Side Channel attacks on ECC

Review of possible attacks

Franck De Goër
Guillaume Jeanne

SCCI - Ensimag

January, 2014
Introduction

**Subject** Side channel attacks over ECC algorithms.

**Goal** Perform attacks to recover a secret key from a cryptographic algorithm using elliptic curve. Make an overview of different methods and detail some of them that have an important impact on ECC cryptography.
Overview

1. The double-and-add algorithm
   - Universality
   - Importance

2. Non-invasive attacks
   - Timing attacks
   - Power analysis

3. Perturbation attacks
   - Fault Model
   - Naive attack
   - During the computation
What is a side channel attack?

- Take advantage of the interaction between a device and its environment.
- Apply to the embedded world: smartcards, microcontrollers
- All environment variables: time, power consumption, electromagnetic field, acoustic...
Overview

1. The double-and-add algorithm
   - Universality
   - Importance

2. Non-invasive attacks
   - Timing attacks
   - Power analysis

3. Perturbation attacks
   - Fault Model
   - Naive attack
   - During the computation
The double-and-add algorithm

- Used to compute scalar multiplication
- Responsible of the calculation acceleration
- We consider DLP as a hard problem
- Used in all cryptographic algorithms based on DLP
Double and add algorithm

**Data**: P : base point ; d : scalar  
**Result**: Q = [d]P resulting point  
Q ← 0; 
l ← \( \log_2(d) \);  
for \( i \) from \( l \) downto 0 do  
    /* Using point doubling */  
    Q ← [2]Q;  
    if \( d[i] == 1 \) then  
        /* Using point adding */  
        Q ← Q + P;  
    end  
end  
return Q

**Algorithm 1**: Basic Double-and-Add algorithm
Double and add in ECIES

Data: \((R, c, d)\): cipher-text, \(EC\): elliptic curve, \(P\): base point, \(n\): order of \(P\), \(k\): private key, \(E\): symmetric encryption scheme, \(MAC\): message authentication code function, \(KDF\): key derivation function

Result: \(m\): message

1. \((x, y) \leftarrow [k]R;\)
2. if \(x == 0 \text{ and } y == 0\) then
3. \qquad return Failure;
4. end
5. /* Key derivation */
6. \(k_e || k_m \leftarrow \text{KDF}(x);\)
7. if \(d \neq \text{MAC}(k_m, c)\) then
8. \qquad return Failure;
9. end
10. /* Decryption of the symmetric cipher */
11. \(m = E^{-1}(k_e, c);\)
12. return \(m\)

Algorithme 2: Decipher \(c\) in ECIES
Overview

1. The double-and-add algorithm
   - Universality
   - Importance

2. Non-invasive attacks
   - Timing attacks
   - Power analysis

3. Perturbation attacks
   - Fault Model
   - Naive attack
   - During the computation
Non-invasive attacks

- Do not destroy or alter the functioning of the device
- Cannot be detected by the owner of the device
- Physical enclosures, auditing capabilities, and attack detectors are ineffective
- $\implies$ Keys are not revoked!
Timing attacks

- First discovered side channel attack.
- Measuring the time dedicated to the computation of an algorithm.
- Deduce how many times the program runs into a given loop.
- Not used anymore because Power analysis is as simple and more efficient.
Model for processor consumption power

- Each low level instruction consumes a given amount of power $p_i$ (depending on the number of clock cycles, registers, etc...)
- Manipulated data impact the power consumption with their hamming weight $h_{data_i}$.

$$e_i = p_i + \sum_{j=1}^{n} \alpha_j \cdot h_{data_j,i}$$

$$p_i \approx 10^2 \cdot \alpha_i \cdot h_{data_i}$$
Simple power analysis

- Need an oscilloscope and a card reader
- One measurement of the consumption, use \( p_i \)
- Point doubling in EC is more expensive than the point adding
- If the addition is computed then \( d[i] = 1 \)
Can you recover the secret key?

**Figure**: SPA against double-and-add from an IPod touch
Protection against SPA

Data : P : base point ; d : scalar
Result : Q = \([d]P\) resulting point
Q \leftarrow 0;
R \leftarrow 0;
l \leftarrow \log_2(d);
for \(i\) from \(l\) downto 0 do
    /* Using point doubling */
    Q \leftarrow [2]Q;
    if \(d[i] == 1\) then
        /* Using point adding */
        Q \leftarrow Q + P;
    end
    else
        /* R is not used */
        R \leftarrow Q + P;
    end
end
return Q

Algorithme 3 : Resistant Double-and-Add algorithm
Differential power analysis

- A statistical method
- Rely on small variations of power $\alpha_i . h_{data}$
- Require a lot of experiments, acquisition of many curves
- Need to be more precise on the measurement
Differential power analysis

- Choose a selection function \( F(m, k_{[i..i+l]}) \)
- Acquire several curves with variety of plaintexts (but the same key)
- Make an hypothesis about the \([i..i + l]\) bits of the key
- Sort all curves according to this hypothesis in 2 sets
- Compute the difference of the means of both sets
- If you have a peak, hypothesis is true
Differential power analysis
Differential power analysis
Overview

1. The double-and-add algorithm
   - Universality
   - Importance

2. Non-invasive attacks
   - Timing attacks
   - Power analysis

3. Perturbation attacks
   - Fault Model
   - Naive attack
   - During the computation
Fault Model

Generally, in smart cards, we consider that we can:

- Inject 0x00
- Inject 0xFF
- Inject 0x??
Fault Model - How?

- Modifying VCC

![Graph showing the relationship between voltage and single bit faults/faulty multiplications]
Fault Model - How?

- Modifying VCC
- With a laser
Fault Model - In the case of the double-and-add

- Modification of a register
- $\Rightarrow$ impact on one of the coordinates
- Bit flip
Naive attack - Reminder: add formula

$K$ a finite field, $(a_1, a_2, a_3, a_4, a_6) \in K$

$E(K) = \{(x, y) \in K, y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6\} \cup \{O\}$

$$\begin{cases} x_3 = \lambda^2 + a_1 \lambda - a_2 - x_1 - x_2 \\ y_3 = -y_1 - (x_3 - x_1) \lambda - a_1 x_3 - a_3 \end{cases}$$

with

$$\lambda \left\{ \begin{array}{cl} \frac{3x_1^2 + 2a_2 x_1 + a_4 - a_1 y_1}{2y_1 + a_1 x_1 + a_3} & \text{if } x_1 = x_2 \text{ and } y_1 = y_2 \\ \frac{y_2 - y_1}{x_1 - x_2} & \text{otherwise} \end{array} \right.$$
Naive attack - Remarks

- Same with $P'_1 \notin E(K) \Rightarrow$ result not in $E(K)$
- Does not depend on $a_6 \Rightarrow$ same addition on $E'(K)$
Naive attack: malicious input $P'$

**Idea:** Move the DLP problem (of finding $d$ from $P$ and $[d]P$) from $E$ to $E'$

1. Find $E'$ of equation $y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a'_6$ of cardinality with a small factor
2. Find $P' \in E'(K)$ of small order
3. Give to double-and-add $P'$ and get back $Q'$
4. Solve DLP in $\langle P' \rangle$
During the computation

**Idea:** Inject fault during the $i^{th}$ round of the **double-and-add** algorithm

⇒ From $Q = [d]P$ and $Q'$ (result of the computation with fault), we deduce the last bits of $d$. 
During the computation

Notations:

- $Q^{(i)}$: value of $Q$ at the beginning of the $i^{th}$ round of the computation
- $Q'^{(i)}$: value of $Q$ at the beginning of the $i^{th}$ round of the computation, after the fault injection
- $d[i : j]$ design the $i$ to $j$ bits of $d$ written in Big-Endian
During the computation

Goals :

- Retrieve $Q'(i)$
- Retrieve $d[i:]$

To do that :

- Iteration on $i \leq j \leq n - 1$ : first bit of $d[i:]$ to be 1
- Iteration on all possible values of $d[j:]$ denoted $\times$
- Iteration on all possible values of $Q'(i)$
During the computation

Algorithm template:

\[
\text{for } j \text{ in } i..n-1 \text{ do} \\
\quad \text{for } x \text{ in } 2^j .. 2^{n-1} \text{ do} \\
\qquad \text{for } R \text{ mutation of } Q'_x^{(i)} \text{ do} \\
\qquad 
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{[...]} \; ; \\
\qquad \quad \text{end} \\
\quad \text{end} \\
\text{end}
\]
During the computation

Find possible values of $Q'(i) \Rightarrow$ reverse the last $n - 1 - i$ rounds of double-and-add algorithm
During the computation

Data: $Q = [d]P$ : resulting point, $d$ : secret key, $P$ : base point, $k$ : number of iterations to reverse

Result: $R$ : intermediate point

$R \leftarrow Q$;

$l \leftarrow \log_2(d)$;

for $i$ from 0 downto $k$ do

  if $d[l - i] == 1$ then
    /* Using point adding */
    $R \leftarrow R - P$;
  end

  /* POINT HALVING */
  $R \leftarrow Q/2$;

end

return $R$

Algorithme 4 : Basic Double-and-Add algorithm
During the computation

Then, from $Q'_x(i)$, we:

- compute all possible 1-bit mutations of $Q'_x(i)$ denoted $R$
- compute $Q'' = [x]R$
- compare $Q''$ with $Q'$
During the computation

Complexity: **polynomial**
(if we retrieve $d$ bit by bit from the LSB)
DEMO
Conclusion

- These attacks are efficient and not difficult to set up
- All countermeasures increase the cost of the algorithm
- Effectiveness depend on the hardware precision of the attacker
Thank you for your attention.