

# Fault Attacks on Elliptic Curve Cryptosystems

Marc Joye

Thomson Security Labs  
marc.joye@thomson.net

Crypto'Puces 2009 – Porquerolles, June 2–6, 2009



**THOMSON**

## Outline

---

Elliptic Curve Cryptography

Inducing Faults

Fault Attacks

Countermeasures

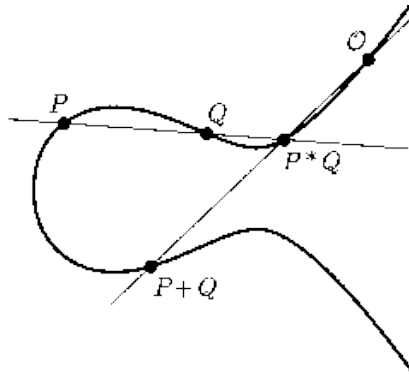
Concluding Remarks

**THOMSON**

# Elliptic Curve Cryptography

---

- Invented [independently] by Neil Koblitz and Victor Miller in 1985



- Useful for key exchange, encryption, digital signature, etc.

THOMSON

## Basics on Elliptic Curves (1/3)

---

### Definition

An elliptic curve over a field  $\mathbb{K}$  is the set of points  $(x, y) \in E$

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

along with the point  $O$  at infinity

- $\text{Char } \mathbb{K} \neq 2, 3 \Rightarrow a_1 = a_2 = a_3 = 0$
- $\text{Char } \mathbb{K} = 2$  (non-supersingular case)  $\Rightarrow a_1 = 1, a_3 = a_4 = 0$

### Fact

The set  $E(\mathbb{K})$  forms an **additive group** where

- $O$  is the neutral element
- the group law is given by the “chord-and-tangent” rule

THOMSON

# Basics on Elliptic Curves (2/3)

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$
- **Group law**
  - $P + O = O + P = P$
  - $-P = (x_1, -y_1 - a_1x_1 - a_3)$
  - $P + Q = (x_3, y_3)$  where

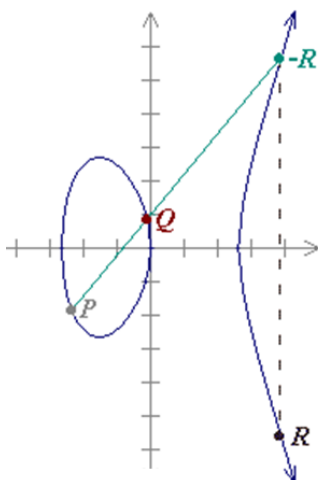
$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \quad y_3 = (x_1 - x_3)\lambda - y_1 - a_1x_3 - a_3$$

$$\text{with } \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{[addition]} \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{[doubling]} \end{cases}$$

THOMSON

# Basics on Elliptic Curves (3/3)

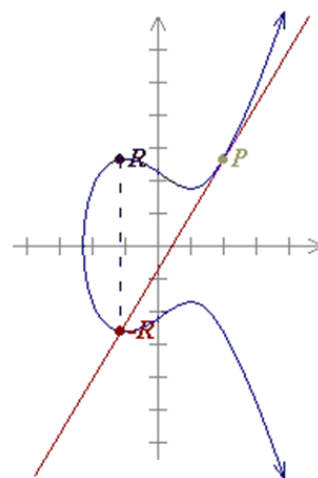
- Elliptic curves over  $\mathbb{R}$



$$y^2 = x^3 - 7x$$

$$P = (-2.35, -1.86), \quad Q = (-0.1, 0.836)$$

$$R = (3.89, -5.62)$$



$$y^2 = x^3 - 3x + 5$$

$$P = (2, 2.65)$$

$$R = (1.11, 2.64)$$

THOMSON

# EC Primitive

---

- EC primitive = point multiplication (a.k.a. scalar multiplication)

$$E(\mathbb{K}) \times \mathbb{Z} \rightarrow E(\mathbb{K}), (P, k) \mapsto Q = [k]P$$

- one-way function
- Cryptographic elliptic curves
  - $\mathbb{K} = \mathbb{F}_q$  with  $q = p$  (a prime) or  $q = 2^m$
  - $\#E(\mathbb{K}) = h n$  with  $h \in \{1, 2, 3, 4\}$  and  $n$  prime
  - typical size:  $|n|_2 = 160$  ( $\approx |\mathbb{K}|_2$ )

## Definition (ECDL Problem)

Let  $\mathbb{G} = \langle P \rangle \subseteq E(\mathbb{K})$  a subgroup of prime order  $n$ .

Given points  $P, Q \in \mathbb{G}$ , compute  $k$  such that  $Q = [k]P$

THOMSON

## EC Digital Signature Algorithm (1/2)

---

- Elliptic curve variant of the Digital Signature Algorithm
  - a.k.a. Digital Signature Standard – DSS
  - included in IEEE P1363, ANSI X9.62, FIPS 186.2, SECG, and ISO 15946-2
  - highest security level
- Domain parameters
  - finite field  $\mathbb{F}_q$
  - elliptic curve  $E/\mathbb{F}_q$  with  $\#E(\mathbb{F}_q) = h n$ 
    - cofactor  $h \leq 4$  and  $n$  prime
  - cryptographic hash function  $H$
  - point  $G \in E$  of prime order  $n$

$$\{\mathbb{F}_q, E, n, h, H, G\}$$

THOMSON

# EC Digital Signature Algorithm (2/2)

---

- Key generation:  $\mathbf{Y} = [d]\mathbf{G}$  with  $d \xleftarrow{\$} \{1, \dots, n-1\}$   
 $pk = \{\text{domain params}, \mathbf{Y}\}$  and  $sk = \{d\}$

- Signing

**Input** message  $m$  and **private key**  $sk$

**Output** signature  $S = (r, s)$

1. pick a **random**  $k \in \{1, \dots, n-1\}$
2. compute  $\mathbf{T} = [k]\mathbf{G}$  and set  $r = x(\mathbf{T}) \pmod{n}$
3. if  $r = 0$  then goto Step 1
4. compute  $s = (H(m) + dr)/k \pmod{n}$
5. return  $S = (r, s)$

- Verification

1. compute  $u_1 = H(m)/s \pmod{n}$  and  $u_2 = r/s \pmod{n}$
2. compute  $\mathbf{T} = [u_1]\mathbf{G} + [u_2]\mathbf{Y}$
3. check whether  $r \equiv x(\mathbf{T}) \pmod{n}$

**THOMSON**

## Public Key Validation

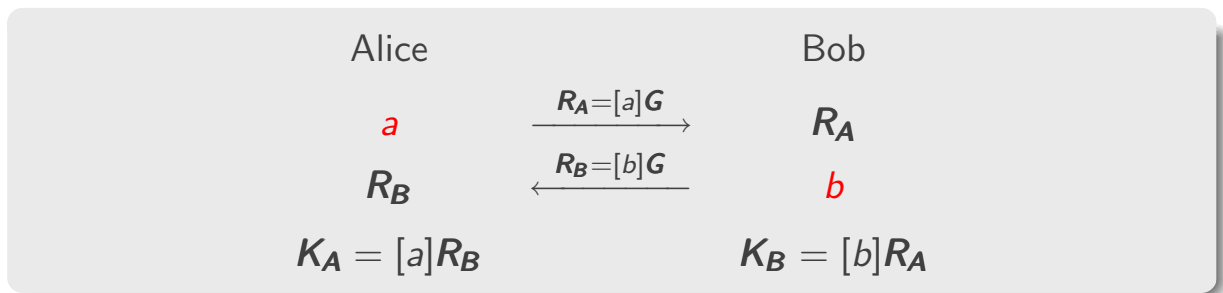
---

- For each received  $pk = \{\text{domain params}, \mathbf{Y}\}$ , check that
  1.  $\mathbf{Y} \in E$
  2.  $\mathbf{Y} \neq \mathbf{O}$
  3. **(optional)**  $[n]\mathbf{Y} = \mathbf{O}$

**THOMSON**

# EC Diffie-Hellman Key Exchange

- ECDH = Elliptic Curve Diffie-Hellman protocol
  - elliptic curve variant of the Diffie-Hellman key exchange

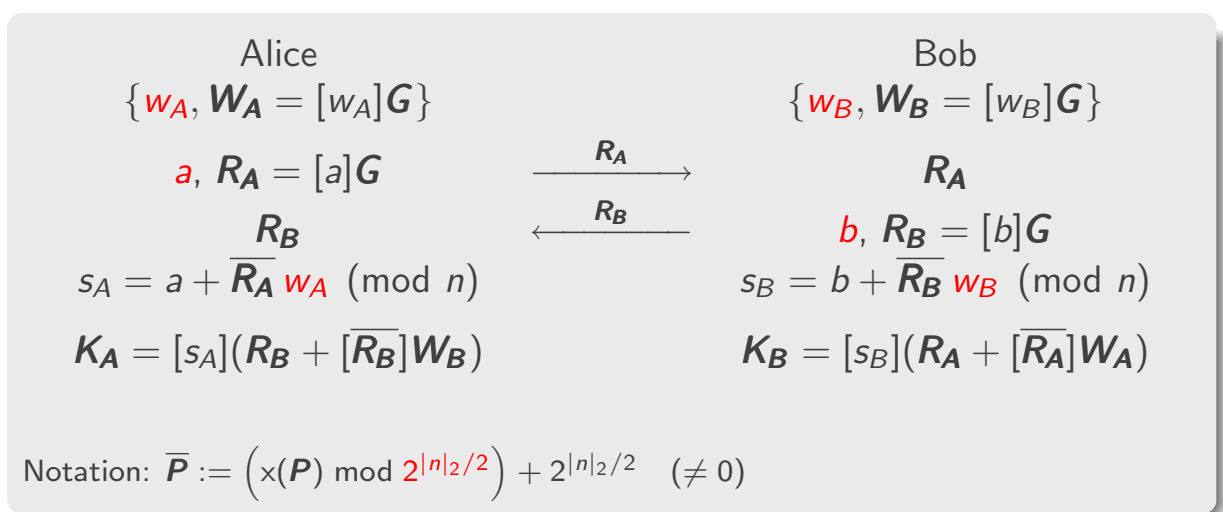


- cofactor variant:
 
$$K_A = [h]([a]R_B) \text{ and } K_B = [h]([b]R_A)$$
- suffers from the man-in-the-middle attack
  - no data-origin authentication
  - exchanged messages should be signed

**THOMSON**

# EC Menezes-Qu-Vanstone Protocol

- ECMQV = Elliptic Curve Menezes-Qu-Vanstone protocol
  - implicit authentication



- cofactor variant

**THOMSON**

# ECDH Augmented Encryption (1/2)

- ECIES = Elliptic Curve Integrated Encryption System
  - proposed by Michel Abdalla, Mihir Bellare and Phillip Rogaway in 2000
  - submitted to IEEE P1363a
  - highest security level (IND-CCA2)
- Domain parameters
  - finite field  $\mathbb{F}_q$
  - elliptic curve  $E/\mathbb{F}_q$  with  $\#E(\mathbb{F}_q) = hn$
  - “special” hash functions
    - message authentication code  $\text{MAC}_K(c)$
    - key derivation function  $\text{KD}(T, \ell)$
  - symmetric encryption algorithm  $\text{Enc}_K(m)$
  - point  $G \in E$  of prime order  $n$

$\{\mathbb{F}_q, E, n, h, \text{MAC}, \text{KD}, \text{Enc}, G\}$

THOMSON

# ECDH Augmented Encryption (2/2)

- Key generation:  $Y = [d]G$  with  $d \xleftarrow{\$} \{1, \dots, n-1\}$   
 $pk = \{\text{domain params}, Y\}$  and  $sk = \{d\}$
- ECIES encryption
  1. pick a random  $k \in \{1, \dots, n-1\}$
  2. compute  $U = [k]G$  and  $T = [k]Y$  (resp.  $T = [h][k]Y$ )
  3. set  $(K_1 || K_2) = \text{KD}(T, l)$
  4. compute  $c = \text{Enc}_{K_1}(m)$  and  $r = \text{MAC}_{K_2}(c)$
  5. return  $(U, c, r)$
- ECIES decryption
  - Input** ciphertext  $(U, c, r)$  and **private key**  $sk$
  - Output** plaintext  $m$  or  $\perp$ 
    1. compute  $T' = [d]U$  (resp.  $T' = [h][d]U$ )
    2. set  $(K'_1 || K'_2) = \text{KD}(T', l)$
    3. if  $\text{MAC}_{K'_2}(c) = r$  then return  $m = \text{Enc}_{K'_1}^{-1}(c)$

THOMSON

# History (1/2)

---

- 1996
  - September
    - Attacks on RSA-CRT by **Bellcore's researchers** (D. Boneh, R. DeMillo & R. Lipton)
    - Attack improvements by A. Lenstra
  - October
    - 18: DFA on **DES** by E. Biham & A. Shamir
    - 29: Attacks on RSA and ElGamal by F. Bao & R. Deng
    - 30: DFA on unknown cryptosystems by E. Biham & A. Shamir
  - November
    - Attacks on LUC and Demytko by M. Joye & J.-J. Quisquater

**THOMSON**

# History (2/2)

---

- 2000
  - Attacks on **ECC** by I. Biehl, B. Meyer & V. Müller
- 2003
  - Attacks on **AES** (5) by J. Blömer, C.-N. Chen, P. Dusart, C. Giraud, G. Letourneux, G. Piret, J.-J. Quisquater, J.-P. Seifert, O. Vivilo & S.-M. Yen

**THOMSON**



# Methods of Fault Injection (1/2)

---

## Glitch attacks

- Variations in **supply voltage** during execution may cause the processor to misinterpret or skip instructions
- Variations in the **external clock** may cause data misread or an instruction miss

## Temperature attacks

- Variations in **temperature** may cause
  - random modification of RAM cells
  - stopping read operations in NVMs to work

**THOMSON**

# Methods of Fault Injection (2/2)

---

## Light attacks

- Photoelectric effect (duration, power and location of the emission)
- **White light** (flash camera)
  - cheap equipment
- **Laser**
  - allows to **precisely** target a circuit area

## Magnetic attacks

- Emission of a powerful magnetic pulse near the silicon (duration, power and location of the emission)

**THOMSON**

# Types of Faults

---

- **Permanent faults**
  - destructive faults
  - the value of a cell is definitely changed
    - data (EEPROM or RAM)
    - code (EEPROM)
- **Transient faults**
  - provisional faults
  - the circuit recovers its original behavior after reset or when the fault's stimulus ceases
  - the code execution or a computation is perturbed:
    - instruction byte** a different instruction is executed (call to a routine skipped, test avoided, ...)
    - parameter byte** a different value or address is considered (operation with another operand, ...)

**THOMSON**

## (Transient) Fault Models

---

1. Fault model #1: **Precise bit errors**
  - The attacker can cause a fault in a single bit
  - Full control over the timing and location of the fault
2. Fault model #2: **Precise byte errors**
  - The attacker can cause a fault in a single byte
  - Full control over the timing but only partial control over the location (e.g., which byte is affected)
    - new faulty value cannot be predicted
3. Fault model #3: **Unknown byte errors**
  - The attacker can cause a fault in a single byte
  - Partial control over the timing and location of the fault
    - new faulty value cannot be predicted
4. Fault model #4: **Random errors**
  - Partial control over the timing and no control over the location

**THOMSON**

# Fault Attacks on ECC

---

- Bit-level vs. byte-level attacks
- Transient vs. permanent faults
- Private vs. public routines
- Unsigned vs. signed representations
- Fixed vs. variable base point
- Basic vs. provably secure systems

THOMSON

## Forcing-Bit Attack (1/2)

---

- Let  $d = \sum_{i=0}^{\ell-1} d_i 2^i$
- Forcing bit:  $d_j \rightarrow 0$

### ECDSA

- Check whether  $S = (r, s)$  is a valid signature
  - if so, then  $d_j = 0$
  - if not, then  $d_j = 1$
- (Similarly applies when  $k_j \rightarrow 0$  in Step 4)

THOMSON

# Forcing-Bit Attack (2/2)

---

- Let  $d = \sum_{i=0}^{\ell-1} d_i 2^i$
- Forcing bit:  $d_j \rightarrow 0$

## ECIES

- Check the ciphertext validity
  - if the output is  $m$  then  $d_j = 0$
  - if the output is  $\perp$  then  $d_j = 1$
- Replacing  $d$  with  $d \leftarrow d + r \text{ord}_E(\mathbf{P})$  may help to prevent the attack

THOMSON

# Flipping-Bit Attack

---

## Against ECDSA

- Let  $d = \sum_{i=0}^{\ell-1} d_i 2^i$
- Flipping bit:  $d_j \rightarrow \bar{d}_j$ 
  - $\Rightarrow \hat{S} = (r, \hat{s})$  with  $\begin{cases} \hat{s} = (H(m) + \hat{d}r)/k \pmod{n} \\ \hat{d} = (\bar{d}_j - d_j)2^j + d \end{cases}$
- Define  $\hat{u}_1 = H(m)/\hat{s} \pmod{n}$  and  $\hat{u}_2 = r/\hat{s} \pmod{n}$
- Compute  $\hat{T} = [\hat{u}_1]\mathbf{G} + [\hat{u}_2]\mathbf{Y}$
- For  $j = 0$  to  $\ell - 1$  and  $\sigma \in \{-1, 1\}$ , check if
$$\times \left( \hat{T} + \left[ \frac{\sigma 2^j r}{\hat{s}} \right] \mathbf{G} \right) = \times([\mathbf{k}]\mathbf{G}) = r \Rightarrow \bar{d}_j - d_j = \sigma$$
$$\Rightarrow d_j = \frac{1-\sigma}{2}$$

THOMSON

# Sign-Change Fault Attack

---

- Point inversion is inexpensive on elliptic curves

$$P = (x_1, y_1) \Rightarrow -P = (x_1, -y_1 - a_1 x_1 - a_3)$$

- Signed-digit point multiplication algorithms are preferred for computing  $Q = [d]P$

– e.g., NAF-based method gives a speed-up factor of 11.11%

- $d = \sum_{i=0}^{\ell} \delta_i 2^i$  with  $\delta_i \in \{0, 1, -1\}$

- Signed-digit encoding:  $\delta_i = (\text{sign bit}, \text{value bit})$ ,

$$0 = (\star, 0), \quad 1 = (0, 1), \quad -1 = (1, 1)$$

## Sign-change attack (specialized flipping-bit attack)

Induce a fault in the **sign bit** of  $\delta_i$

- on the fly
- during exponent recoding

THOMSON

## Safe-Error Attack (1/4)

---

- Double-and-add algorithm
  - additive variant of the square-and-multiply

---

Input:  $U, d = (d_{\ell-1}, \dots, d_0)_2$

Output:  $T = [d]U$

---

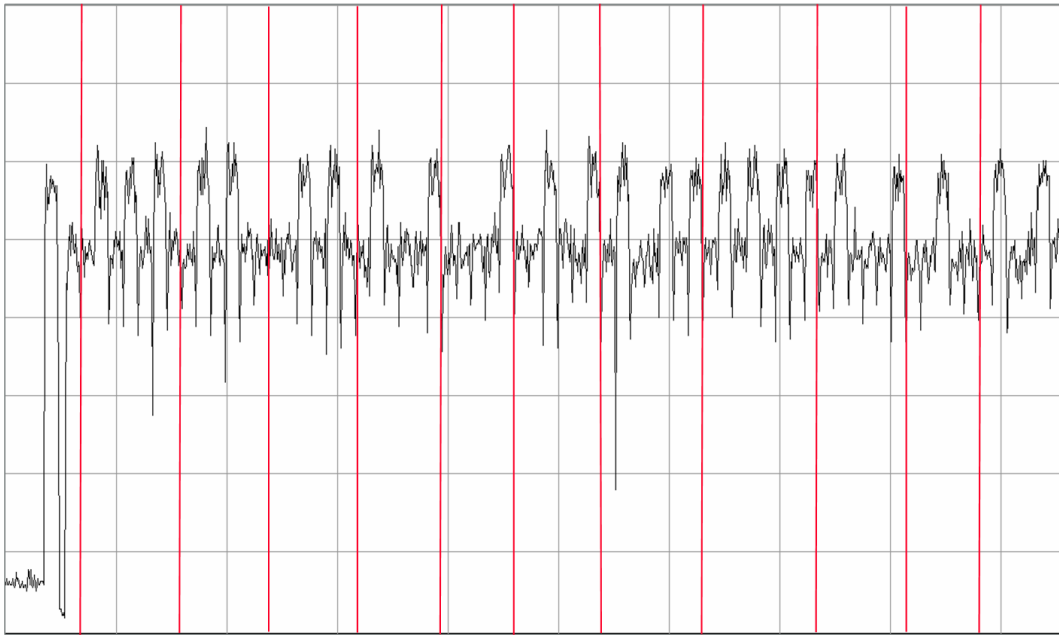
1.  $R_0 \leftarrow O; R_1 \leftarrow O$
  2. For  $i = \ell - 1$  downto 0 do
    - $R_0 \leftarrow [2]R_0$
    - if  $(d_i = 1)$  then  $R_0 \leftarrow R_0 + U$
  3. Return  $R_0$
- 

- ...subject to SPA

THOMSON

# Safe-Error Attack (2/4)

---



Secret:  $d = 2E\ C6\ 91\ 5B\ FE\ 4A\ \dots$

**THOMSON**

# Safe-Error Attack (3/4)

---

- Double-and-add-*always* algorithm
  - additive variant of the square-and-multiply-*always*

---

Input:  $U, d = (d_{\ell-1}, \dots, d_0)_2$

Output:  $T = [d]U$

---

1.  $R_0 \leftarrow O; R_1 \leftarrow O$
  2. For  $i = \ell - 1$  downto 0 do
    - $R_0 \leftarrow [2]R_0$
    - $b \leftarrow 1 - d_i; R_b \leftarrow R_b + U$
  3. Return  $R_0$
- 

- when  $b = 1$ , there is a **dummy** point addition
- the power trace now appears as a regular succession of doubles and adds

**THOMSON**

# Safe-Error Attack (4/4)

---

## Against ECIES

- **Timely** induce a fault into the ALU during the add operation at iteration  $i$
- Check the output
  - if an invalid ciphertext is notified (i.e.,  $\perp$ ) then the error was effective  $\Rightarrow d_i = 1$
  - if the result is correct then the point addition was dummy **[safe error]**  $\Rightarrow d_i = 0$
- Re-iterate the attack for another value of  $i$

## Lesson

Protection against certain implementation attacks (e.g., SPA) may introduce new vulnerabilities

**THOMSON**

## Errors in Public Routines

---

- Digital signatures are often used for authentication purposes
  - e.g., only signed software can run on a given device
- Idea: inject a fault during the **verification** process

**Public routines** (parameters) should be checked for faults

**THOMSON**

# Random Errors Against EC Primitive

---

## Attack model

- EC parameters are in non-volatile memory
  - permanent faults in a unknown position, in any system parameter
  - transient fault during parameter transfer

## Adversary's goal

- Recover the value of  $d$  in the computation of  $Q = [d]P$

THOMSON

## Key Observation (1/2)

---

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$
- $P + Q = (x_3, y_3)$  where
$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \quad y_3 = (x_1 - x_3)\lambda - y_1 - a_1x_3 - a_3$$
$$\text{with } \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{[addition]} \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{[doubling]} \end{cases}$$
- Parameter  $a_6$  is not involved in point addition (or point doubling)

THOMSON



## Key Observation (2/2)

---

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- If a 'point'  $\tilde{P} = (\tilde{x}, \tilde{y}) \in \mathbb{F}_q \times \mathbb{F}_q$  but  $\tilde{P} \notin E$  then the computation of  $\tilde{Q} = [d]\tilde{P}$  will take place on the curve

$$\tilde{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + \tilde{a}_6$$

where  $\tilde{a}_6 = \tilde{y}^2 + a_1\tilde{x}\tilde{y} + a_3\tilde{y} - \tilde{x}^3 - a_2\tilde{x}^2 - a_4\tilde{x}$

- Now if
  1.  $\text{ord}_{\tilde{E}}(\tilde{P}) = t$  is small
  2. discrete logarithms are computable in  $\langle \tilde{P} \rangle$

then

can be recovered from  $\tilde{Q} \quad d \pmod{t}$

**THOMSON**

## Chosen Input Point Attack

---

- Construct a 'point'  $\tilde{P}_i = (\tilde{x}_i, \tilde{y}_i) \in \tilde{E}_i$  such that
  1.  $\text{ord}_{\tilde{E}_i}(\tilde{P}_i) = t_i$  is small
  2. discrete logarithms are computable in  $\langle \tilde{P}_i \rangle$
- Query the device with  $\tilde{P}_i$  and receive  $\tilde{Q}_i = [d]\tilde{P}_i$
- Solve the discrete logarithm and recover  $d \pmod{t_i}$
- Iterating the process gives
  - $d \pmod{t_i}$  for several  $t_i$
  - $d$  by **Chinese remaindering**

**THOMSON**

# Faults in the Base Point

---

Recover  $d$  in  $Q = [d]P$  on  $E/\mathbb{F}_p : y^2 = x^3 + a_4x + a_6$

- Fault:  $P = (x_1, y_1) \rightarrow \hat{P} = (\hat{x}_1, y_1) \in \tilde{E}$
  - Device outputs  $\hat{Q} = [d]\hat{P}$
  - $\hat{Q} = [d](\hat{x}_1, y_1) = (\hat{x}_d, \hat{y}_d) \in \tilde{E}$   
 $\Rightarrow \tilde{a}_6 = \hat{y}_d^2 - \hat{x}_d^3 - a_4\hat{x}_d \pmod{p}$
  - $\hat{x}_1$  is a **root** in  $\mathbb{F}_p[X]$  of  $X^3 + a_4X + \tilde{a}_6 - y_1^2$
  - Compute  $d \pmod{t}$  from  $\hat{Q} = [d]\hat{P}$
- 
- Similar attack when the  $y$ -coordinate of  $P$  is corrupted
  - More assumptions are needed when both coordinates are corrupted

**THOMSON**

# Faults in the Definition Field

---

Recover  $d$  in  $Q = [d]P$  on  $E/\mathbb{F}_p : y^2 = x^3 + a_4x + a_6$

- Fault:  $p \rightarrow \hat{p}$
- Device outputs  $\hat{Q} = [d]\hat{P}$  with  $\hat{P} = (\hat{x}_1, \hat{y}_1)$  and  
 $\hat{x}_1 \equiv x_1 \pmod{\hat{p}}$  and  $\hat{y}_1 \equiv y_1 \pmod{\hat{p}}$
- $\hat{Q} = [d](\hat{x}_1, \hat{y}_1) = (\hat{x}_d, \hat{y}_d) \in \tilde{E}$   
 $\Rightarrow \tilde{a}_6 \equiv \hat{y}_d^2 - \hat{x}_d^3 - a_4\hat{x}_d \equiv \hat{y}_1^2 - \hat{x}_1^3 - a_4\hat{x}_1 \pmod{\hat{p}}$
- $\hat{p}$  **divides**  $(\hat{y}_d^2 - \hat{x}_d^3 - a_4\hat{x}_d) - (\hat{y}_1^2 - \hat{x}_1^3 - a_4\hat{x}_1)$
- Compute  $d \pmod{t}$  from  $\hat{Q} = [d]\hat{P}$

- Case of Mersenne primes; i.e.,  $p = 2^m \pm 2^t \pm 1$

**THOMSON**

# Faults in the Curve Parameters

---

Recover  $d$  in  $Q = [d]P$  on  $E/\mathbb{F}_p : y^2 = x^3 + a_4x + a_6$

- Fault:  $a_4 \rightarrow \hat{a}_4$
- Device outputs  $\hat{Q} = [d]P$  on  
 $\hat{E} : y^2 = x^3 + \hat{a}_4x + \tilde{a}_6$
- $\hat{Q} = [d](x_1, y_1) = (\hat{x}_d, \hat{y}_d) \in \hat{E}$
- Two equations:  
$$\begin{cases} y_1^2 = x_1^3 + \hat{a}_4x_1 + \tilde{a}_6 \\ \hat{y}_d^2 = \hat{x}_d^3 + \hat{a}_4\hat{x}_d + \tilde{a}_6 \end{cases}$$
  
 $\Rightarrow \hat{a}_4 = \dots, \tilde{a}_6 = \dots$
- Compute  $d \pmod{t}$  from  $\hat{Q} = [d]P$

**THOMSON**

## Countermeasures

---

- Algorithmic countermeasures
  - memory checks, randomization, duplication, verification
  - Shamir's trick (redundancy)
  - [rich] mathematical structure
- Basic vs. concrete systems
- Fixed vs. variable base point

**THOMSON**

# Basic Countermeasures

---

- Add CRC checks
  - for private **and** public parameters
- Randomize the computation
  - e.g.,  $d \leftarrow d + r n$  with  $n = \text{ord}_E(P)$
- Compute the operations twice
  - doubles the **running time**
- Verify the signatures
  - ECDSA verification is **slower** than signing
- Check that the **output point**  $Q = [k]P$  is in  $\langle P \rangle$ 
  - $Q \in E$
  - $[h]Q \neq O$  (only implies of large order)
- Use the **cofactor** variants

THOMSON

## Multiplier Randomization (1/2)

---

- Scalar  $d$  should be randomized
- $d^* \leftarrow d + r \#E$  may not be a good solution
  - **security issue**

### Example (secp160k1)

$p = 2^{160} - 2^{32} - 538D_{16}$  [generalized] Mersenne prime

$\#E = 01\ 00000000\ 00000000\ 0001B8FA\ 16DFAB9A\ CA16B6B3_{16}$

$\Rightarrow d^* = d + r \#E = (r)_2 \parallel d_{\ell-1} \cdots d_{\ell-t} \parallel$  some bits

THOMSON

# Multiplier Randomization (2/2)

---

- Use **splitting** methods

- additive:

$$[d]P = [d - r]P + [r]P$$

- multiplicative:

$$[d]P = [d r^{-1}]([r]P)$$

## Euclidean splitting

Write  $d = \lfloor d/r \rfloor r + (d \bmod r)$  for a random  $r$

$$\implies [d]P = [d \bmod r]P + [\lfloor d/r \rfloor]([r]P)$$

- Strauss-Shamir double ladder

THOMSON

# Infective Computation

---

- Observation [Sung-Ming Yen *et al.*, 2003]
  - Decisional tests should be avoided
  - Inducing a random fault in the status register flips the value of the zero flag bit with a probability of 50%

## Infective computation

Make the decisional tests implicit and “infect” the computation in case of error detection

Example:

*If  $(T[a] = b)$  then return  $a$  else error*

*$\implies$  Return  $(T[a] - b) \cdot r + a$*

THOMSON

# BOS<sup>+</sup> Algorithm (1/2)

---

- J. Blömer, M. Otto, and J.-P. Seifert
- Application of Shamir's trick to elliptic curves

## Definition (Elliptic curves over a ring)

$$\tilde{E}(\mathbb{Z}_{pt}) := \{(x, y) \in \mathbb{Z}_{pt} \times \mathbb{Z}_{pt} : y^2 = x^3 + \tilde{a}_4x + \tilde{a}_6\} \cup \{\mathbf{O}_{pt}\}$$
$$\subsetneq E(\mathbb{F}_p) \times E(\mathbb{F}_t)$$

$$P = (x_1, y_1) \in E/\mathbb{F}_p : y^2 = x^3 + a_4x + a_6$$

1. Define  $a_{6,t} = y_1^2 - x_1^3 - a_4x_1 \pmod{t}$  so that  
 $P_t := P \pmod{t} \in E/\mathbb{F}_t : y^2 = x^3 + a_4x + a_{6,t}$ 
  - $\#E(\mathbb{F}_t)$  can be smooth
2. Fix  $E_t(\mathbb{F}_t) = \langle P_t \rangle$ , a **prime order** elliptic curve

THOMSON

# BOS<sup>+</sup> Algorithm (2/2)

---

Input:  $P \in E, k$

Output:  $Q = [k]P$

In memory:  $\{E_t, P_t \in E_t, n_t = \#E_t\}$

---

1. Compute

1.1  $P_{pt} \leftarrow \text{CRT}(P, P_t)$  and  $\tilde{E}_{pt} \leftarrow \text{CRT}(E, E_t)$

1.2  $Q_{pt} \leftarrow [k]P_{pt} \in \tilde{E}_{pt} = (X_{pt} : Y_{pt} : Z_{pt})$

1.3  $Q_t \leftarrow [k \pmod{n_t}]P_t \in E_t = (X_t : Y_t : Z_t)$

1.4 
$$\begin{cases} c_x \leftarrow 1 + \theta_x(X_{pt}Z_t - X_tZ_{pt}) \pmod{t} \\ c_y \leftarrow 1 + \theta_y(Y_{pt}Z_t - Y_tZ_{pt}) \pmod{t} \end{cases} \quad [\theta_x, \theta_y > t]$$

2. For a  $\kappa$ -bit random  $r$ , compute  $\gamma \leftarrow \lfloor \frac{rc_x + (2^\kappa - r)c_y}{2^\kappa} \rfloor$

3. Return  $Q = [\gamma]Q_p \pmod{p} \in E$

---

THOMSON

# Recommendations

---

- Consider **fault attacks** when implementing cryptographic routines
  - **check** that the countermeasures do not introduce new vulnerabilities
- Protect private **and** public routines/parameters
  - perform memory checks
- **Randomize** the execution
  - prefer the **splitting** methods
- Avoid **decisional** tests (single points of failure)
  - make use of infective computation
- Always use cryptographic **standards**
  - prefer the **cofactor** variants
- Combine **hardware** and **software** protections

**THOMSON**

## Further Research

---

### For the cryptanalyst:

1. Mount a fault attack against ECDSA
2. Mount a fault attack against ECIES

- Known [strong] attacks:
  - forcing-bit attack (safe-error attack): ECDSA and ECIES
  - flipping-bit attack (sign-change fault attack): ECDSA
- Extend the attacks when  $d$  is randomized





### For the designer:

Prove the security of the above schemes against faults in a given security model

**THOMSON**

# Bibliography

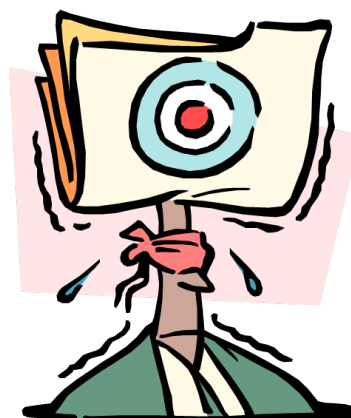
---

-  I. Biehl, B. Meyer, and V. Müller  
Differential fault analysis of elliptic curve cryptosystems  
*Proc. of CRYPTO 2000*, pp. 131–146
-  J. Blömer and M. Otto, and J.-P. Seifert  
Sign change attacks on elliptic curve cryptosystems  
*Proc. of FDTC 2005*, pp. 25–40
-  D. Boneh, R.A. DeMillo, and R.J. Lipton  
On the importance of eliminating errors in cryptographic computations  
*J. Cryptology* **14**(2):101–119, 2001
-  M. Ciet and M. Joye  
Elliptic curve cryptosystems in the presence of permanent and transient faults  
*Designs, Codes and Cryptography* **36**(1):33–43, 2005

**THOMSON**

## Comments/Questions?

---



<http://www.geocities.com/MarcJoye/>

<http://joye.site88.net/>

**THOMSON**