Elliptic Curve Scalar Multiplication, Side-Channel Attacks and Counter-measures

Jean-Marc ROBERT

Team DALI/LIRMM, Université de Perpignan, France

28 November 2014

Work funded by ANR PAVOIS 12 BS02 002 02 project

Outline

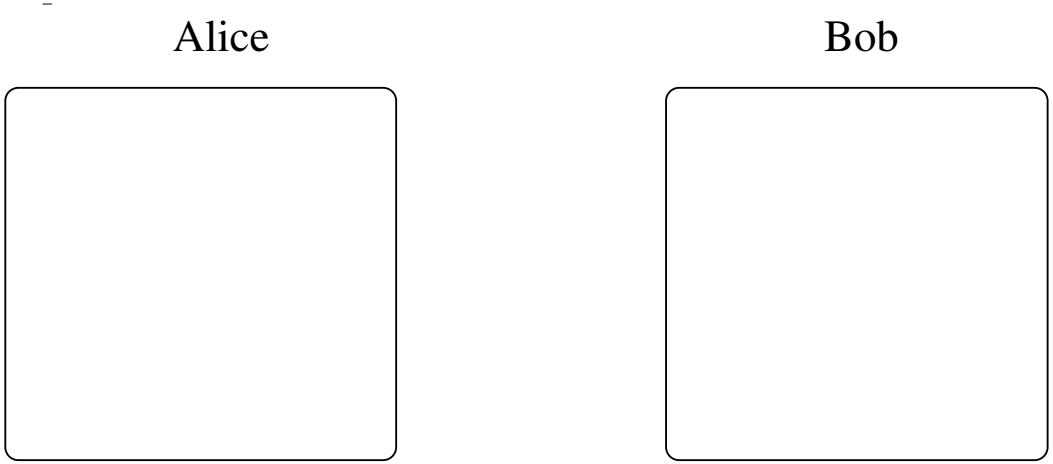
1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

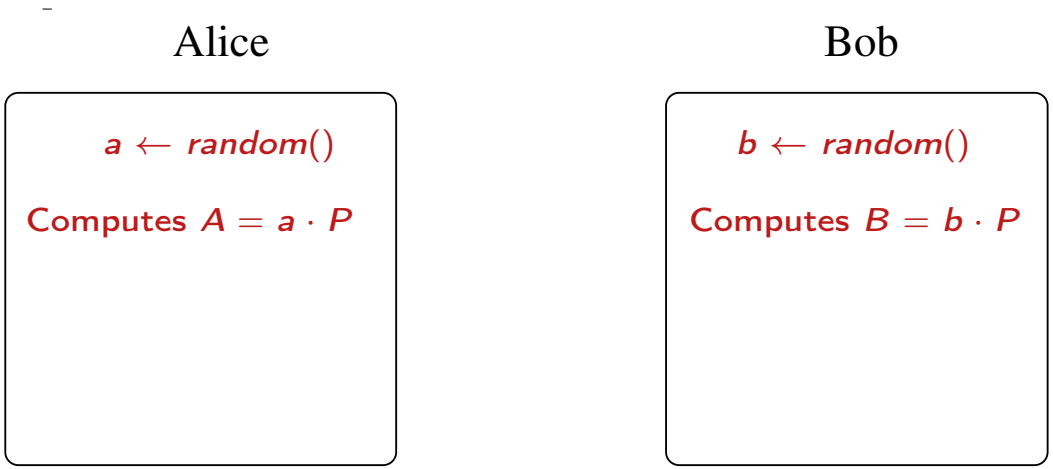
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group $(G, +, \mathcal{O})$ and a group generator P.



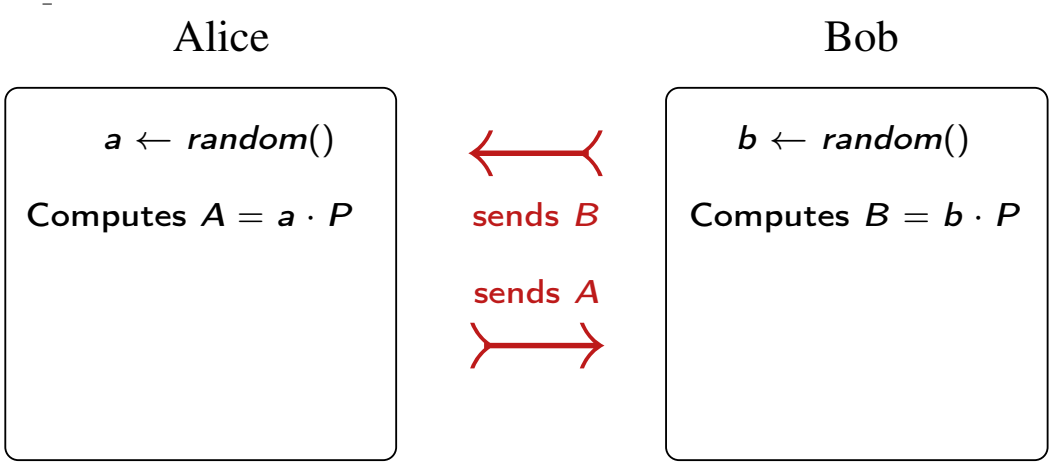
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group $(G, +, \mathcal{O})$ and a group generator P.



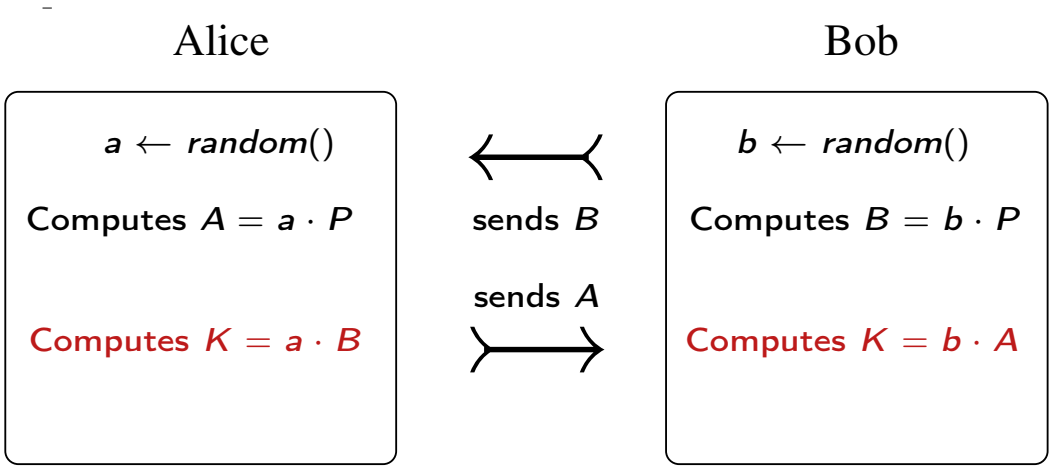
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group $(G, +, \mathcal{O})$ and a group generator P.



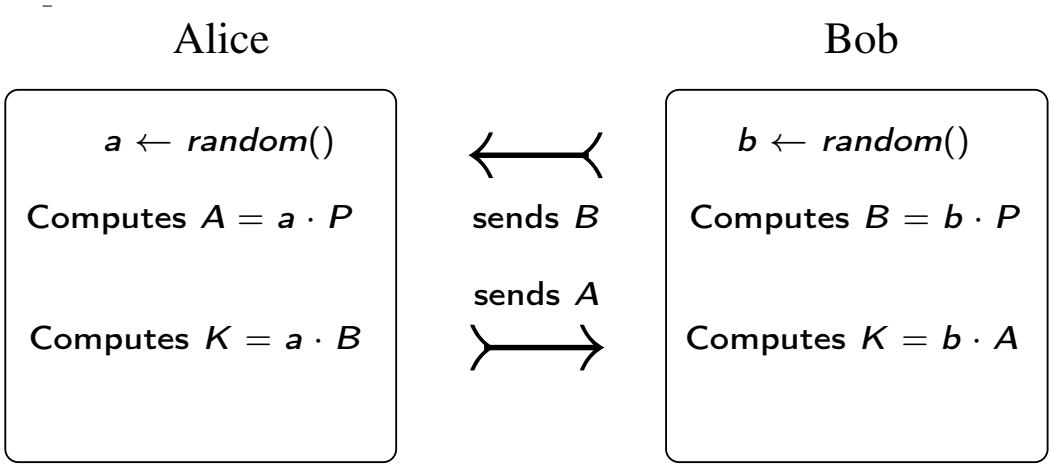
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group $(G, +, \mathcal{O})$ and a group generator P.



Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group $(G, +, \mathcal{O})$ and a group generator P.



Shared Secret Key $K = a \cdot b \cdot P$

$\rightarrow$The main operation is the scalar multiplication $a \cdot P$.

Multiplicative groups vs Abelian (additive) groups

	Multiplicative group $(\mathcal{G}, \times, 1)$	Abelian group $(\mathcal{G}, +, \mathcal{O})$
Group operation	$a \times b$	$P + Q$
	a^2	$[2] \cdot P$
Neutral element	1	0 or $\mathcal{O}$
	a^e exponentiation	$[k] \cdot P$ scalar multiplication
Discrete logarithm problem	knowing X and a, find e such as $X = a^e$	knowing X and P, find k such as $X = [k] \cdot P$

Multiplicative groups vs Abelian (additive) groups

	Multiplicative group $(\mathcal{G}, \times, 1)$	Abelian group $(\mathcal{G}, +, \mathcal{O})$
Group operation	$a \times b$	$P + Q$
	a^2	$[2] \cdot P$
Neutral element	1	0 or $\mathcal{O}$
	a^e exponentiation	$[k] \cdot P$ scalar multiplication
Discrete logarithm problem	knowing X and a, find e such as $X = a^e$	knowing X and P, find k such as $X = [k] \cdot P$

Example: El-Gamal encryption		
Alice's private key	x	
Alice's public key	$h \leftarrow g^x$ $(\mathcal{G}, g, h)$	$H \leftarrow [x] \cdot P$ $(\mathcal{G}, P, H)$
Bob's encryption (c_1, c_2)	y = Bob's secret parameter	
	$c_1 \leftarrow g^y, s \leftarrow h^y$ $c_2 \leftarrow m \cdot s$	$c_1 \leftarrow [y] \cdot P, S \leftarrow [y] \cdot H$ $c_2 \leftarrow m + S$
Alice's decryption	$s \leftarrow c_1^x$ $m' \leftarrow c_2 \cdot s^{-1}$ $(m' \leftarrow m \cdot s \cdot s^{-1})$	$S \leftarrow [x] \cdot c_1$ $m' \leftarrow c_2 - S$ $(m' \leftarrow m + S - S)$

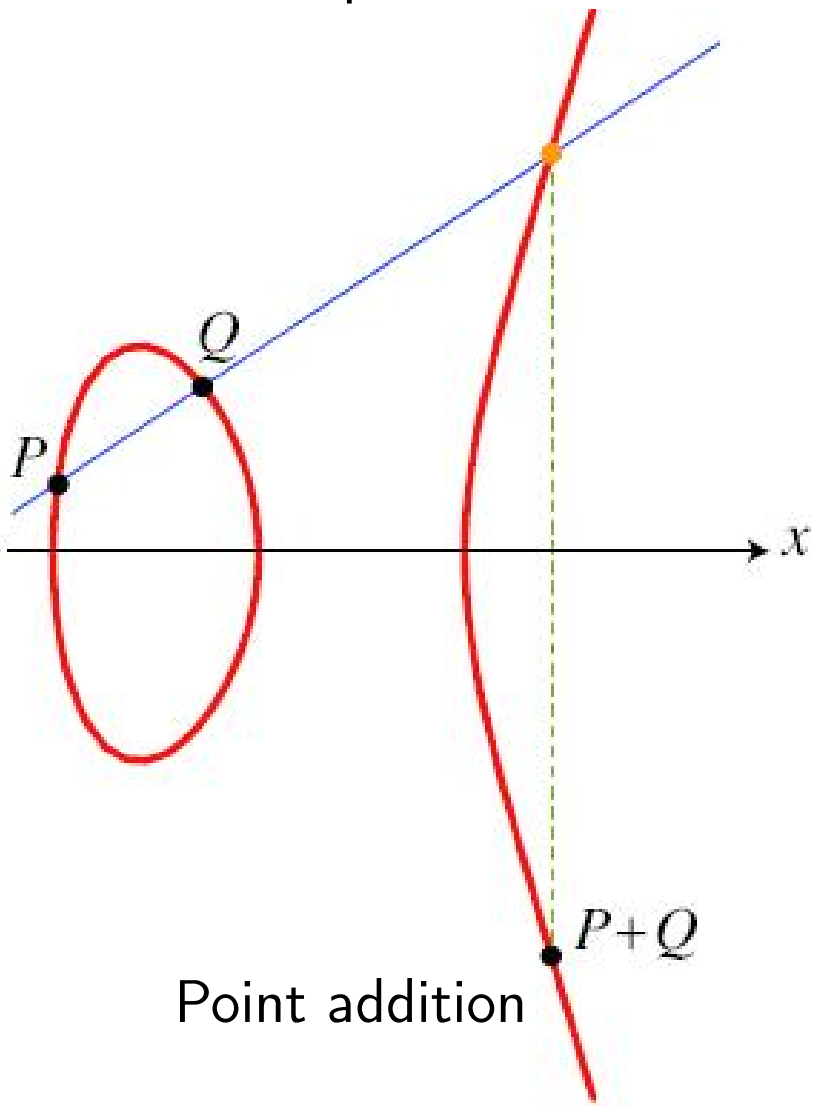
ECC vs Exponentiation over $\mathbb{F}_p$

Date	Minimum of Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
2010 (Legacy)	80	2TDEA*	1024	160	1024	160	SHA-1** SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
2011 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
> 2030	128	AES-128	3072	256	3072	256	SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
>> 2030	192	AES-192	7680	384	7680	384	SHA-384 SHA-512	SHA-224 SHA-256 SHA-384 SHA-512
>>> 2030	256	AES-256	15360	512	15360	512	SHA-512	SHA-256 SHA-384 SHA-512

NIST Recommendations (2012, from http://www.keylength.com/en/4/).

Our group: the set of Elliptic Curve points

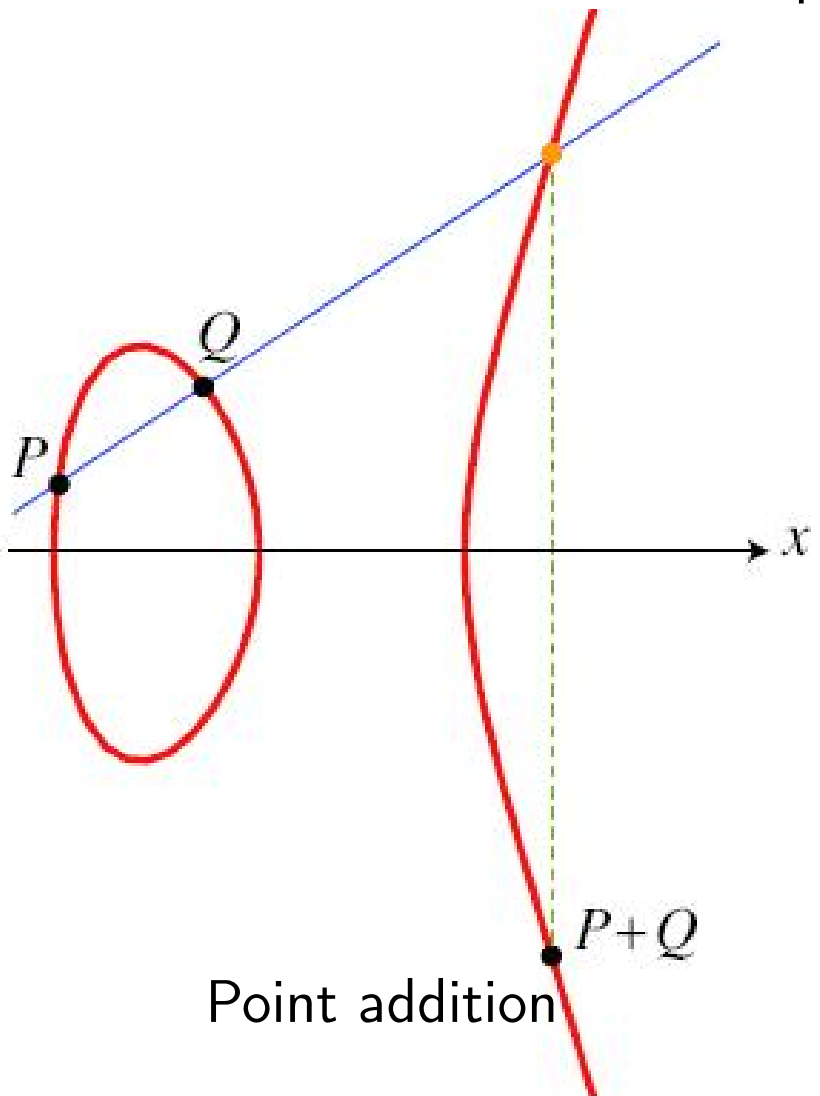
Example over $\mathbb{R}$:



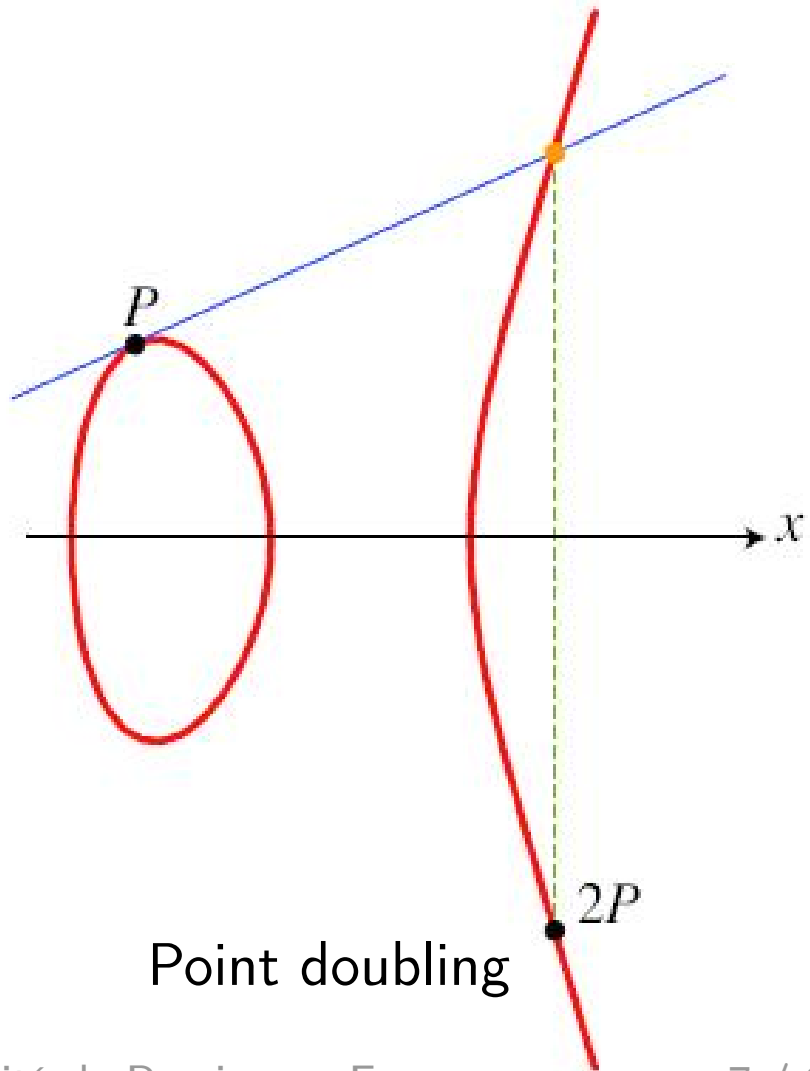
Point addition

Our group: the set of Elliptic Curve points

Example over $\mathbb{R}$:



Point addition



Point doubling

Our group: the set of Elliptic Curve points

Our curve is over a finite field $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$ (instead of $\mathbb{R}$):
$E : Y^2 = X^3 + aX + b, \quad a, b \in \mathbb{F}_p.$
$E : Y^2 + XY = X^3 + aX^2 + b, \quad a, b \in \mathbb{F}_{2^m}.$

- Finite field operations

Operation	$A, B \in \mathbb{F}_p$	$A, B \in \mathbb{F}_{2^m}$ $(= \mathbb{F}_2[x]/(f(x) \cdot \mathbb{F}_2[x]))$
Field element	$0 \leq A < p$ $0 \leq B < p$ p large prime	$A = \sum_{i=0}^{m-1} a_i \cdot x^i$ $B = \sum_{i=0}^{m-1} b_i \cdot x^i$ $a_i, b_i \in \{0, 1\}$
Addition $A + B =$	$A + B \mod p$	$\sum_{i=0}^{m-1} (a_i + b_i) \cdot x^i$
Multiplication $A \times B =$	$A \times B \mod p$	$A \cdot B \mod f$

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

Elliptic Curve Point Operations

Doubling and addition formulas: $P_1 + P_2 = P_3$
(with $P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3)$.)

$$\begin{array}{c|c} \mathbb{F}_p & \mathbb{F}_{2^m} \\ \left\{ \begin{array}{l} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = (x_1 - x_3)\lambda - y_1 \end{array} \right. & \begin{array}{l} x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 = (x_1 + x_3)\lambda + x_3 + y_1 \end{array} \end{array}$$

with

$$\left\{ \begin{array}{l|l} \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2 & \lambda = \frac{y_1 + y_2}{x_1 + x_2} \text{ if } P_1 \neq P_2 \\ \\ \lambda = \frac{3x_1^2 + a}{2y_1} \text{ if } P_1 = P_2 & \lambda = \frac{y_1}{x_1} + x_1 \text{ if } P_1 = P_2 \end{array} \right.$$

Elliptic Curve Point Operations

Doubling and addition formulas: $P_1 + P_2 = P_3$
(with $P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3)$.)

$$\begin{array}{l|l} \mathbb{F}_p & \mathbb{F}_{2^m} \\ \begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = (x_1 - x_3)\lambda - y_1 \end{cases} & \begin{cases} x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 = (x_1 + x_3)\lambda + x_3 + y_1 \end{cases} \end{array}$$

with

$$\begin{cases} \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2 \\ \lambda = \frac{3x_1^2 + a}{2y_1} \text{ if } P_1 = P_2 \end{cases} \quad \begin{cases} \lambda = \frac{y_1 + y_2}{x_1 + x_2} \text{ if } P_1 \neq P_2 \\ \lambda = \frac{y_1}{x_1} + x_1 \text{ if } P_1 = P_2 \end{cases}$$

- One doubling requires 1 inversion, 2 multiplications, 1 or 2 squaring(s), and some additions;
- One addition requires 1 inversion, 2 multiplications, 1 squaring, and some additions;

Elliptic curve point representation: Projective Coordinate Systems

- In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.
  Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

$$P = (x, y) \text{ is transformed into } (X : Y : Z) \text{ with } \begin{cases} x = \frac{X}{Z} \\ y = \frac{Y}{Z^2} \end{cases}$$

Elliptic curve point representation: Projective Coordinate Systems

- In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.
  Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

$$P = (x, y) \text{ is transformed into } (X : Y : Z) \text{ with } \begin{cases} x = \frac{X}{Z} \\ y = \frac{Y}{Z^2} \end{cases}$$

- Now, the doubling is computed as follows:

$$2.(X : Y : Z) = (X_1 : Y_1 : Z_1) \text{ with } \begin{cases} X_1 = X^4 + b \cdot Z^4 \\ Y_1 = bZ^4 \cdot Z_1 + X_1 \cdot (aZ_1 + Y^2 + bZ^4) \\ Z_1 = X^2 \cdot Z^2 \end{cases}$$

Elliptic curve point representation: Projective Coordinate Systems

- In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.
  Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

$$P = (x, y) \text{ is transformed into } (X : Y : Z) \text{ with } \begin{cases} x = \frac{X}{Z} \\ y = \frac{Y}{Z^2} \end{cases}$$

- Now, the doubling is computed as follows:

$$2.(X : Y : Z) = (X_1 : Y_1 : Z_1) \text{ with } \begin{cases} X_1 = X^4 + b \cdot Z^4 \\ Y_1 = bZ^4 \cdot Z_1 + X_1 \cdot (aZ_1 + Y^2 + bZ^4) \\ Z_1 = X^2 \cdot Z^2 \end{cases}$$

Operation	Affine	Lopez-Dahab
Doubling	2M + 1S + 1I	4M + 4S
Addition	2M + 1S + 1I	13M + 4S
Addition (mixed coordinates)	-	9M + 5S (Projective + Affine)

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

ECSM algorithm : *Double-and-add*

- Let $k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}, P \in E(\mathbb{F}_{2^m})$

$$\begin{aligned} k \cdot P &= (\sum_{i=0}^{t-1} 2^i k_i) \cdot P \\ &= ((\ldots((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P \\ &= ((\ldots((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P) \end{aligned}$$

ECSM algorithm : *Double-and-add*

- Let $k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}, P \in E(\mathbb{F}_{2^m})$

$$\begin{aligned} k \cdot P &= (\sum_{i=0}^{t-1} 2^i k_i) \cdot P \\ &= ((\ldots((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P \\ &= ((\ldots((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P) \end{aligned}$$

- This is the following algorithm:

Left-to-Right double-and-add Elliptic Curve Scalar Multiplication (ECSM)

Require: $k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m})$
Ensure: $Q = k \cdot P$
1: $Q \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q \leftarrow 2 \cdot Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** (Q)

ECSM algorithm : *Double-and-add*

- Let $k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}, P \in E(\mathbb{F}_{2^m})$

$$\begin{aligned} k \cdot P &= (\sum_{i=0}^{t-1} 2^i k_i) \cdot P \\ &= ((\ldots((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P \\ &= ((\ldots((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P) \end{aligned}$$

- This is the following algorithm:

"mixed coordinate addition" $\Rightarrow$

Left-to-Right double-and-add Elliptic Curve Scalar Multiplication (ECSM)

Require: $k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m})$
Ensure: $Q = k \cdot P$
1: $Q \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q \leftarrow 2 \cdot Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** (Q)

Double-and-add improvement : *NAF* and *W-NAF*.

- *NAF* replaces the sequences of consecutive 1: let $k \in \mathbb{N}$ such as $k = 2^i - 1$, then

$$(k)_2 = \underbrace{111...1}_{i \text{ times}} \text{ and one has: } (k)_{NAF} = \underbrace{100...00 - 1}_{i+1 \text{ digits}}.$$

$\Rightarrow$ Average number of non zero digits: from $n/2$ down to $n/3$.

Double-and-add improvement : *NAF* and *W-NAF*.

- *NAF* replaces the sequences of consecutive 1: let $k \in \mathbb{N}$ such as $k = 2^i - 1$, then

$$(k)_2 = \underbrace{111...1}_{i \text{ times}} \text{ and one has: } (k)_{NAF} = \underbrace{100...00 - 1}_{i+1 \text{ digits}}.$$

  $\Rightarrow$ Average number of non zero digits: from $n/2$ down to $n/3$.

- *W-NAF* decreases even more the number of non zero digits: $n/(w+1)$ now by using:

$$\{-2^{w-1} + 1, ..., -5, -3, -1, 0, 1, 3, 5, ..., 2^{w-1} - 1\}.$$

- Complexity balance for the ECSM over $E(\mathbb{F}_p)$ or $E(\mathbb{F}_p)$:

	doubling number	additions number
Double-and-add	n	$n/2$
NAF Double-and-add	n	$n/3$
W-NAF Double-and-add	n	$n/(w+1) + 2^{w-2}$

State of the art

	Scalar multiplication	Curve	Security	processor	Method	Cycles
$E(\mathbb{F}_p)$	Hamburg [4]	Montgomery	126	i7 SB	Montg. ladder	153000
	Langley [7]	Curve25519	128	i7 SB	Montg. ladder	229000
	Bernstein [2, 1]	Curve25519	128	i7 SB	Montg. ladder	194000
	Longa *et al.* [8]	jac256189	128	2 Duo	WNAF D&A	337000
	Longa *et al.* [8]	ted256189	128	2 Duo	WNAF D&A	281000
$E(\mathbb{F}_{2^m})$	Nègre *et al.* [9]	B233	112	i7 SB	WNAF D-H&A	98000
	Taverne *et al.* [11]	B233	112	i7 SB	WNAF D-H&A	102000
	Nègre *et al.* [9]	B409	192	i7 SB	WNAF D-H&A	347000
	Taverne *et al.* [11]	B409	192	i7 SB	WNAF D-H&A	358000

State of the art

	Scalar multiplication	Curve	Security	processor	Method	Cycles
$E(\mathbb{F}_p)$	Hamburg [4]	Montgomery	126	i7 SB	Montg. ladder	153000
	Langley [7]	Curve25519	128	i7 SB	Montg. ladder	229000
	Bernstein [2, 1]	Curve25519	128	i7 SB	Montg. ladder	194000
	Longa *et al.* [8]	jac256189	128	2 Duo	WNAF D&A	337000
	Longa *et al.* [8]	ted256189	128	2 Duo	WNAF D&A	281000
$E(\mathbb{F}_{2^m})$	Nègre *et al.* [9]	B233	112	i7 SB	WNAF D-H&A	98000
	Taverne *et al.* [11]	B233	112	i7 SB	WNAF D-H&A	102000
	Nègre *et al.* [9]	B409	192	i7 SB	WNAF D-H&A	347000
	Taverne *et al.* [11]	B409	192	i7 SB	WNAF D-H&A	358000

This is several hundreds of time faster than the modular exponentiation over $\mathbb{Z}/p\mathbb{Z}$!

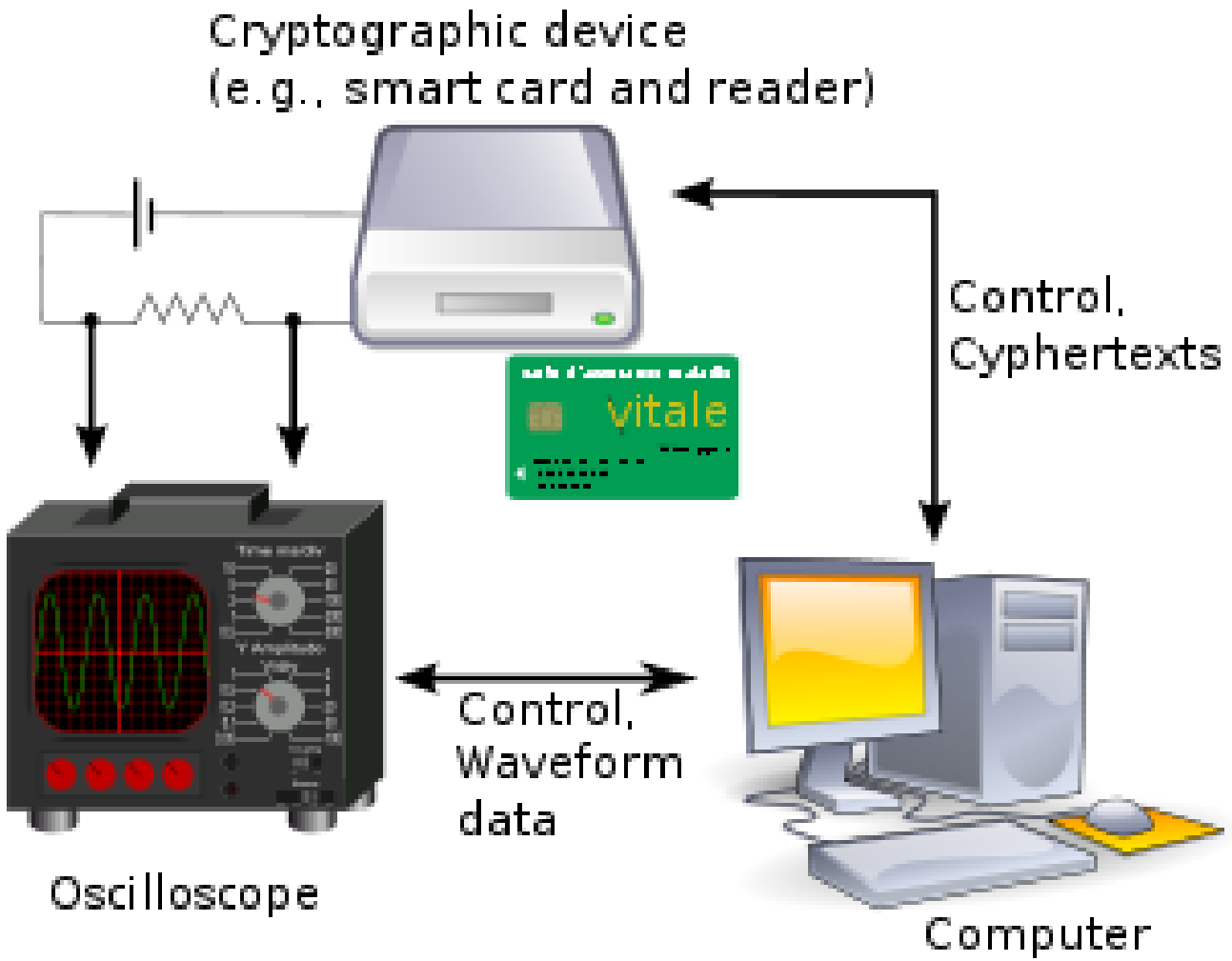
Outline

1 Problematic
- Cryptographic Protocols: what about the Group?
- Elliptic Curve Point Operations
- Elliptic Curve Scalar Multiplication

2 Side-channel Attacks
- First Attack: Simple Power Analysis
- How to thwart SPA?
- Second Attack: Differential Power Analysis
- How to thwart DPA?
- Synthesis

3 Conclusion

Side-Channel attack: What is it?

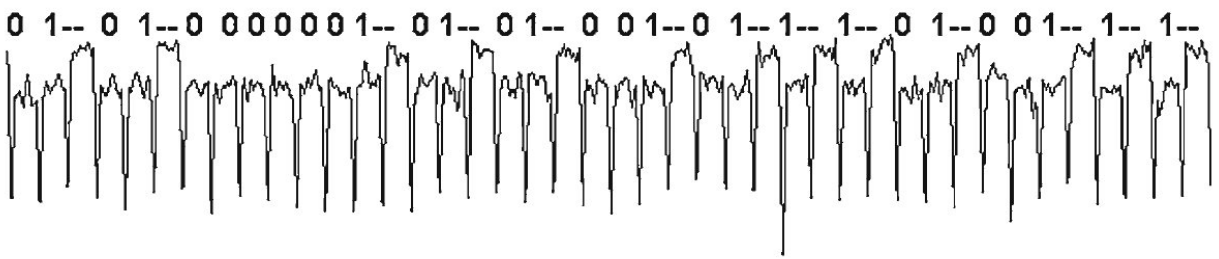


Side Channel test bench.

Vulnerable ECSM: Double-And-Add case

Left-to-Right double-and-add Elliptic Curve Scalar Multiplication (ECSM)

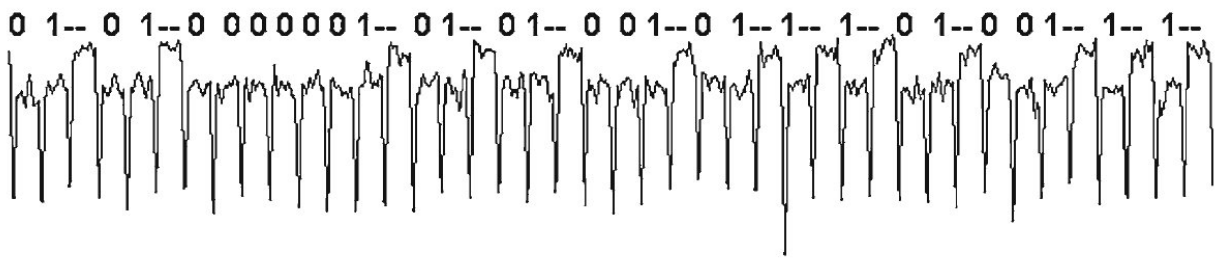
Require: $k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m})$
Ensure: $Q = k \cdot P$
1: $Q \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q \leftarrow 2 \cdot Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** (Q)



Vulnerable ECSM: Double-And-Add case

Left-to-Right double-and-add Elliptic Curve Scalar Multiplication (ECSM)

Require: $k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m})$
Ensure: $Q = k \cdot P$
1: $Q \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q \leftarrow 2 \cdot Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** (Q)



→ Vulnerable: the sequence of operations leaks the secret scalar (no regularity)

→ Simple Power Analysis

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

How to thwart SPA: first idea, Double-and-add-always

As a counter-measure, Coron in [3] suggested the following algorithm:

Double-and-add-always

Require: $k = (k_{t-1}, \ldots, k_1, k_0)$ with $k_{t-1} = 1, P \in E(\mathbb{F}_q)$
Ensure: $Q = k \cdot P$
1: $Q_0 \leftarrow \mathcal{O}, Q_1 \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q_0 \leftarrow 2 \cdot Q_0$
4: $\quad$ **if** $(k_i = 0)$ **then**
5: $\quad\quad Q_1 \leftarrow Q_0 + P$
6: $\quad$ **else**
7: $\quad\quad Q_0 \leftarrow Q_0 + P$
8: $\quad$ **end if**
9: **end for**
10: **return** (Q_0)

Double-and-add-always

Weak against Fault Injection Attacks (see Oviedo in [10]).

How to thwart SPA: first idea, Double-and-add-always

As a counter-measure, Coron in [3] suggested the following algorithm:

Double-and-add-always

Require: $k = (k_{t-1}, \ldots, k_1, k_0)$ with $k_{t-1} = 1, P \in E(\mathbb{F}_q)$
Ensure: $Q = k \cdot P$
1: $Q_0 \leftarrow \mathcal{O}, Q_1 \leftarrow \mathcal{O}$
2: **for** i from $t-1$ downto 0 **do**
3: $\quad Q_0 \leftarrow 2 \cdot Q_0$
4: $\quad$ **if** $(k_i = 0)$ **then**
5: $\quad\quad Q_1 \leftarrow Q_0 + P$ $\quad \Leftarrow$ "dummy addition"
6: $\quad$ **else**
7: $\quad\quad Q_0 \leftarrow Q_0 + P$
8: $\quad$ **end if**
9: **end for**
10: **return** (Q_0)

Double-and-add-always

Weak against Fault Injection Attacks (see Oviedo in [10]).

How to thwart SPA: Montgomery ladder

Montgomery

Require: $k = (k_{t-1}, \ldots, k_1, k_0)$ with $k_{t-1} = 1, P \in E(\mathbb{F}_q)$
Ensure: $Q = k \cdot P$

1: $Q_0 \leftarrow P, Q_1 \leftarrow 2P$
2: **for** i from $t-2$ downto 0 **do**
3: **if** $(k_i = 0)$ **then**
4: $Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
5: **else**
6: $Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
7: **end if**
8: **end for**
9: **return** (Q_0)

Basic Montgomery's ladder ECSM

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0 Q_1						

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0 Q_1	P					

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P					
Q_1	2P					

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P					
Q_1	2P	3P				

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$ $\Longleftarrow$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

Double:

$Q_0 \leftarrow 2 \cdot Q_0$

($Q_1 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$
therefore $Q_1 - Q_0 = P$)

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P				
Q_1	2P	3P				

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1,0,1,1,0,1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

$\Longleftarrow$

Double-and-add:

$Q_0 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$

$(Q_1 \leftarrow 2 \cdot (Q_0 + P)$
therefore $Q_1 - Q_0 = P)$

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P			
Q_1	2P	3P				

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P			
Q_1	2P	3P	6P			

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1,0,1,1,0,1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

$\Longleftarrow$

Double-and-add:

$Q_0 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$

$(Q_1 \leftarrow 2 \cdot (Q_0 + P)$
therefore $Q_1 - Q_0 = P)$

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P		
Q_1	2P	3P	6P			

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P		
Q_1	2P	3P	6P	12P		

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P		
Q_1	2P	3P	6P	12P	23P	

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$ $\Longleftarrow$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

Double:

$Q_0 \leftarrow 2 \cdot Q_0$

($Q_1 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$
therefore $Q_1 - Q_0 = P$)

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P	22P	
Q_1	2P	3P	6P	12P	23P	

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1,0,1,1,0,1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

$\Longleftarrow$

Double-and-add:

$Q_0 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$

$(Q_1 \leftarrow 2 \cdot (Q_0 + P)$
therefore $Q_1 - Q_0 = P)$

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P	22P	45P
Q_1	2P	3P	6P	12P	23P	

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P	22P	45P
Q_1	2P	3P	6P	12P	23P	46P

How does the Montgomery ladder work?

Example : $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: **if** $(k_i = 0)$ **then**
2: $\quad Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: **else**
4: $\quad Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: **end if**

i	init	4	3	2	1	0
k_i	1	0	1	1	0	1
Q_0	P	2P	5P	11P	22P	45P
Q_1	2P	3P	6P	12P	23P	46P

The algorithm returns:

$$\rightarrow Q_0 = 45P$$

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

Second Attack: *Differential Power Analysis*

These attacks are presented in Kocher *et al.* [5, 6]
The attacker uses a power consumption model of the device.
Let $\mathcal{H}(k)$ be the Hamming weight of the processed data Δ at time t.
Then, our model is:

$$\mathcal{P}(t) = \mathcal{C} \cdot \mathcal{H}(\Delta) + \epsilon + K$$

($\mathcal{C}$ and K are constants, ϵ represents the noise.)

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow 0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
0x56	0x1E3C	0x3CCE	0
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
0x56	0x1E3C	0x3CCE	0
0xC2	0x4434	0x892A	1
⋮	⋮	⋮	⋮

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
0x56	0x1E3C	0x3CCE	0
0xC2	0x4434	0x892A	1
⋮	⋮	⋮	⋮
500-1000 computations			

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ $0xB5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
0x56	0x1E3C	0x3CCE	0
0xC2	0x4434	0x892A	1
⋮	⋮	⋮	⋮
500-1000 computations			

over 16 bits,

- average Hamming weight of the "red ones": 8.5;
- average Hamming weight of the "black ones": 7.5.

Second Attack: *Differential Power Analysis*

A toy example: The attacker ask the device to compute a serie of multiplications of chosen $I[i]$ by a secret constant k over 12 bits (CPA).
Assume the attacker knows the seven first bits of k : $[1,0,1,1,1,0,0]_2 = 0x5A$.
The attacker select one bit of the intermediate value computed:

Chosen value ($I[i]$)	intermediate value loop round 6 (Q_0)	Assumption $k_i = 1$ first 8 bits $\rightarrow$ 0x$B5$ $(= (0x5A << 1) \wedge 0x1)$	selected bit: n°8
0x25	0xD02	0x1A29	0
0x13	0x6AE	0xD6F	1
0xB2	0x3E94	0x7DDA	1
0x56	0x1E3C	0x3CCE	0
0xC2	0x4434	0x892A	1
⋮	⋮	⋮	⋮
500-1000 computations			

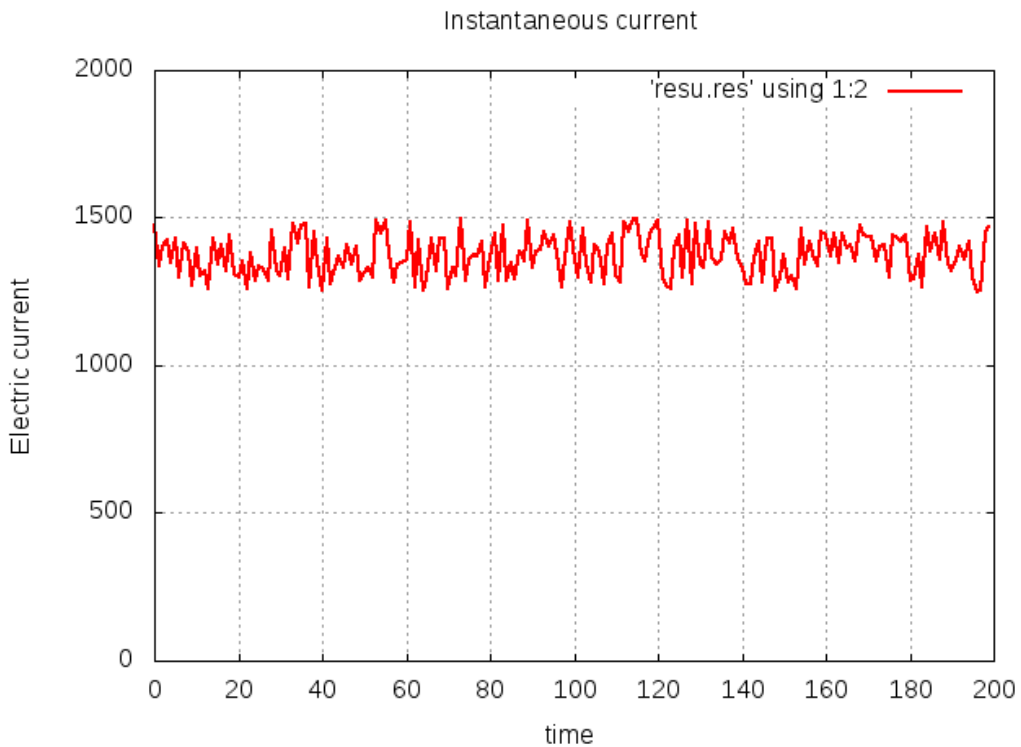
over 16 bits,

- average Hamming weight of the "red ones": 8.5;
- average Hamming weight of the "black ones": 7.5.

When the device processes the selected bit (n°8), if the assumption on k_i is correct, the "red ones" need more electric power then the "black ones".

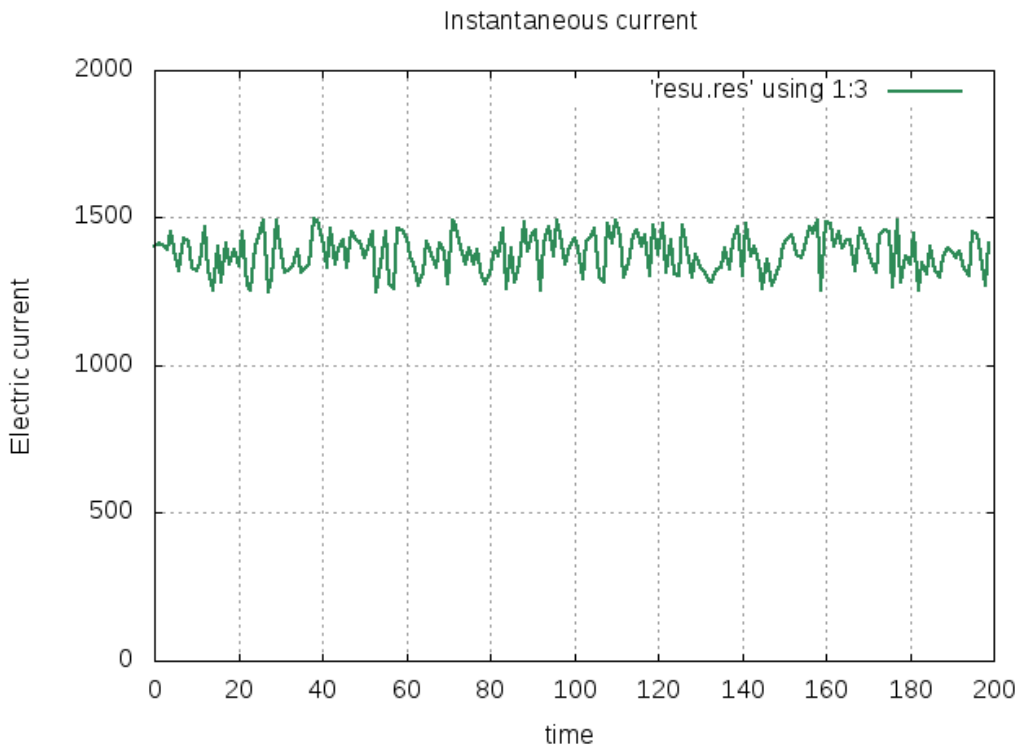
Second Attack: *Differential Power Analysis*

The attacker collects the m power traces of the m computations $\mathbf{T}_{1\ldots m}[j]$.
Power trace of a "red one":



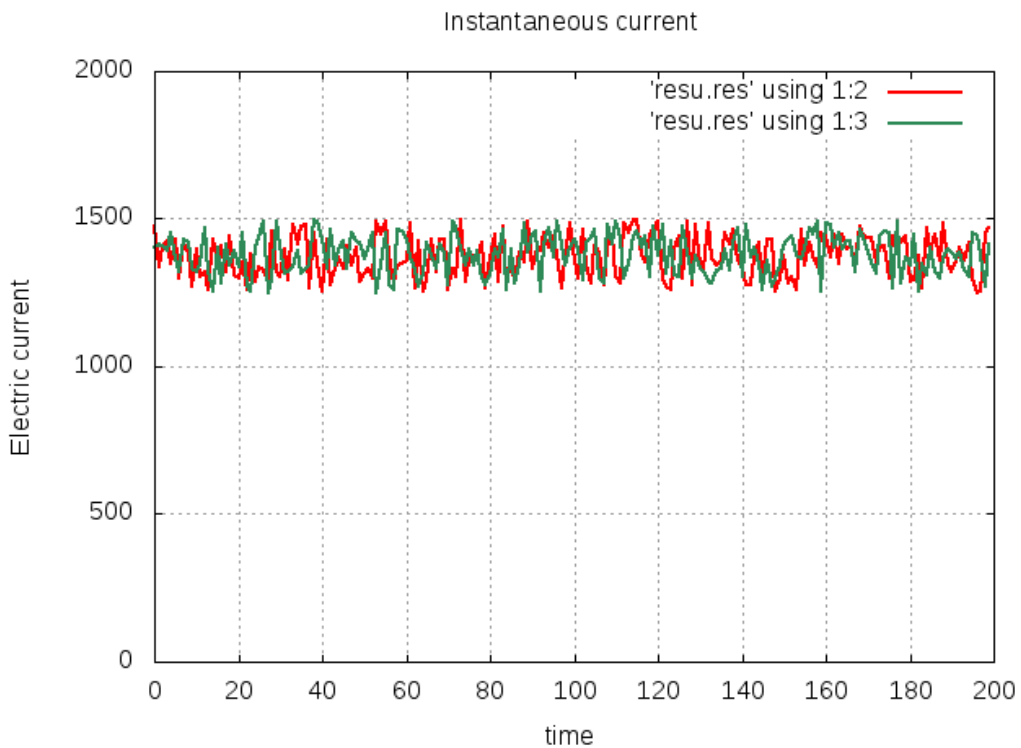
Second Attack: *Differential Power Analysis*

The attacker collects the m power traces of the m computations $\mathbf{T}_{1\ldots m}[j]$.
Power trace of a "black one":



Second Attack: *Differential Power Analysis*

Both power traces:



Second Attack: *Differential Power Analysis*

The attacker now computes the average trace for the red ones and the black ones, and the difference between both average traces:

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(P_i, s) \cdot \mathbf{T}_i[j]}{\sum_{i=1}^{m} D(P_i, s)} - \frac{\sum_{i=1}^{m} (1 - D(P_i, s)) \cdot \mathbf{T}_i[j]}{\sum_{i=1}^{m} (1 - D(P_i, s))}$$

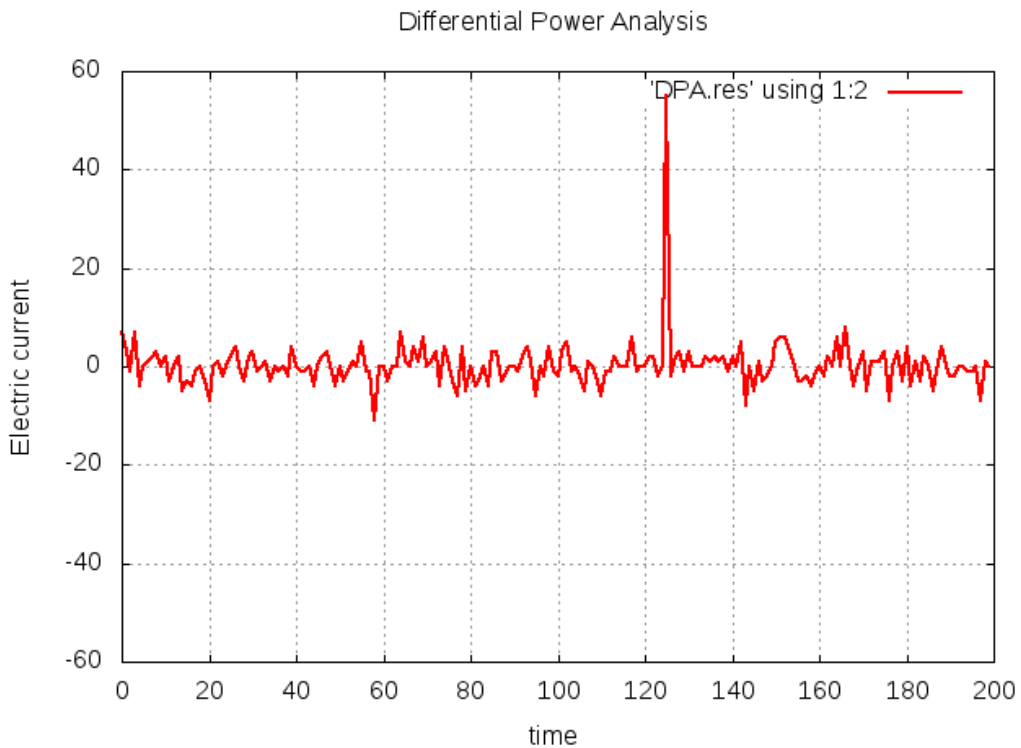
$$\approx 2 \cdot \left(\frac{\sum_{i=1}^{m} D(P_i, s) \cdot \mathbf{T}_i[j]}{\sum_{i=1}^{m} D(P_i, s)} - \frac{\sum_{i=1}^{m} \mathbf{T}_i[j]}{m} \right).$$

with:

- $D(P_i, s) =$ bit 8 of $0xB5 \cdot I[i]$
- $\mathbf{T}_{1\ldots m}[j]$

Second Attack: *Differential Power Analysis*

The attacker now computes the average trace for the red ones and the black ones, and the difference between both average traces:

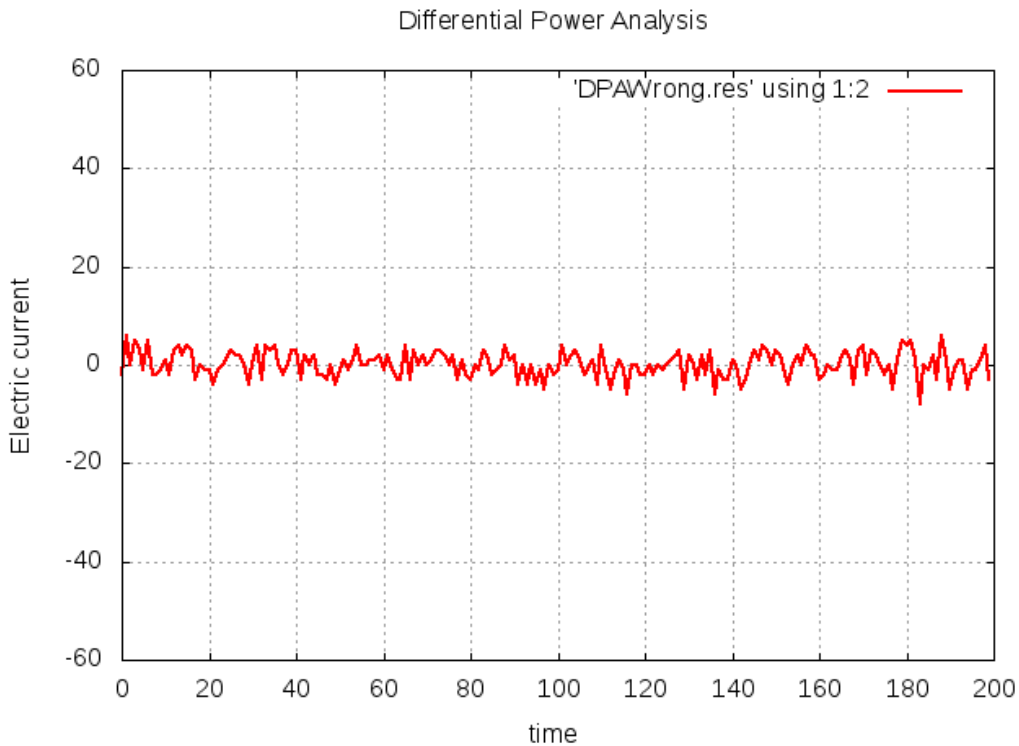


$\rightarrow$ At the time the selected bit is processed, one can see a peak: good guess!

(The eightth bit of the secret scalar is 1!)

Second Attack: *Differential Power Analysis*

We now do the same with the assumption of a bit equal to 0 (same computations as previously, $0xB4$ instead of $0xB5$):



$\rightarrow$ Only noise! Wrong guess!

(However, the attacker wins the game anyway: if the eightth bit of the secret scalar is not 0, it is 1 !)

DPA in pictures, Coron, CHES 99

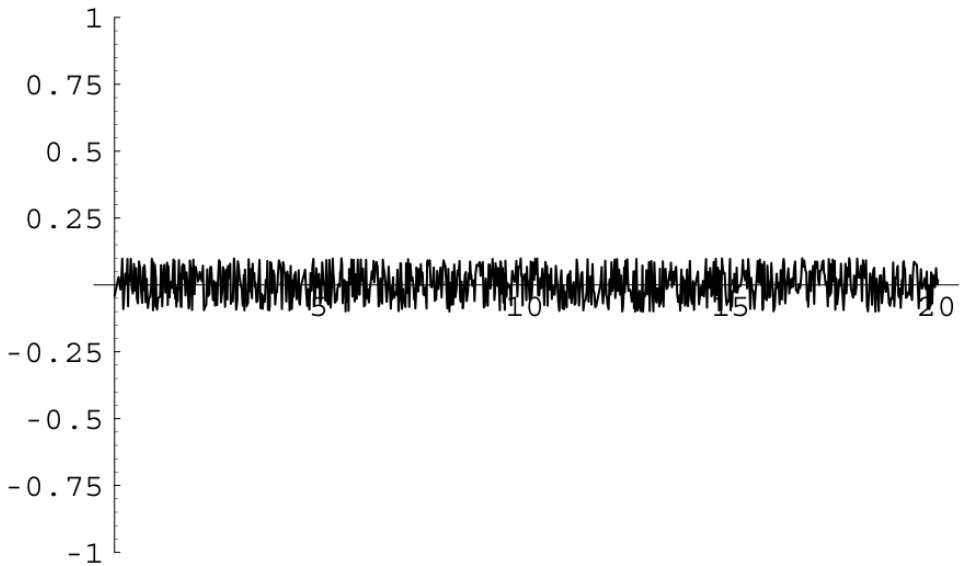


Fig. 2. Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{\ell-2} = 1$. No peak is observed since the points $4P_i$ are never computed by the card.

In this exemple quoted form Coron, a smart card computes an elliptic curve scalar multiplication $d \cdot P$ with the Left-to-right-Double-and-add algorithm, and the attacker guesses the second bit to be processed.

DPA in pictures, Coron, CHES 99

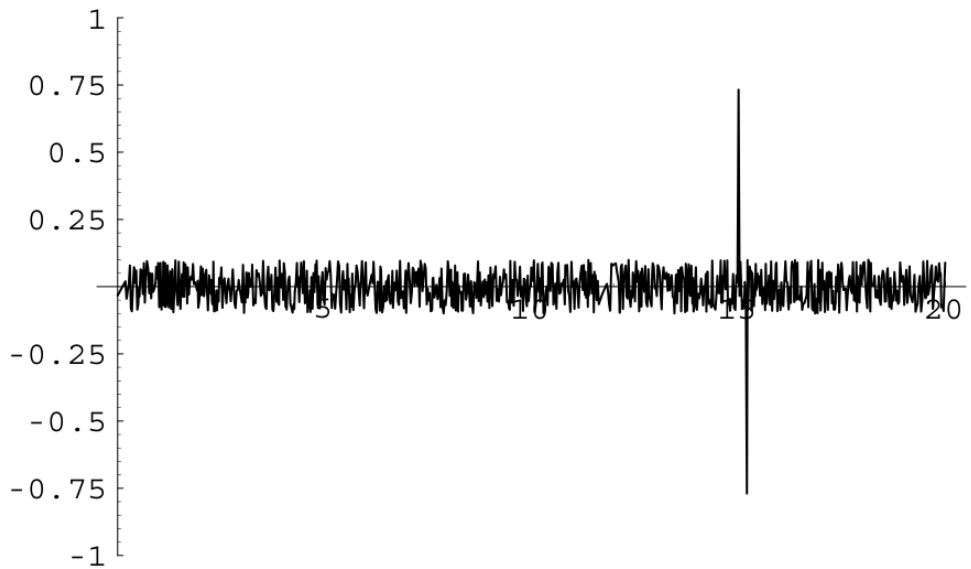


Fig. 1. Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{\ell-2} = 0$. A peak is observed corresponding to the computation of $4P_i$ inside the card.

In this exemple quoted form Coron, a smart card computes an elliptic curve scalar multiplication $d \cdot P$ with the Left-to-right-Double-and-add algorithm, and the attacker guesses the second bit to be processed.

Analysis of the algorithms: Double-and-Add-Always

Double-and-Add-Always									
round i:				$i = l-2$			$i = l-3$		
$k_{l-1} = 1$	k_{l-2}	k_{l-3}	...	Q_0	Q_1	Chosen result	Q_0	Q_1	Chosen result
1	0	0	...	$2P$	$3P$	$2P$	$4P$	$5P$	$4P$
1	0	1	...	$2P$	$3P$	$2P$	$4P$	$5P$	$5P$
1	1	0	...	$2P$	$3P$	$3P$	$6P$	$7P$	$6P$
1	1	1	...	$2P$	$3P$	$3P$	$6P$	$7P$	$7P$

Table : First loop rounds of Algorithm 19, *Double-and-Add-Always*

Analysis of the algorithms: Montgomery's Binary Ladder

Montgomery's binary ladder							
round i:				$i = l-2$		$i = l-3$	
$k_{l-1} = 1$	k_{l-2}	k_{l-3}	...	Q_0	Q_1	Q_0	Q_1
1	0	0	...	$2P$	$3P$	$4P$	$5P$
1	0	1	...	$2P$	$3P$	$5P$	$6P$
1	1	0	...	$3P$	$4P$	$6P$	$7P$
1	1	1	...	$3P$	$4P$	$7P$	$8P$

Table : First loop rounds of Algorithm 20, Montgomery's binary ladder

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*

An elliptic curve over $\mathbb{F}_{2^m}$ or $\mathbb{F}_p$ is a finite set of points, whose number of points is denoted $\#\mathcal{E}$. The order of a point divides $\#\mathcal{E}$.

$$Q = d \cdot P = (d + k \cdot \#\mathcal{E}) \cdot P, \forall k \in \mathbb{Z}.$$

(Indeed, one has: $\#\mathcal{E} \cdot P = \mathcal{O}$).

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*
- *Blinding the point P*

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*
- *Blinding the point P*

One adds a random point R to the base point P, whose multiple $S = d \cdot R$ is known in advance.
-> Point used in the operation $P' = P + R$
Variant:
-> $R \leftarrow (-1)^b 2R, \quad S \leftarrow (-1)^b 2S,$ (b is a random bit).

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*
- *Blinding the point P*
- *Randomized projective coordinates*

Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*
- *Blinding the point P*
- *Randomized projective coordinates*

For one point in affine coordinates (x, y), there is a huge quantity of projective points (X, Y, Z) corresponding such as $(x = X/Z, y = Y/Z^2)$.

Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication
2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis
3. Conclusion

The DPA attack:

- requires more power form the attacker (multiple Chosen Plaintext Attack and access to the device);

The DPA attack:

- requires more power form the attacker (multiple Chosen Plaintext Attack and access to the device);
- is also sensitive to the noise;
  - improvements by CPA, Template Attacks...
  - Hardware counter-measures;

The DPA attack:

- requires more power form the attacker (multiple Chosen Plaintext Attack and access to the device);
- is also sensitive to the noise;
  - improvements by CPA, Template Attacks...
  - Hardware counter-measures;
- Moreover, the counter-measures are costly.

Conclusion

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;
- DPA attack;

Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;
- DPA attack;
- Other attacks every day... New counter-measures...

Thank you for your attention,

Any questions ?

mail : jean-marc.robert@univ-perp.fr

home page : http://perso.univ-perp.fr/jeanmarc.robert


D. J. Bernstein.
Curve25519: New Diffie-Hellman Speed Records.
In *Public Key Cryptography*, pages 207–228, 2006.


D.J. Bernstein and Lange T. (eds).
eBACS: ECRYPT Benchmarking of Cryptograhic Systems.
`http://bench.cr.yp.to/`, 2012.
accessed May 25th, 14.


Jean-Sébastien Coron.
Resistance against differential power analysis for elliptic curve cryptosystems.
In *CHES*, pages 292–302, 1999.


Mike Hamburg.
Fast and compact elliptic-curve cryptography.
Cryptology ePrint Archive, Report 2012/309, 2012.
`http://eprint.iacr.org/`.


Paul C. Kocher, Joshua Jaffe, and Benjamin Jun.

Differential power analysis.
In *Advances in Cryptology, CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.


Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi.
Introduction to differential power analysis.
J. Cryptographic Engineering, 1(1):5–27, 2011.


A. Langley.
C25519 code.
http://code.google.com/p/curve25519-donna/, 2008.
`http://code.google.com/p/curve25519-donna/`.


P. Longa and C. H. Gebotys.
Efficient Techniques for High-Speed Elliptic Curve Cryptography.
In *CHES*, pages 80–94, 2010.


Christophe Nègre and Jean-Marc Robert.
Impact of optimized field operations ab, ac and $ab + cd$ in scalar multiplication over binary elliptic curve.

In *AFRICACRYPT*, pages 279–296, 2013.

Agustin Dominguez Oviedo.
On Fault-based Attacks and Countermeasures for Elliptic Curve Cryptosystems.
PhD thesis, 2008.

Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López.
Speeding scalar multiplication over binary elliptic curves using the new carry-less multiplication instruction.
J. Cryptographic Engineering, 1(3):187–199, 2011.