Elliptic Curve Cryptosystems in the Presence of Faults

Marc Joye

Thomson Security Labs
marc.joye@thomson.net

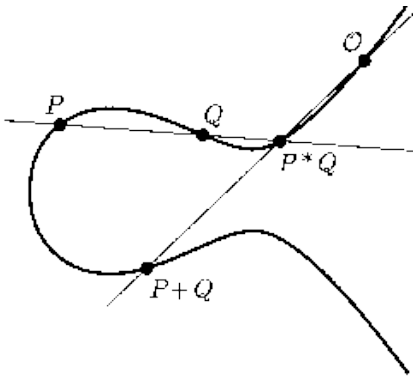
Outline

Outline

THOMSON images & beyond

Elliptic Curve Cryptography

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



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

THOMSON images & beyond

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

- Char $\mathbb{K} \neq 2, 3 \Rightarrow a_1 = a_2 = a_3 = 0$
- 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

Basics on Elliptic Curves (2/3)

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

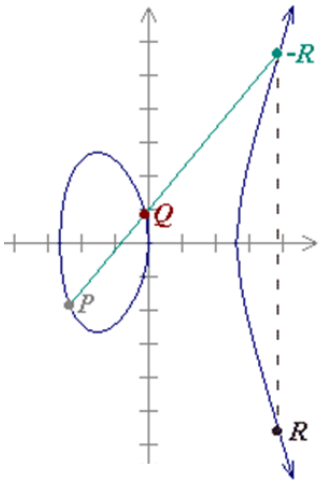
- Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$
- Group law
  - $P + O = O + P = P$
  - $-P = (x_1, -y_1 - a_1 x_1 - a_3)$
  - $P + Q = (x_3, y_3)$ where

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

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

Basics on Elliptic Curves (3/3)

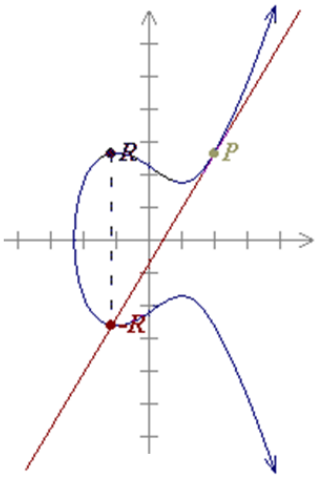
- Elliptic curves over $\mathbb{R}$



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

$\boldsymbol{P} = (-2.35, -1.86), \boldsymbol{Q} = (-0.1, 0.836)$

$\boldsymbol{R} = (3.89, -5.62)$



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

$\boldsymbol{P} = (2, 2.65)$

$\boldsymbol{R} = (1.11, 2.64)$

EC Primitive

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

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

  - 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 $\mathbb{G} = \langle \boldsymbol{P} \rangle \subseteq E(\mathbb{K})$ a subgroup of prime order n.
Given points $\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{G}$, compute k such that $\boldsymbol{Q} = [k]\boldsymbol{P}$

THOMSON
images & beyond

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 in the GM
- Domain parameters
  - finite field $\mathbb{F}_q$
  - elliptic curve $E/\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = h\,n$
    - cofactor $h \leqslant 4$ and n prime
  - cryptographic hash function H
  - point $\boldsymbol{G} \in E$ of prime order n

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

EC Digital Signature Algorithm (2/2)

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

- Signing

  Input message m and private key sk
  Output signature $S = (r, s)$

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

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

Public Key Validation

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

THOMSON images & beyond

EC Diffie-Hellman Key Exchange

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

Alice		Bob
a	$\xrightarrow{R_A=[a]G}$	R_A
R_B	$\xleftarrow{R_B=[b]G}$	b
$K_A = [a]R_B$		$K_B = [b]R_A$

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

THOMSON images & beyond

EC Menezes-Qu-Vanstone Protocol

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

Alice		Bob
$\{w_A, \boldsymbol{W_A} = [w_A]\boldsymbol{G}\}$		$\{w_B, \boldsymbol{W_B} = [w_B]\boldsymbol{G}\}$
$a,\ \boldsymbol{R_A} = [a]\boldsymbol{G}$	$\xrightarrow{\boldsymbol{R_A}}$	$\boldsymbol{R_A}$
$\boldsymbol{R_B}$	$\xleftarrow{\boldsymbol{R_B}}$	$b,\ \boldsymbol{R_B} = [b]\boldsymbol{G}$
$s_A = a + \overline{\boldsymbol{R_A}}\, w_A \pmod{n}$		$s_B = b + \overline{\boldsymbol{R_B}}\, w_B \pmod{n}$
$\boldsymbol{K_A} = [s_A](\boldsymbol{R_B} + [\overline{\boldsymbol{R_B}}]\boldsymbol{W_B})$		$\boldsymbol{K_B} = [s_B](\boldsymbol{R_A} + [\overline{\boldsymbol{R_A}}]\boldsymbol{W_A})$

Notation: $\overline{\boldsymbol{P}} := \left(\mathrm{x}(\boldsymbol{P}) \bmod 2^{|n|_2/2}\right) + 2^{|n|_2/2} \quad (\neq 0)$

  - cofactor variant

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) in the GM/ROM/SM
- Domain parameters
  - finite field $\mathbb{F}_q$
  - elliptic curve $E/\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = h\,n$
  - "special" hash functions
    - message authentication code $\mathrm{MAC}_K(c)$
    - key derivation function $\mathrm{KD}(\boldsymbol{T}, \ell)$
  - symmetric encryption algorithm $\mathtt{Enc}_K(m)$
  - point $\boldsymbol{G} \in E$ of prime order n

$$\{\mathbb{F}_q, E, n, h, \mathrm{MAC}, \mathrm{KD}, \mathtt{Enc}, \boldsymbol{G}\}$$

ECDH Augmented Encryption (2/2)

- Key generation: $\boldsymbol{Y} = [d]\boldsymbol{G}$ with $d \xleftarrow{\$} \{1, \ldots, n-1\}$
  $pk = \{\text{domain params}, \boldsymbol{Y}\}$ and $sk = \{d\}$
- ECIES encryption
  1. pick a random $k \in \{1, \ldots, n-1\}$
  2. compute $\boldsymbol{U} = [k]\boldsymbol{G}$ and $\boldsymbol{T} = [k]\boldsymbol{Y}$ (resp. $\boldsymbol{T} = [h][k]\boldsymbol{Y}$)
  3. set $(K_1 \| K_2) = \mathrm{KD}(\boldsymbol{T}, I)$
  4. compute $c = \mathtt{Enc}_{K_1}(m)$ and $r = \mathrm{MAC}_{K_2}(c)$
  5. return $(\boldsymbol{U}, c, r)$
- ECIES decryption

  Input ciphertext $(\boldsymbol{U}, c, r)$ and private key sk
  Output plaintext m or $\perp$
  1. compute $\boldsymbol{T'} = [d]\boldsymbol{U}$ (resp. $\boldsymbol{T'} = [h][d]\boldsymbol{U}$)
  2. set $(K_1' \| K_2') = \mathrm{KD}(\boldsymbol{T'}, I)$
  3. if $\mathrm{MAC}_{K_2'}(c) = r$ then return $m = \mathtt{Enc}^{-1}_{K_1'}(c)$

THOMSON images & beyond

Outline

THOMSON images & beyond

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

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

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 images & beyond

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 images & beyond

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 images & beyond

(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 images & beyond

Outline

THOMSON images & beyond

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 images & beyond

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 images & beyond

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 $d \leftarrow d + r \, \mathrm{ord}_E(P)$ may help to prevent the attack

THOMSON images & beyond

Flipping-Bit Attack

Against ECDSA

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

$$\Rightarrow \hat{S} = (r, \hat{s}) \text{ with } \begin{cases} \hat{s} = (H(m) + \hat{d}\, r)/k \pmod{n} \\ \hat{d} = (\overline{d_j} - d_j)2^j + d \end{cases}$$

- Define $\hat{u}_1 = H(m)/\hat{s} \pmod{n}$ and $\hat{u}_2 = r/\hat{s} \pmod{n}$
- Compute $\hat{\boldsymbol{T}} = [\hat{u}_1]\boldsymbol{G} + [\hat{u}_2]\boldsymbol{Y}$
- For $j = 0$ to $\ell - 1$ and $\sigma \in \{-1, 1\}$, check if

$$\mathrm{x}\left(\hat{\boldsymbol{T}} + \left[\frac{\sigma\, 2^j r}{\hat{s}}\right]\boldsymbol{G}\right) = \mathrm{x}([k]\boldsymbol{G}) = r \;\Rightarrow \overline{d_j} - d_j = \sigma \;\Rightarrow d_j = \tfrac{1-\sigma}{2}$$

Sign-Change Fault Attack

- Point inversion is inexpensive on elliptic curves

$$\boldsymbol{P} = (x_1, y_1) \quad \Rightarrow \quad -\boldsymbol{P} = (x_1, -y_1 - a_1\, x_1 - a_3)$$

- Signed-digit point multiplication algorithms are preferred for computing $\boldsymbol{Q} = [d]\boldsymbol{P}$
  - 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: $\delta_i = (\text{sign bit}, \text{value bit})$,

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

Sign-change attack (specialized flipping-bit attack)

Induce a fault in the sign bit of δ_i

- on the fly
- during exponent recoding

Safe-Error Attack (1/4)

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

Input: $\boldsymbol{U}, d = (d_{\ell-1}, \ldots, d_0)_2$
Output: $\boldsymbol{T} = [d]\boldsymbol{U}$

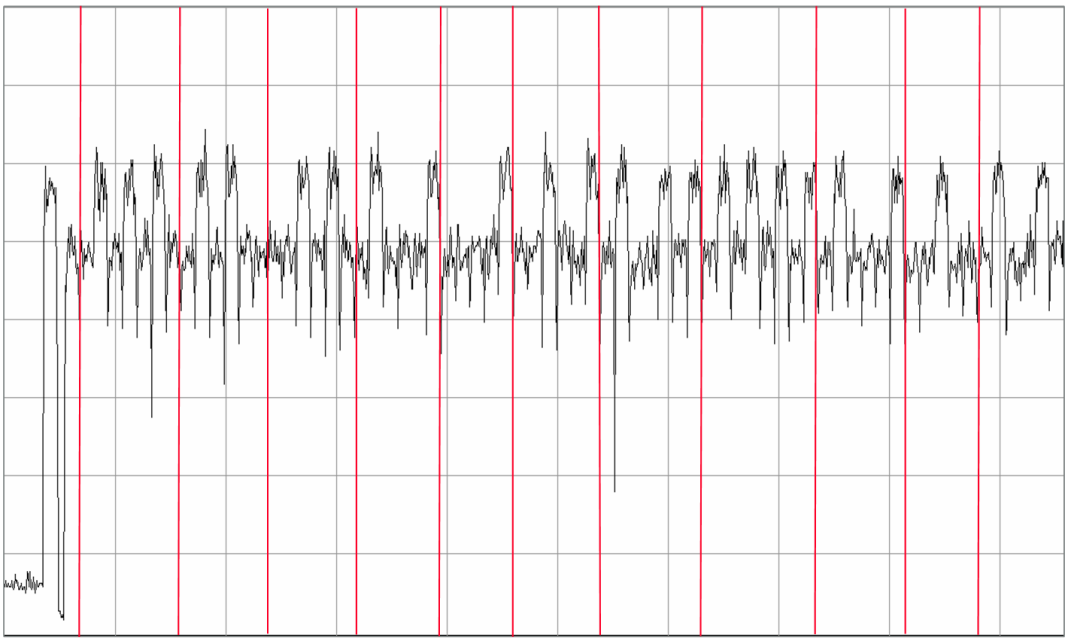
---

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

  - . . . subject to SPA

THOMSON images & beyond

Safe-Error Attack (2/4)



Secret: $d = \texttt{2E C6 91 5B FE 4A} \ldots$

THOMSON images & beyond

Safe-Error Attack (3/4)

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

Input: $\boldsymbol{U}, d = (d_{\ell-1}, \ldots, d_0)_2$
Output: $\boldsymbol{T} = [d]\boldsymbol{U}$

---

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

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

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

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 images & beyond

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 $\boldsymbol{Q} = [d]\boldsymbol{P}$

THOMSON images & beyond

Key Observation (1/2)

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

- Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$
- $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ where

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

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

- Parameter a_6 is not involved in point addition (or point doubling)

Key Observation (2/2)

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

- If a 'point' $\tilde{\boldsymbol{P}} = (\tilde{x}, \tilde{y}) \in \mathbb{F}_q \times \mathbb{F}_q$ but $\tilde{\boldsymbol{P}} \notin E$ then the computation of $\tilde{\boldsymbol{Q}} = [d]\tilde{\boldsymbol{P}}$ 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. $\mathrm{ord}_{\tilde{E}}(\tilde{\boldsymbol{P}}) = t$ is small
  2. discrete logarithms are computable in $\langle\tilde{\boldsymbol{P}}\rangle$

  then

$$d \pmod t$$

  can be recovered from $\tilde{\boldsymbol{Q}}$

Chosen Input Point Attack

- Construct a 'point' $\tilde{P}_i = (\tilde{x}_i, \tilde{y}_i) \in \tilde{E}_i$ such that
  1. $\mathrm{ord}_{\tilde{E}_i}(\tilde{P}_i) = t_i$ 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 images & beyond

Faults in the Base Point

Recover d in $Q = [d]P$ on $E_{/\mathbb{F}_p} : y^2 = x^3 + a_4 x + a_6$

- 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 $X^3 + a_4 X + \tilde{a}_6 - y_1^2$
- 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 images & beyond

Faults in the Definition Field

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

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

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

Faults in the Curve Parameters

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

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

Outline

THOMSON images & beyond

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 images & beyond

Basic Countermeasures

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

Multiplier Randomization (1/2)

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

Example (secp160k1)

$p = 2^{160} - 2^{32} - 538\mathrm{D}_{16}$ [generalized] Mersenne prime

$\#E = 01\ 00000000\ 00000000\ 0001\mathrm{B8FA}\ 16\mathrm{DFAB9A}\ \mathrm{CA16B6B3}_{16}$

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

Multiplier Randomization (2/2)

- Use splitting methods
  - additive:
$$[d]\boldsymbol{P} = [d - r]\boldsymbol{P} + [r]\boldsymbol{P}$$
  - multiplicative:
$$[d]\boldsymbol{P} = [d\, r^{-1}]([r]\boldsymbol{P})$$

Euclidean splitting

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

$$\Longrightarrow [d]\boldsymbol{P} = [d \bmod r]\boldsymbol{P} + \left[\lfloor d/r \rfloor\right]([r]\boldsymbol{P})$$

- Strauss-Shamir double ladder

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 $(\mathrm{T}[a] = b)$ *then return* a *else* `error`

$\Rightarrow$ *Return* $(\mathrm{T}[a] - b) \cdot r + a$

THOMSON images & beyond

BOS$^+$ 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)

$$\tilde{E}(\mathbb{Z}_{pt}) := \{(x,y) \in \mathbb{Z}_{pt} \times \mathbb{Z}_{pt} : y^2 = x^3 + \tilde{a}_4 x + \tilde{a}_6\} \cup \{O_{pt}\}$$
$$\subsetneq E(\mathbb{F}_p) \times E(\mathbb{F}_t) = \tilde{E}(\mathbb{Z}_{pt}) \cup \{[O_p, P_t], [P_p, O_t]\}$$

$$P = (x_1, y_1) \in E_{/\mathbb{F}_p} : y^2 = x^3 + a_4 x + a_6$$

1. Define $a_{6,t} = y_1^2 - x_1^3 - a_4 x_1 \pmod t$ so that
   $P_t := P \bmod t \in E_{/\mathbb{F}_t} : y^2 = x^3 + a_4 x + a_{6,t}$
   - $\#E(\mathbb{F}_t)$ can be smooth
2. Fix $E_t(\mathbb{F}_t) = \langle P_t \rangle$, a prime order elliptic curve

BOS$^+$ 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 $P_{pt} \leftarrow \mathrm{CRT}(P, P_t)$ and $\tilde{E}_{pt} \leftarrow \mathrm{CRT}(E, E_t)$
   - 1.2 $Q_{pt} \leftarrow [k]P_{pt} \in \tilde{E}_{pt}$ $\quad = (X_{pt} : Y_{pt} : Z_{pt})$
   - 1.3 $Q_t \leftarrow [k \pmod{n_t}]P_t \in E_t$ $\quad = (X_t : Y_t : Z_t)$
   - 1.4 $\begin{cases} c_x \leftarrow 1 + X_{pt}Z_t - X_t Z_{pt} \pmod t \\ c_y \leftarrow 1 + Y_{pt}Z_t - Y_t Z_{pt} \pmod t \end{cases}$
2. For a κ-bit random r, compute $\gamma \leftarrow \left\lfloor \frac{r\,c_x + (2^\kappa - r)c_y)}{2^\kappa} \right\rfloor$
3. Return $Q = [\gamma]Q_p \pmod p \in E$

Outline

THOMSON images & beyond

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 images & beyond

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 images & beyond

Bibliography

I. Biehl, B. Meyer, and V. Müller
Differential fault analysis of elliptic curve cryptosystems
Proc. of *CRYPTO 2000*, pp. 131–146

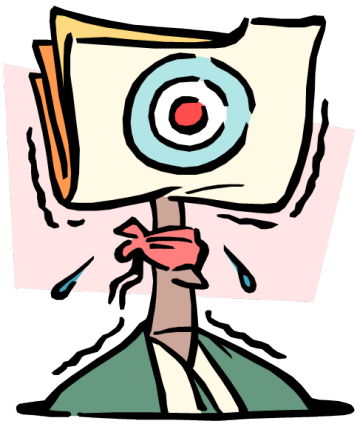
J. Blömer and M. Otto, and J.-P. Seifert
Sign change attacks on elliptic curve cryptosystems
Proc. of *FDTC 2005*, pp. 25–40

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 images & beyond

Comments/Questions?



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

THOMSON
images & beyond