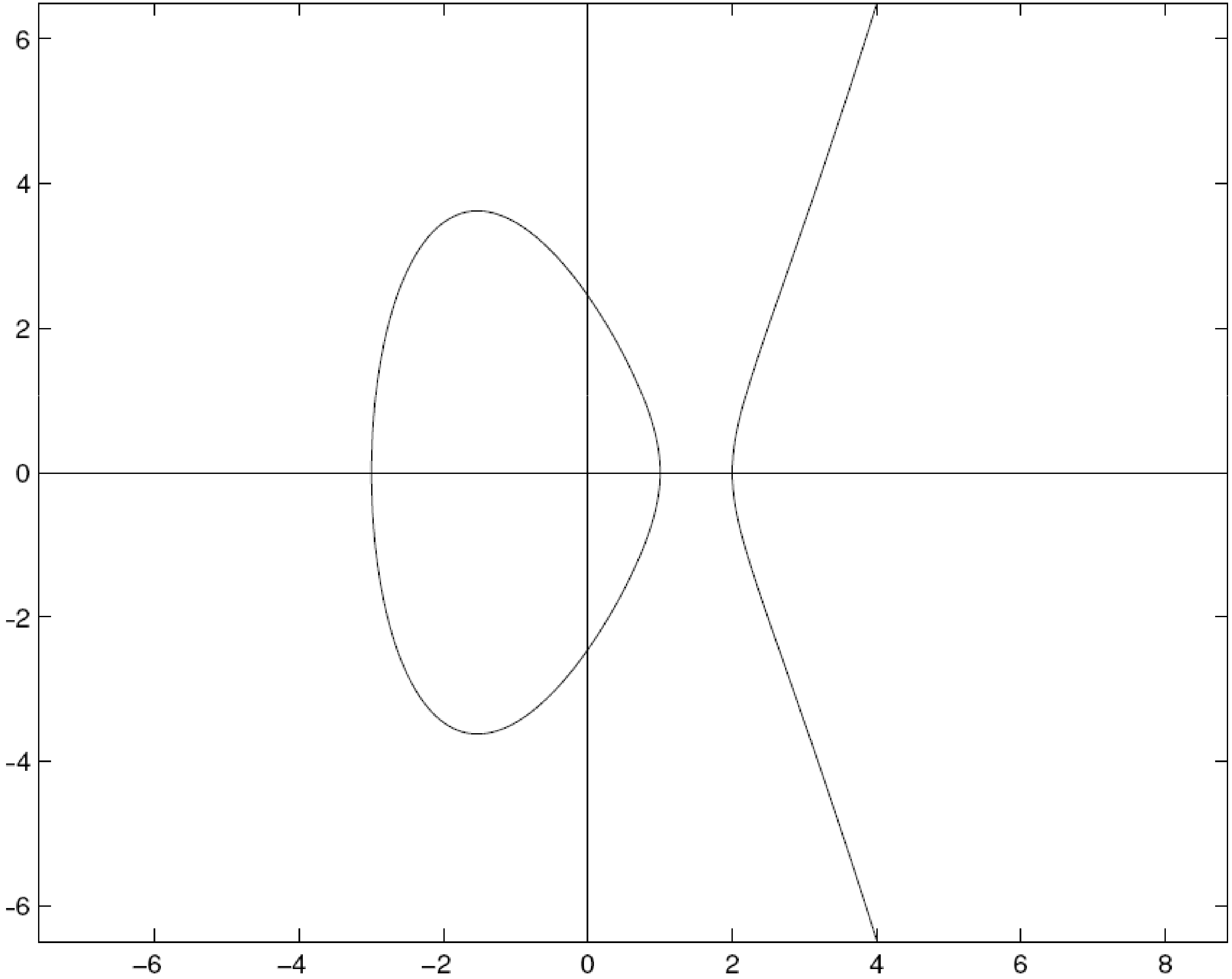


Background

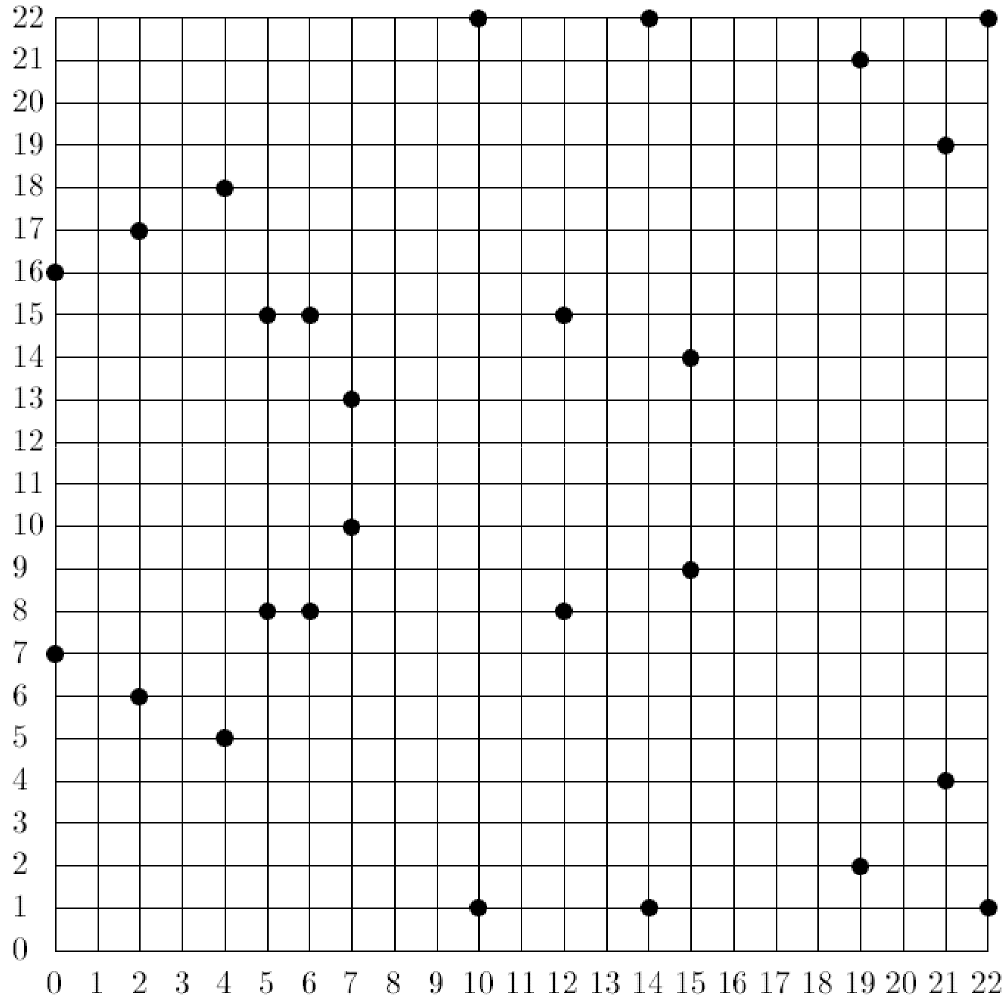
Elliptic curves



Elliptic curves



Elliptic curves over finite fields



The elliptic curve $y^2 = x^3 + x + 3 \bmod 23$

ECC versus RSA

- ECC has several advantages on embedded platforms
- For the same security level
  - Shorter keys
  - Smaller operands
  - Shorter keys: less operations and faster
  - Smaller operands: less memory

Example:

80-bit security
- RSA with 1248-bit keys
- ECC with 160-bit keys

128-bit security
- RSA with 3248-bit keys
- ECC with 256-bit keys

Elliptic curves over finite fields

- E over $\mathbf{F}_p$: $y^2 = x^3 + ax + b \quad a, b \in \mathbf{F}_p \quad 4a^3 + 27b^2 \neq 0$
- $\mathrm{E(K)} := \{(x, y) \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$
- E(K) is abelian group
- $\#\mathrm{E(K)} \simeq \#\mathrm{K}$ with error $\leq 2\sqrt{\#\mathrm{K}}$

- Use in crypto: scalar multiplication k·P
  - EC discrete logarithm problem: given P and k·P, find k
  - Hard because order of P huge on strong curves
  - Implemented as sequence of 'small' operations

P → scalar multiplication → k·P

Scalar multiplication on elliptic curves

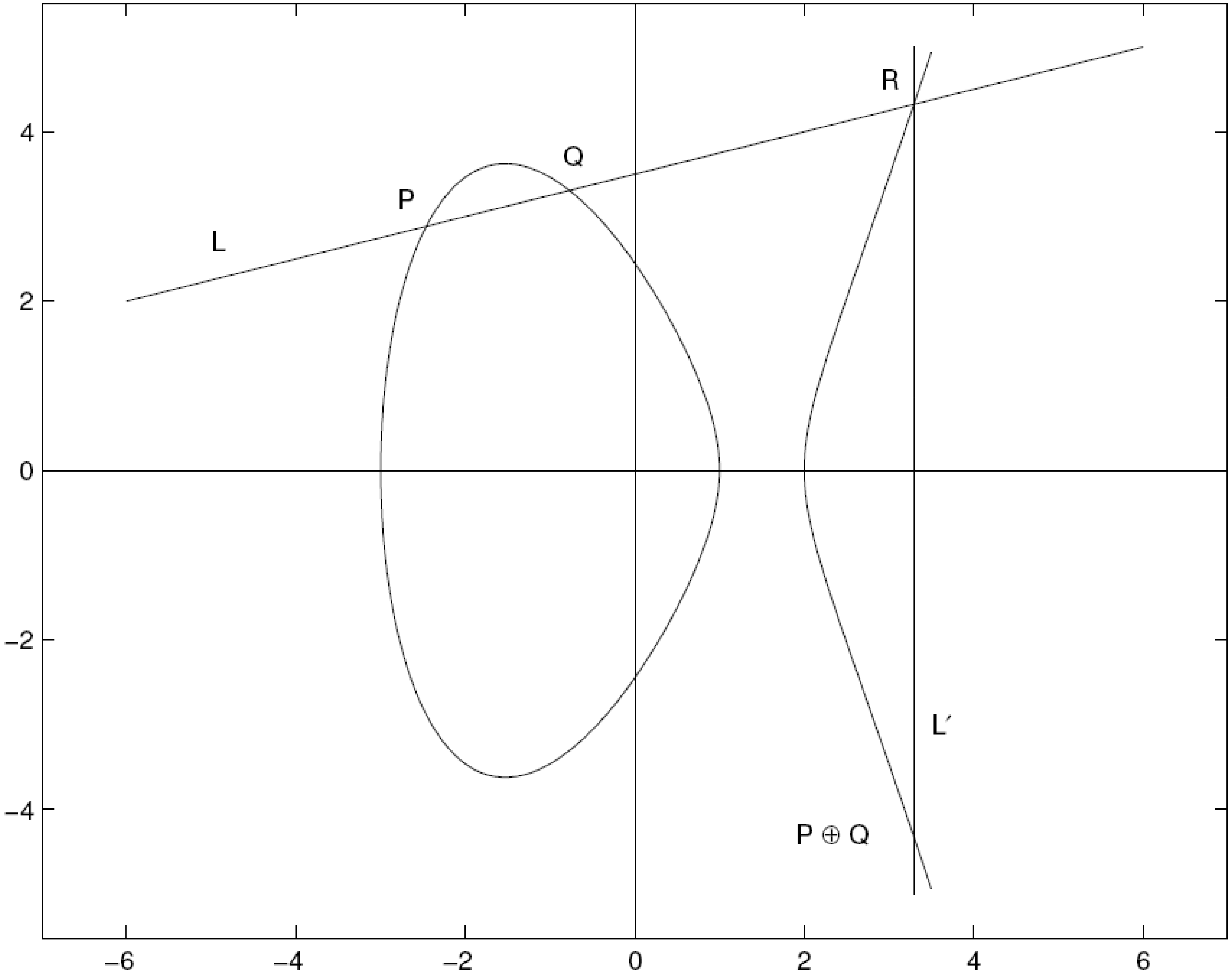
- Given integer k and point P, compute Q = k·P
- Consider k in its binary representation, e.g. k = $10010...110_2$

Algorithm 1: Double and Add Left-to-Right

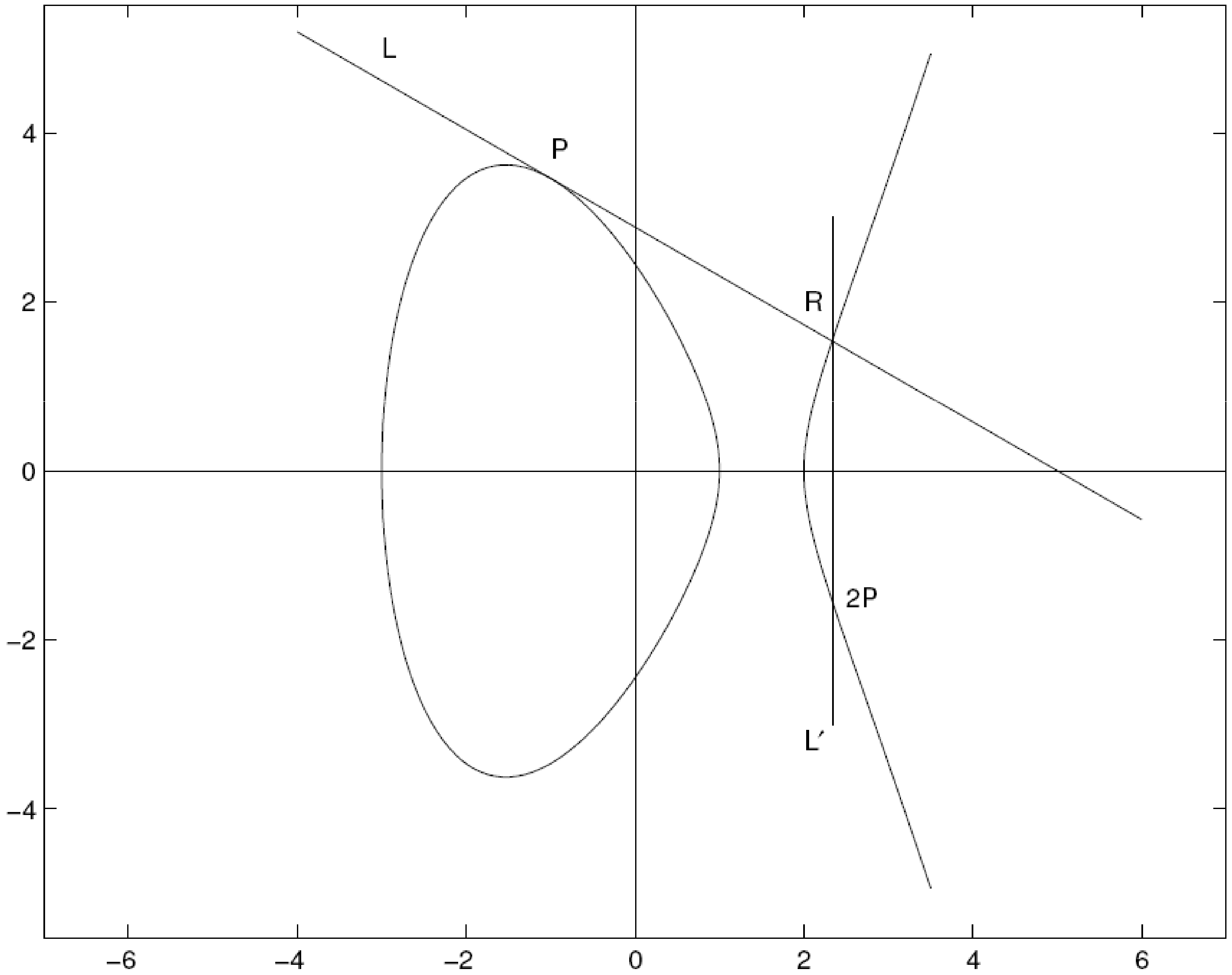
Input: $\boldsymbol{P}$, $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R} \leftarrow P$;
for $i \leftarrow n-2$ **down to** 0 **do**
  $\boldsymbol{R} \leftarrow 2 \cdot \boldsymbol{R}$;
  if $(k_i = 1)$ **then** $\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{P}$;
end
return $\boldsymbol{R}$

Group operations: point addition



Group operations: point doubling



Group law and implementation

- Addition: $P + Q = (x_3, y_3)$ with $x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2$

  if $P \neq \pm Q$ else $\mathcal{O}$ appears $y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1$

- Doubling: $2P = (x_3, y_3)$ with $x_3 = (\frac{3x_1^2 + a}{2y_1}) - x_1 - x_2$

  if $ord(P) > 2$ else $\mathcal{O}$ appears $y_3 = (\frac{3x_1^2 + a}{2y_1})(x_1 - x_3) - y_1$

- But these cases should never occur anyway
- Note that b is not used in the formulae

Group law and implementation (2)

- Implementations:
  - Coordinate system: affine, projective, Jacobian, etc.

- Full domain correct:
  - Implementation computes P+Q and 2·P correctly on the whole domain
  - For Weierstrass curves this typically requires IF statements

- Partial domain correct: not full domain correct
  - For some inputs, implementation
    - Crashes, e.g. division by zero for affine coordinates
    - No crash but gets stuck at fixed / invalid point for Jacobian and projective coord.

Group law and implementation (3)

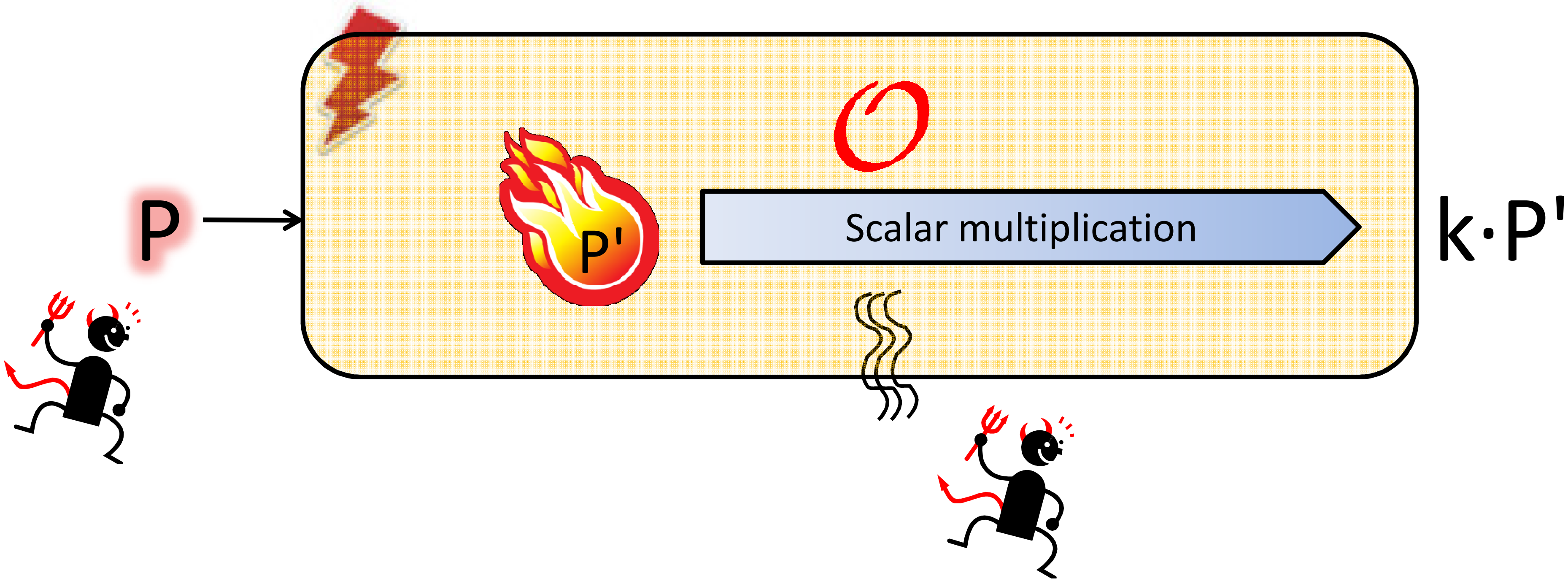
$E(\mathbb{F}_p)$: $y^2 = x^3 + ax + b$					
Coordinate System	Operation	Using a	Using b	Input	Output
Projective	PA($\boldsymbol{P_1}$, $\boldsymbol{P_2}$)	-	-	$\boldsymbol{P_1}=\boldsymbol{P_2}$	(0,0,0)
				$\boldsymbol{P_1}=-\boldsymbol{P_2}$	(0,*,0)
				$\boldsymbol{P_1}$=(0,*,0)	(0,0,0)
	PD($\boldsymbol{P_1}$)	+	-	Order($\boldsymbol{P_1}$)=2	(0,*,0)
				$\boldsymbol{P_1}$=(0,*,0)	(0,0,0)
Jacobian	PA($\boldsymbol{P_1}$, $\boldsymbol{P_2}$)	-	-	$\boldsymbol{P_1}=\boldsymbol{P_2}$	(0,0,0)
				$\boldsymbol{P_1}=-\boldsymbol{P_2}$	(*,*,0)
				$\boldsymbol{P_1}$=(*,*,0)	(*,*,0)
				$\boldsymbol{P_1}$=(0,0,0)	(0,0,0)
	PD($\boldsymbol{P_1}$)	+	-	Order($\boldsymbol{P_1}$)=2	(*,*,0)
				$\boldsymbol{P_1}$=(*,*,0)	(*,*,0)
				$\boldsymbol{P_1}$=(0,0,0)	(0,0,0)

Borderline cases for projective and Jacobian coordinates

Our attack

The attack

- Setting: target computes k·P for any given P, k is secret
- Idea: choose rogue, valid input P s.t. a fault ε turns it into 
  - P' is point of very low order



Points with low-order neighbours

- Given curve E: $y^2 = x^3 + ax + b$, integers l and δ
- Construct P (x_p, y_p) on E s.t.
  - $\exists$ curve E': $y^2 = x^3 + ax + b'$
  - With P'$(x_{p'}, y_{p'})$ of order l on E'
  - Hamming dist. of bit representations $x_p || y_p$ and $x_{p'} || y_{p'}$ is δ
- If $\delta = 1$ we call P and P' neighbours

- Input: E, l, δ ⟶
- Output: P and P' ⟵

Points with low-order neighbours

- NIST P-192 curve over $\mathbf{F}_p$ with $p = 2^{192} - 2^{64} - 1$, $a = -3$ and b = 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1

Order	P	bit-flip
2	xP = 0x6D9D789820A2C19237C96AD4B8D86B87FB49D4D6C728B84F yP = 0x1	0
3	xP = 0x8E1AEBDD6009F114490C7BC2C02509F8E432ED15F10C2D33 yP = 0x7A568946EFA602B3624A61E513E57869CAF2AE854E1A17B	2
4	xP = 0xB317D7BBD023E6293F1506221F5BC4A23D4BE2E05328C5F7 yP = 0xC70D48794F409831097620C0865B7D567329728C634CA6AE	0
5	xP = 0xCC9BCC0061F64371E3C3BDE165DAD5380A7DC1919765940 yP = 0xCC8B36B37928334B8AFD7A9FCCFB4B0773E94A4178093458	8
6	xP = 0xC3F76445E6A52138E283E485092F005BE0821C3F9E96B05E yP = 0x535DBCCB593D72E7885B66E57FD13A8FF9C57A8F8B91CE48	1
7	xP = 0x5C003567728CCBC9F4C06620B9973193837BAEC67A29E43A yP = 0x408D0C3135006B03EFF80961394D890F0E86D9FD1BA4EEC6	3
8	xP = 0x74FD6A1AD39479C75A85305FA786E1DBDC845E03754E723E yP = 0x6EF58ABFC0B71047BA4F425652B3EC1746EBE8FE16FEA1F5	1

Attack against a toy implementation

- Full domain correct
  - $\mathcal{O}$ and all following computations will be handled correctly
- Double and add scalar multiplication

- Input P with neighbour P' of order 4, inject fault and measure
- Doubling: 2·P', 2·2P', 2·3P' or 2·$\mathcal{O}$ (borderline cases)
- Addition: generates always odd multiples of P', never $\mathcal{O}$
- $\mathcal{O}$ occurs only after 2 consecutive doublings
- If $\mathcal{O}$ occurs during processing of bit k_i, bit k_{i+1} must be 0
- Uniquely identifies all 0 key bits (possibly except LSB)

Attack against a toy implementation

- Full domain correct
- Double and add scalar multiplication
- Input P with neighbour P' of order 4, inject fault and measure
- k = 5405 = $\not{1}010100011101_2$

i	11	10	9	8	7	6	5	4	3	2	1	0
k_i	0	1	0	1	0	0	0	1	1	1	0	1
$\boldsymbol{R}$	$2P'$	$\mathcal{O},P'$	$2P'$	$\mathcal{O},P'$	$2P'$	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O},P'$	$2P',3P'$	$2P',3P'$	$2P'$	$\mathcal{O},P'$

view	0p	$\mathcal{O}$	0p	0p	$\mathcal{O}$	0p	0p	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$	0p	0p	0p	0p	0p	0p	$\mathcal{O}$	0p
step 1	0			0			0	0	0							0		
step 2	0	1		0	1		0	0	0	1		1		1		0	1	

- Obtain all of k with a single trace!

Attack against a toy implementation

- Affine coordinates, partial domain correct (crash at 1st $\mathcal{O}$)
- Double and add scalar multiplication

- First occurrence of $\mathcal{O}$ leaks, then no more information
- For P' of order l, we obtain index $\mathrm{I}(l)$ s.t. the first $\mathrm{I}(l)$ bits of k form an integer divisible by l
  - Also information if not divisible by l

- Repeat with P' of increasing orders l
  - Requires several traces with the same k
- Incremental search algorithm, obtain almost all of k

Feasibility of attack

- Need to be able to choose input P s.t. P' is of low order
  - El Gamal encryption/decryption, static Diffie-Hellman, etc.
- Or: system with fixed base point where P is already rogue
  - Nice back-door: impossible to check all error patterns

- Fault injection: need a specific error ε
  - ε is 1 bit-flip, 256 random byte faults, only ε leads to P' and $\mathcal{O}$
  - ε can be adjusted to any likely error pattern, in all coordinates
  - Precise timing

- Side channel: need leakage
  - We assume leakage by IFs, crashes, zero-value coordinates, etc.
- Group law implementation
  - Attack does not apply if all curve coefficients are used in PA/PD formulas

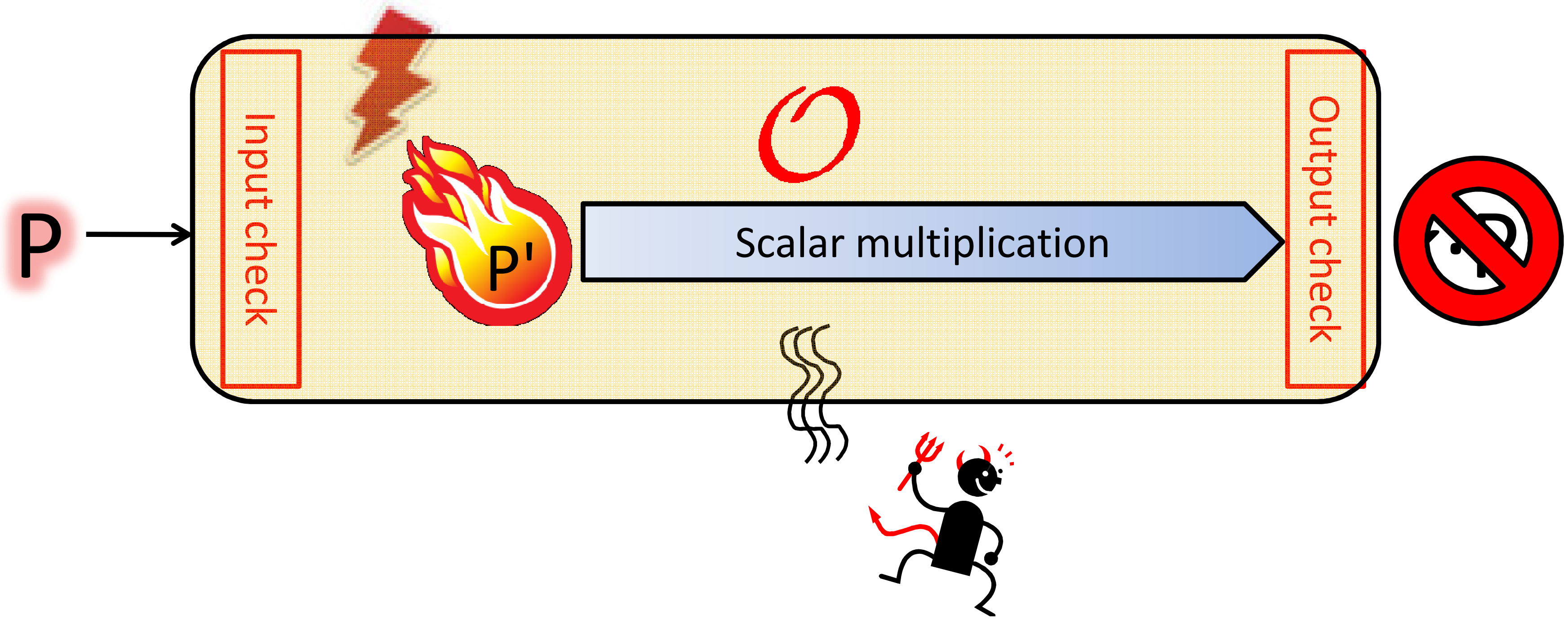
Our attack against protected implementations

Attacks on scalar multiplication and countermeasures

- SPA
  - Solution: regular algorithm / implementation (atomicity)
- DPA
  - Solution: key, field, curve and point randomization
- Faults
  - Solution: check output point and curve parameter validity
- Low-order attack (weak curve attack)
  - Solution: check input point validity
  - Small co-factor check (all NIST curves have co-factor 1)

Attack against protected implementations

- Input point validity check
  - No problem if we can inject fault after check but before mult
- Output point / curve parameters validity check
  - No problem, we already got the info



Attack against protected implementations

- Regular exponentiation algorithms / implementations to protect against SPA
  - Attack is fairly independent of scalar multiplication algorithm
  - Each algorithm computes some multiples of P that depend on k
  - If so, the attack applies

- Example: Montgomery powering ladder

Montgomery powering ladder

Algorithm 3: Montgomery powering ladder

Input: $\boldsymbol{P}$, $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow P$, $\boldsymbol{R}_1 \leftarrow 2 \cdot P$;
for $i \leftarrow n-2$ **down to** 0 **do**
    $\boldsymbol{R}_{\neg k_i} \leftarrow \boldsymbol{R}_{k_i} + \boldsymbol{R}_{\neg k_i}$, $\boldsymbol{R}_{k_i} \leftarrow 2 \cdot \boldsymbol{R}_{k_i}$;
end

return $\boldsymbol{R}_0$

- 2 registers R_0 and $R_1 = R_0 + P$
  - Input P with neighbour P' of order 4
  - If 2 consecutive key bits are equal, R_0 or R_1 doubled twice, $\mathcal{O}$ occurs
  - If 2 consecutive key bits are different, ordinary doublings
  - $\mathcal{O}$ can never be the result of an addition
- Obtain almost all of k with a single trace

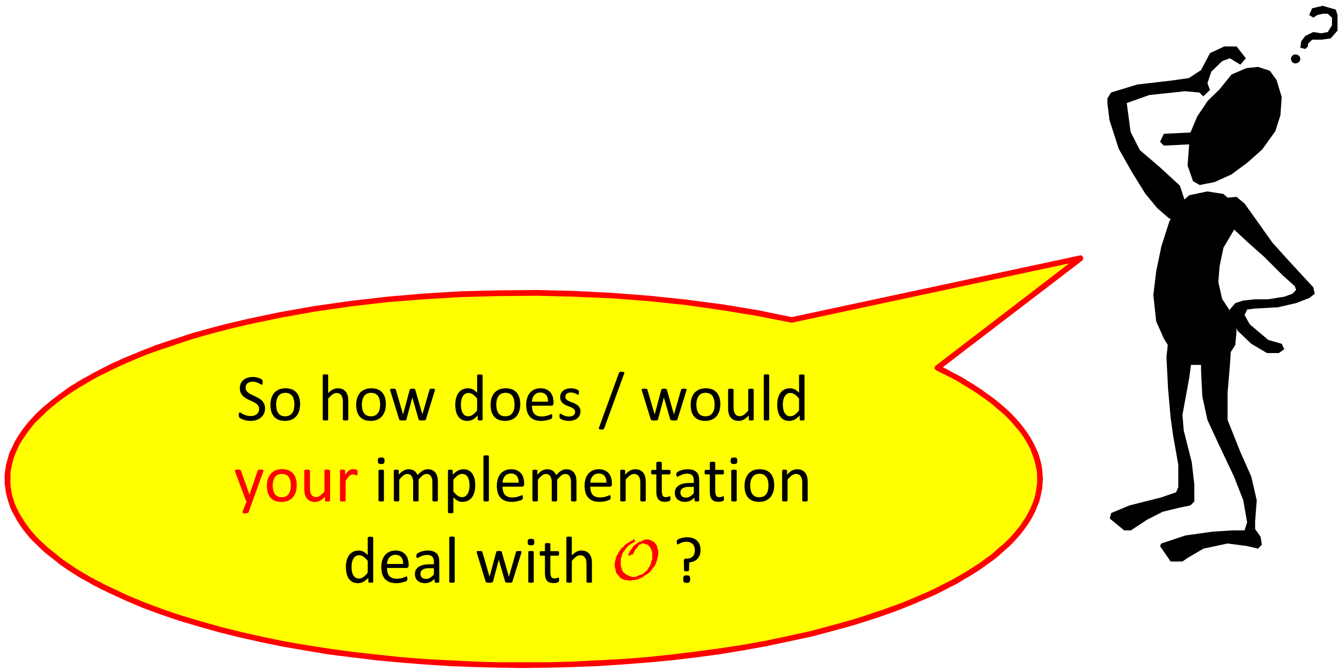
More in the paper

- Countermeasures we looked at
  - Random scalar splitting: $k = k_1+k_2$, $k \cdot P = k_1 \cdot P + k_2 \cdot P$
  - Scalar blinding: $k' = k + r \cdot \#E$
  - Ephemeral keys
  - Coordinate randomization, e.g. random projective coordinates
  - Random elliptic curve isomorphisms
  - Base point blinding
- Binary curves
  - Applicability of attack depends on coordinate system
  - Affine and standard projective coord.: attack applies since only a used
  - Jabobian: attack does not apply since a and b are used
  - Lopez-Dahab: attack does not apply; only b is used but changing a results in isomorphic curve over its quadratic twist

Conclusion

- Our attack:
  - Input rogue P and inject fault after initial checks
  - P turns into P' of low order
  - k·P' leads to $\mathcal{O}$ which can be detected via side channels
- Requires chosen inputs (or rogue fixed base point)
- Very powerful attack on full domain correct implementations
  - Defeats many countermeasures, requires only a single trace
- Combining countermeasures does not automatically protect against combined attacks
- Countermeasures that prevent our attack:
  - Sensors, concurrent validity checks, base point blinding, etc.

Thank you. Questions?



The paper: J. Fan, B. Gierlichs, F. Vercauteren, *To Infinity and Beyond: Combined attack on ECC using Points of Low Order*, CHES 2011