

Power Consumption Analysis and Hardware Security

Arnaud Tisserand

CNRS, Lab-STICC laboratory

Cergy, December 2017



Applications with Security Needs



Applications: smart cards, computers, Internet, telecommunications, set-top boxes, data storage, RFID tags, WSN, smart grids...

Cryptographic Features

Objectives:

- Confidentiality
- Integrity
- Authenticity
- Non-repudiation
- ...

Cryptographic primitives:

- Encryption
- Digital signature
- Hash function
- Random numbers generation
- ...

Implementation issues in hardware:

- **Performances**: speed, delay, throughput, latency
- **Cost**: device (memory, size, weight), low power/energy consumption, design
- **Security**: protection against physical attacks

Square and Multiply Algorithm for RSA

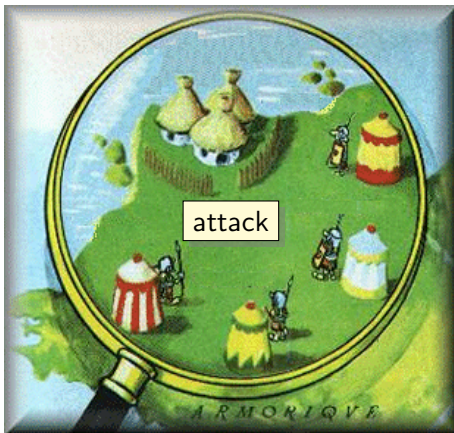
input: a, b, n where $b = (b_{t-1}b_{t-2} \dots b_1b_0)_2$

output: $a^b \bmod n$

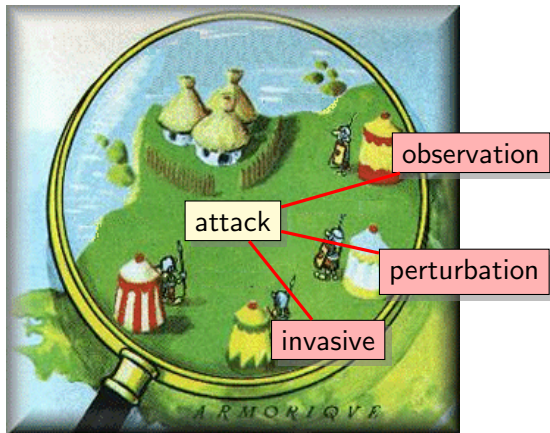
```
 $r = 1$ 
for  $i$  from 0 to  $t - 1$  do
    if  $b_i = 1$  then
         $r = r \cdot a \bmod n$ 
    endif
     $a = a^2 \bmod n$ 
endfor
return  $r$ 
```

This is the right to left version (there exists a left to right one)

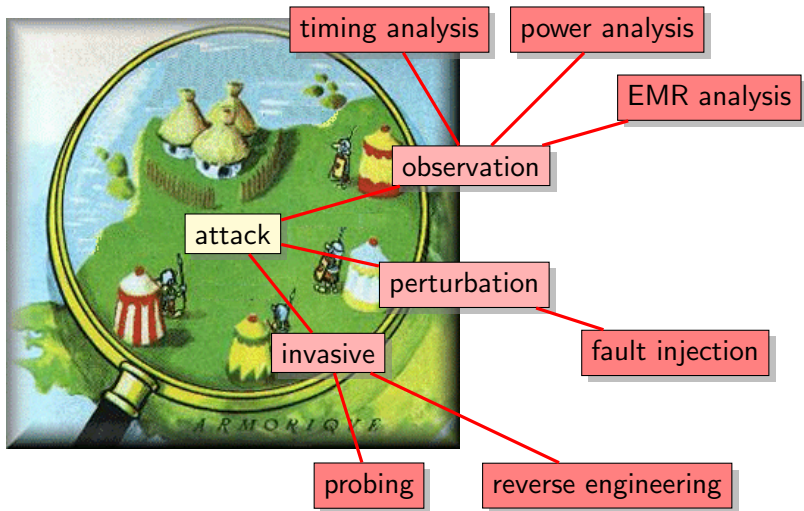
Attacks



Attacks

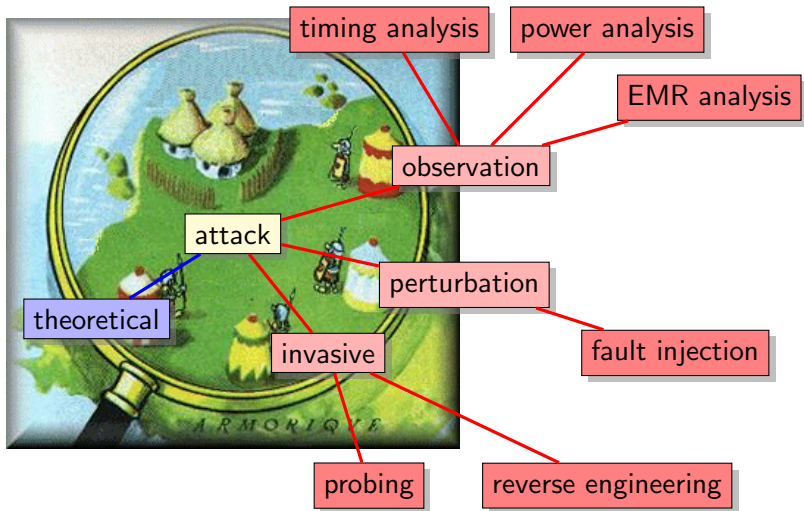


Attacks



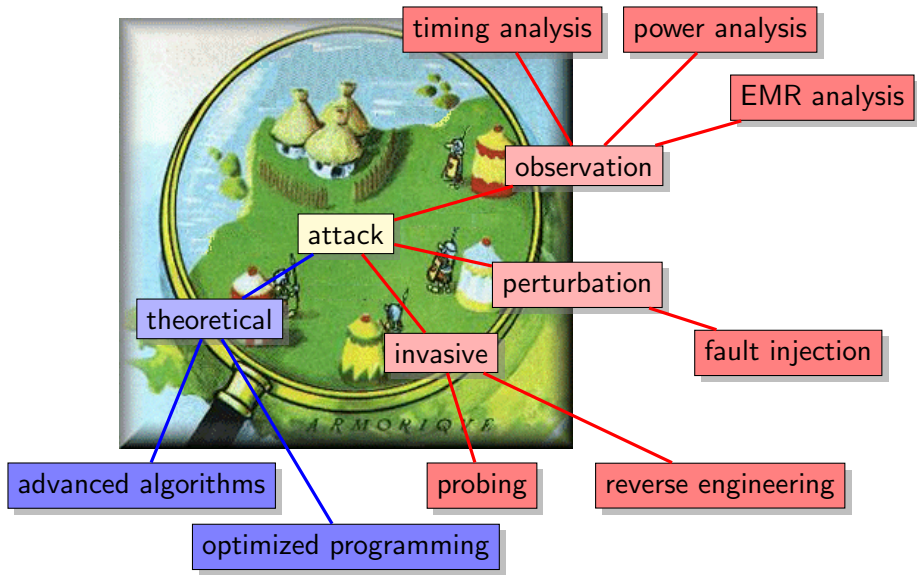
EMR = Electromagnetic radiation

Attacks



EMR = Electromagnetic radiation

Attacks



EMR = Electromagnetic radiation

Side Channel Attacks (SCAs) (1/2)

Attack: attempt to find, **without** any knowledge about the secret:

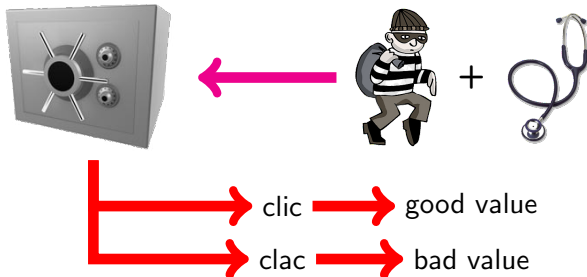
- the message (or parts of the message)
- informations on the message
- the secret (or parts of the secret)

Side Channel Attacks (SCAs) (1/2)

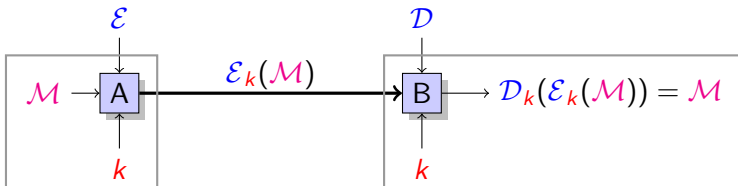
Attack: attempt to find, **without** any knowledge about the secret:

- the message (or parts of the message)
- informations on the message
- the secret (or parts of the secret)

“Old style” side channel attacks:

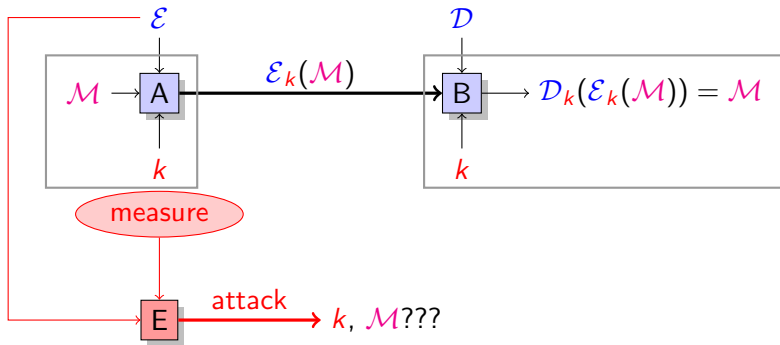


Side Channel Attacks (SCAs) (2/2)



General principle: measure **external parameter(s)** on running device in order to deduce **internal informations**

Side Channel Attacks (SCAs) (2/2)



General principle: measure external parameter(s) on running device in order to deduce internal informations

What Should be Measured?

Answer: **everything** that can “enter” and/or “get out” in/from the device

- power consumption
- electromagnetic radiation
- temperature
- sound
- computation time
- number of cache misses
- number and type of error messages
- ...

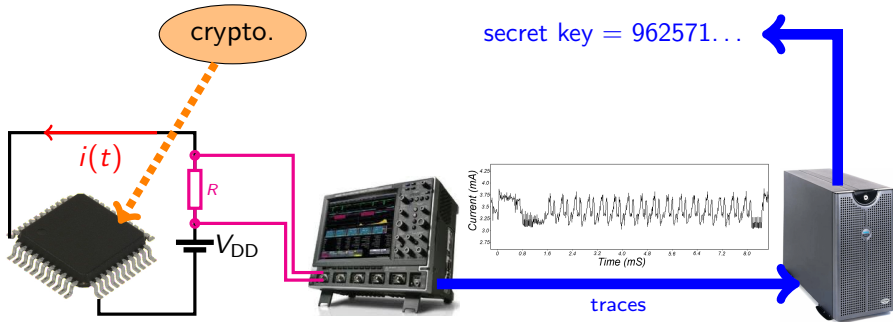
The measured parameters may provide informations on:

- **global** behavior (temperature, power, sound...)
- **local** behavior (EMR, # cache misses...)

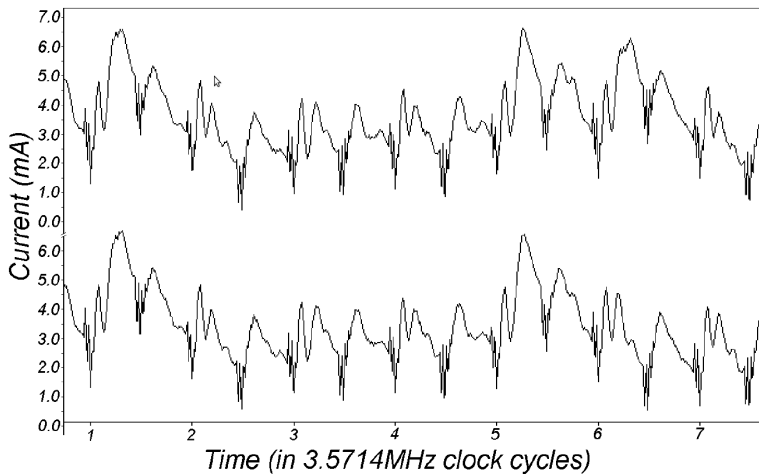
Power Consumption Analysis

General principle:

1. measure the current $i(t)$ in the cryptosystem
2. use those measurements to “deduce” secret informations

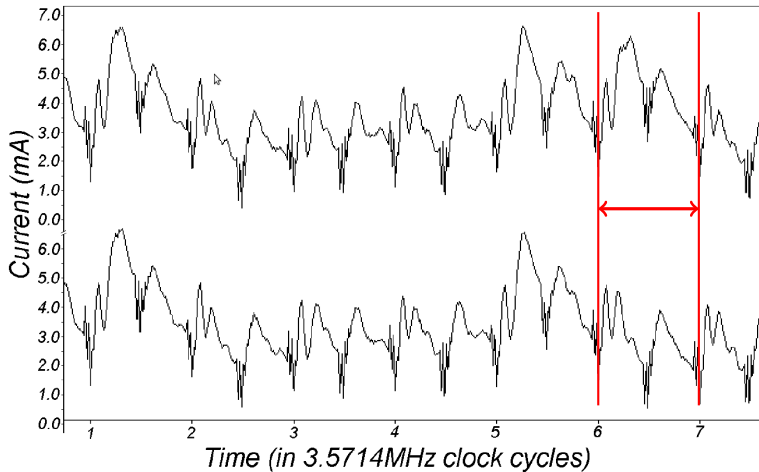


Simple Power Analysis (SPA)



Source: [4]

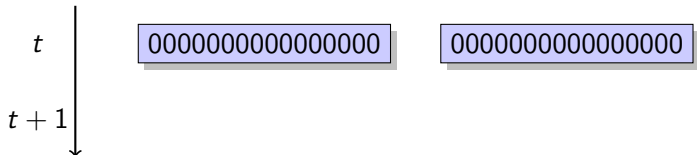
Simple Power Analysis (SPA)



Source: [4]

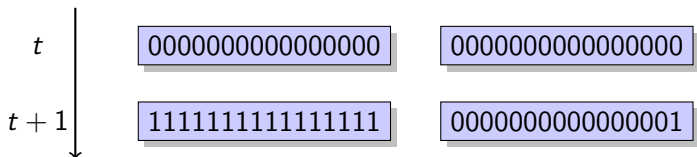
Limits of the SPA

Example of behavior difference: (activity into a register)



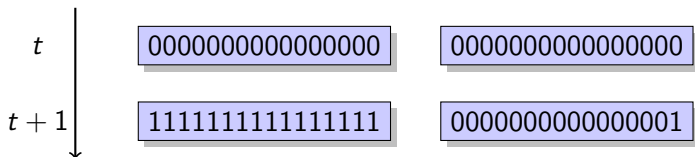
Limits of the SPA

Example of behavior difference: (activity into a register)



Limits of the SPA

Example of behavior difference: (activity into a register)

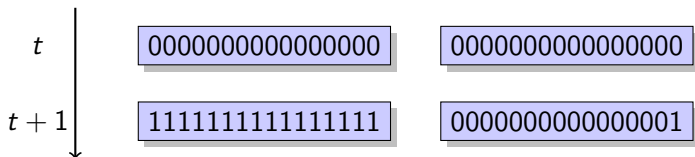


Important: a small difference may be evaluated has a **noise** during the measurement → traces cannot be distinguished

Question: what can be done when differences are too small?

Limits of the SPA

Example of behavior difference: (activity into a register)



Important: a small difference may be evaluated has a **noise** during the measurement → traces cannot be distinguished

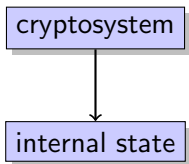
Question: what can be done when differences are too small?

Answer: use **statistics** over **several** traces

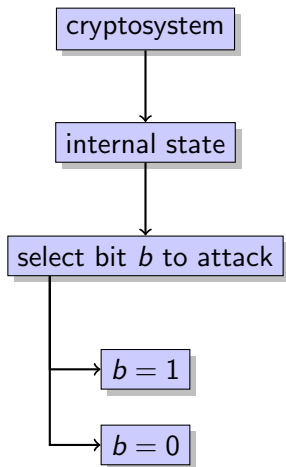
Differential Power Analysis (DPA)

cryptosystem

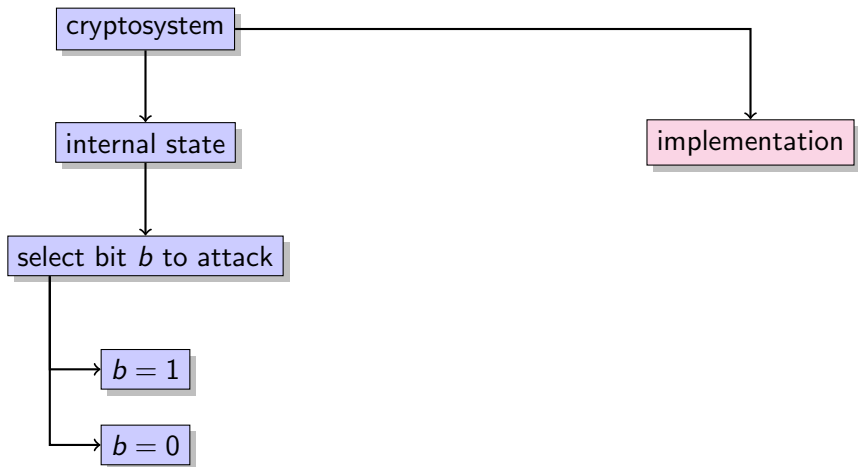
Differential Power Analysis (DPA)



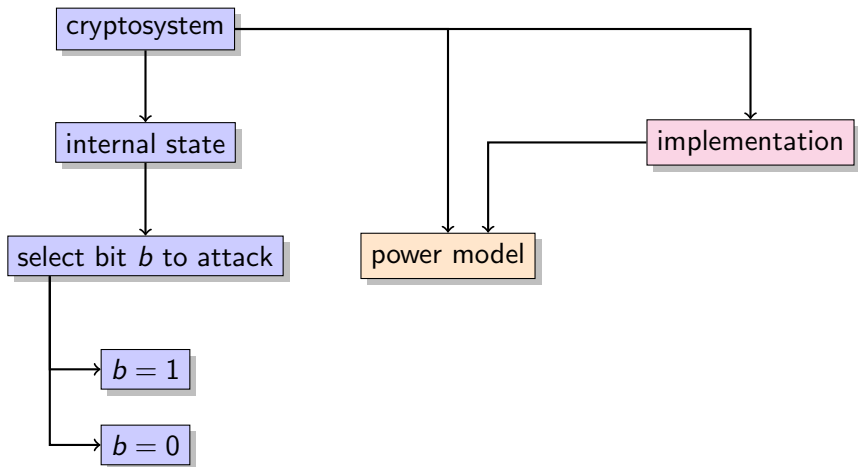
Differential Power Analysis (DPA)



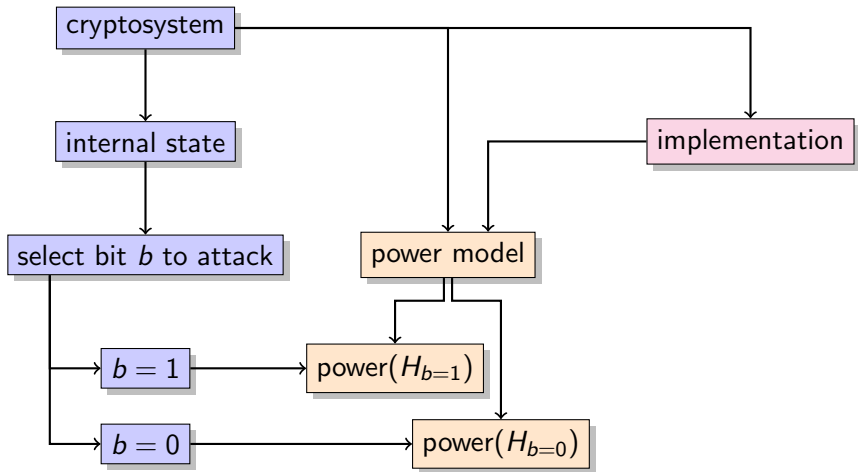
Differential Power Analysis (DPA)



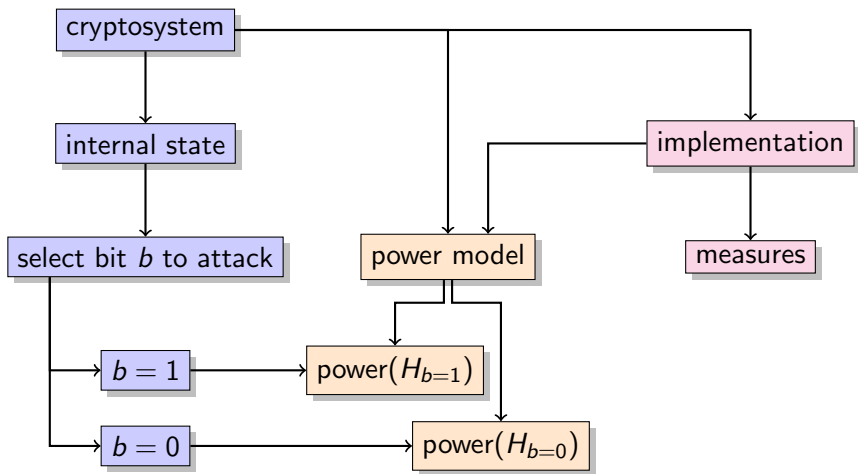
Differential Power Analysis (DPA)



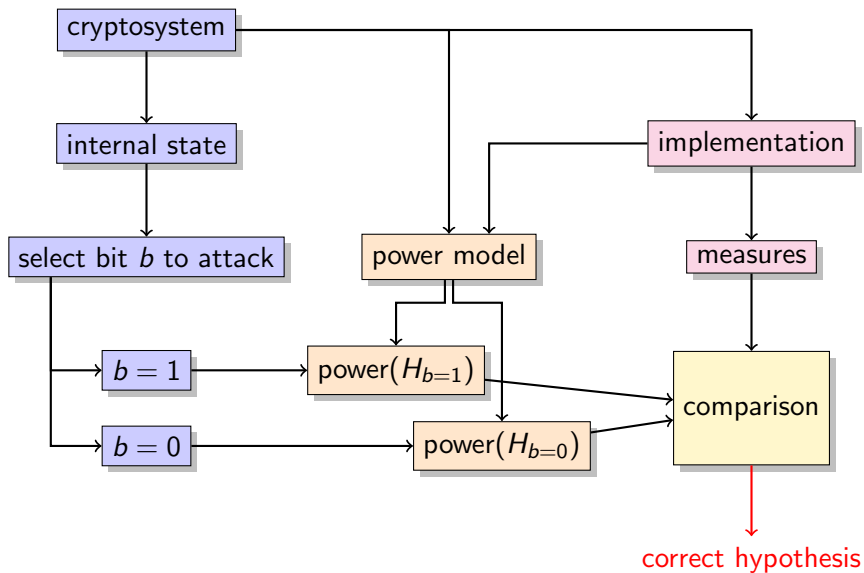
Differential Power Analysis (DPA)



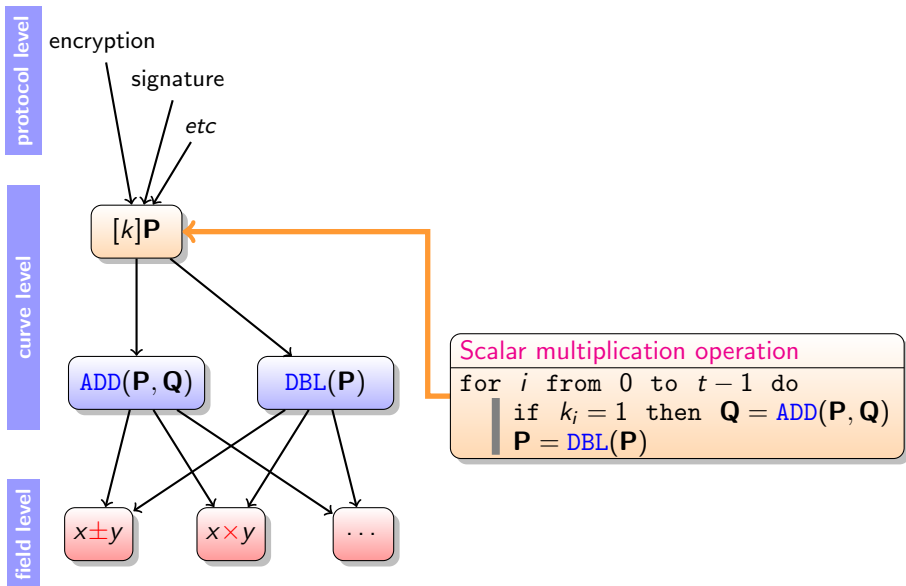
Differential Power Analysis (DPA)



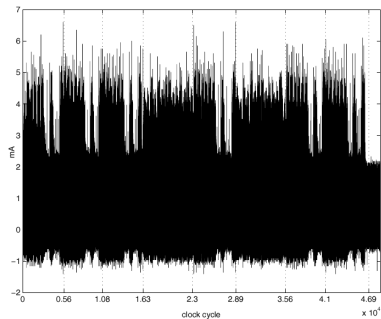
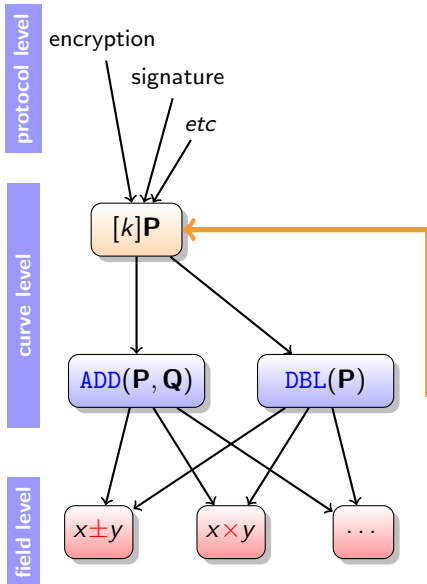
Differential Power Analysis (DPA)



Side Channel Attack on ECC



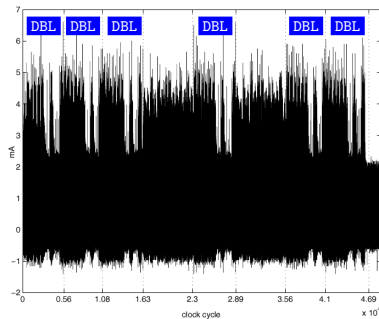
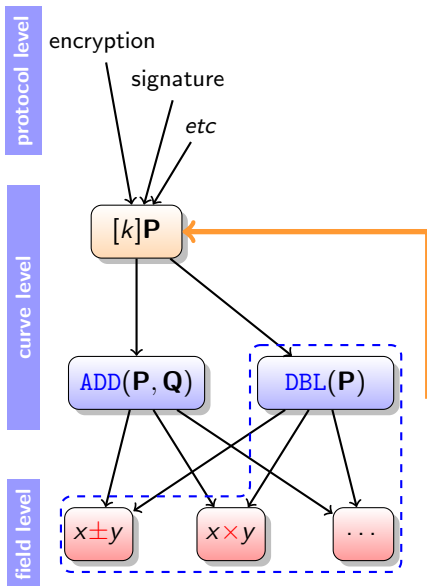
Side Channel Attack on ECC



Scalar multiplication operation

```
for  $i$  from 0 to  $t-1$  do  
    if  $k_i = 1$  then  $Q = \text{ADD}(P, Q)$   
     $P = \text{DBL}(P)$ 
```

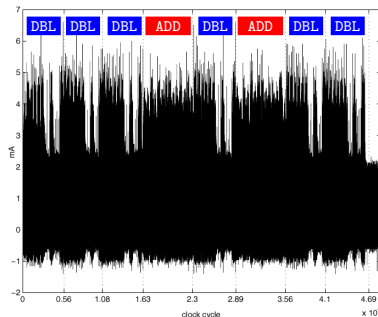
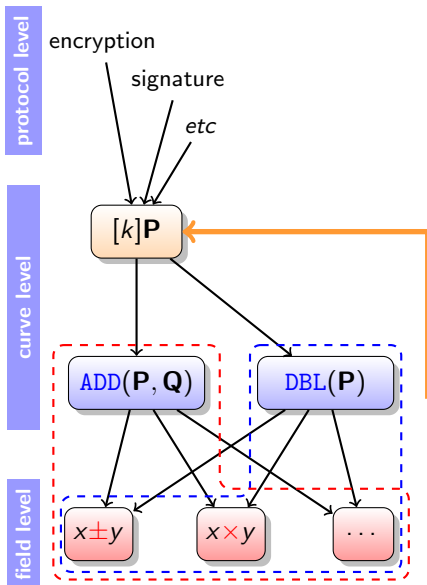
Side Channel Attack on ECC



Scalar multiplication operation

```
for  $i$  from 0 to  $t-1$  do  
    if  $k_i = 1$  then  $Q = \text{ADD}(P, Q)$   
     $P = \text{DBL}(P)$ 
```

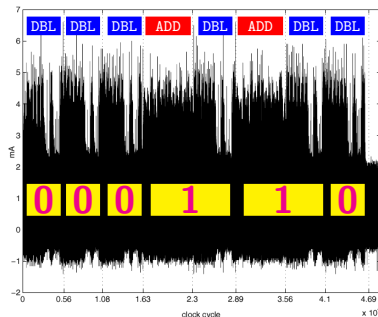
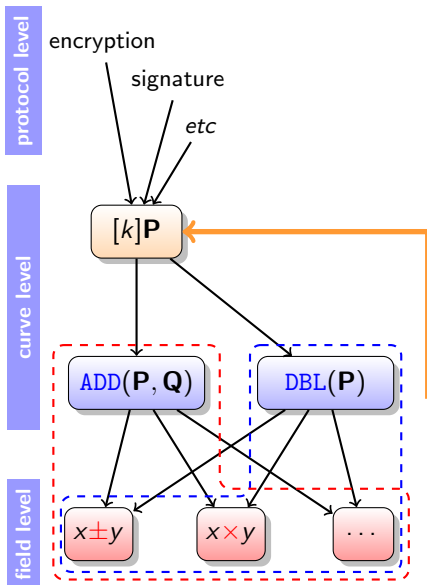
Side Channel Attack on ECC



Scalar multiplication operation

```
for  $i$  from 0 to  $t-1$  do  
    if  $k_i = 1$  then  $Q = ADD(P, Q)$   
     $P = DBL(P)$ 
```

Side Channel Attack on ECC

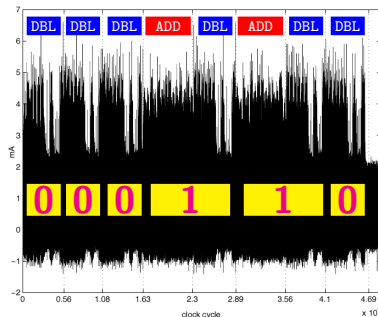
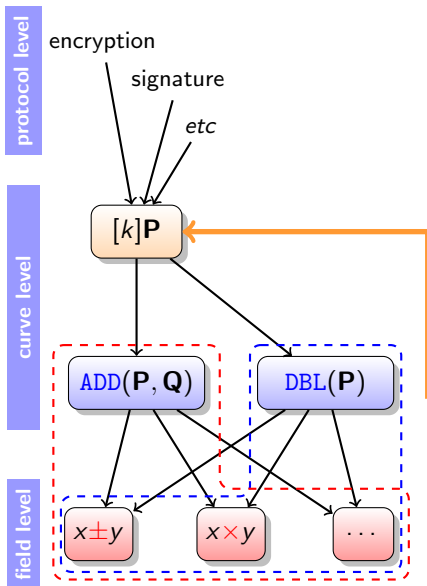


Scalar multiplication operation

```
for  $i$  from 0 to  $t-1$  do  
    if  $k_i = 1$  then  $Q = ADD(P, Q)$   
     $P = DBL(P)$ 
```

- simple power analysis (& variants)

Side Channel Attack on ECC



Scalar multiplication operation

```
for  $i$  from 0 to  $t-1$  do  
    if  $k_i = 1$  then  $Q = ADD(P, Q)$   
     $P = DBL(P)$ 
```

- simple power analysis (& variants)
- differential power analysis (& variants)
- horizontal/vertical/templates/... attacks

Countermeasures

Principles for preventing attacks:

- **embed** additional **protection blocks**
- **modify** the original circuit into a **secured** version
- application levels: circuit, architecture, algorithm, protocol. . .

Countermeasures

Principles for preventing attacks:


- **embed** additional **protection blocks**
- **modify** the original circuit into a **secured** version
- application levels: circuit, architecture, algorithm, protocol. . .

Countermeasures:

- electrical shielding
- detectors, estimators, decoupling
- use uniform computation durations and power consumption
- use detection/correction codes (for fault injection attacks)
- provide a random behavior (algorithms, representation, operations. . .)
- add noise (e.g. masking, useless instructions/computations)
- circuit reconfiguration (algorithms, block location, representation of values. . .)


Low-Level Coding and Circuit Activity

Assumptions:



- b is a bit (i.e. $b \in \{0, 1\}$, logical or mathematical value)
- electrical states for a wire  : V_{DD} (logical 1) or GND (logical 0)

Low-Level Coding and Circuit Activity

Assumptions:


- b is a bit (i.e. $b \in \{0, 1\}$, logical or mathematical value)
- electrical states for a wire  : V_{DD} (logical 1) or GND (logical 0)

Low-level codings of a bit:







	$b = 0$	$b = 1$
standard	 GND	 V_{DD}

Low-Level Coding and Circuit Activity

Assumptions:

- b is a bit (i.e. $b \in \{0, 1\}$, logical or mathematical value)
- electrical states for a wire  : V_{DD} (logical 1) or GND (logical 0)

Low-level codings of a bit:

	$b = 0$	$b = 1$
standard	 GND	 V_{DD}
dual rail	 $r_0 = V_{DD}$  $r_1 = \text{GND}$ $\left. \vphantom{\begin{matrix} r_0 \\ r_1 \end{matrix}} \right] (1, 0)_{DR}$	 $r_0 = \text{GND}$  $r_1 = V_{DD}$ $\left. \vphantom{\begin{matrix} r_0 \\ r_1 \end{matrix}} \right] (0, 1)_{DR}$

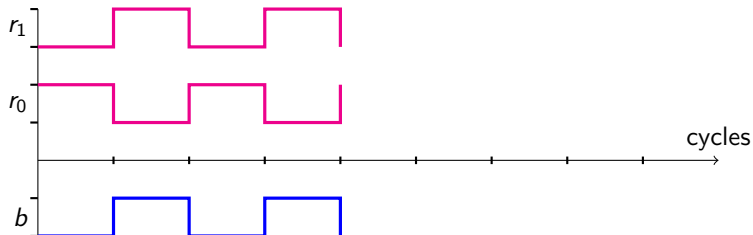
Low-Level Coding and Circuit Activity

Assumptions:

- b is a bit (i.e. $b \in \{0, 1\}$, logical or mathematical value)
- electrical states for a wire ———— : V_{DD} (logical 1) or GND (logical 0)

Low-level codings of a bit:

	$b = 0$	$b = 1$
standard	———— GND	$\text{———— } V_{DD}$
dual rail	$\begin{array}{l} \text{———— } r_0 = V_{DD} \\ \text{———— } r_1 = \text{GND} \end{array} \Big] (1, 0)_{DR}$	$\begin{array}{l} \text{———— } r_0 = \text{GND} \\ \text{———— } r_1 = V_{DD} \end{array} \Big] (0, 1)_{DR}$



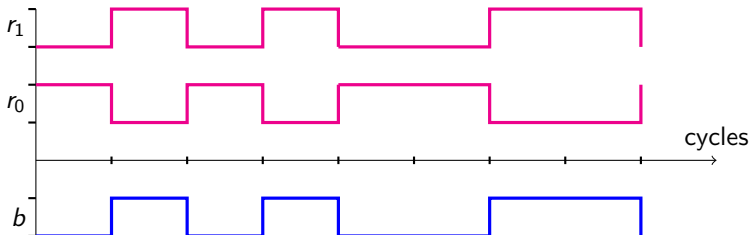
Low-Level Coding and Circuit Activity

Assumptions:

- b is a bit (i.e. $b \in \{0, 1\}$, logical or mathematical value)
- electrical states for a wire ————— : V_{DD} (logical 1) or GND (logical 0)

Low-level codings of a bit:

	$b = 0$	$b = 1$
standard	————— GND	————— V_{DD}
dual rail	$\begin{array}{l} \text{————— } r_0 = V_{DD} \\ \text{————— } r_1 = \text{GND} \end{array} \left. \vphantom{\begin{array}{l} r_0 \\ r_1 \end{array}} \right] (1, 0)_{DR}$	$\begin{array}{l} \text{————— } r_0 = \text{GND} \\ \text{————— } r_1 = V_{DD} \end{array} \left. \vphantom{\begin{array}{l} r_0 \\ r_1 \end{array}} \right] (0, 1)_{DR}$



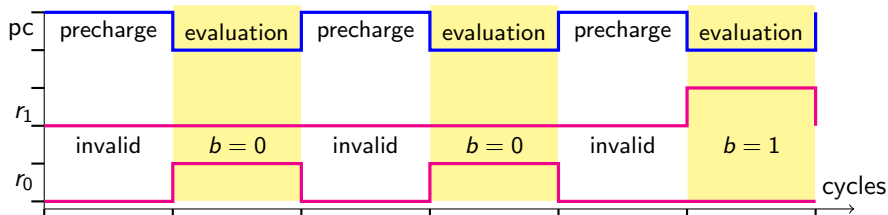
Circuit Logic Styles

Countermeasure principles: **uniformize** circuit activity and **exclusive coding**

Circuit Logic Styles

Countermeasure principles: **uniformize** circuit activity and **exclusive** coding

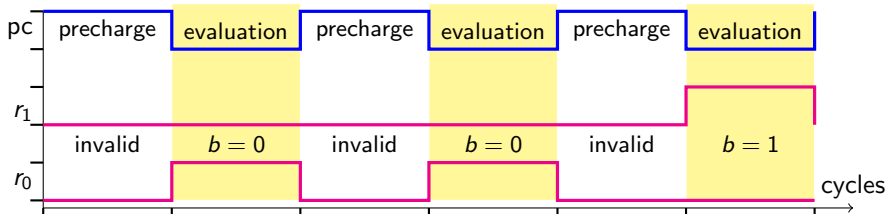
Solution based on precharge logic and dual-rail coding:



Circuit Logic Styles

Countermeasure principles: **uniformize** circuit activity and **exclusive** coding

Solution based on precharge logic and dual-rail coding:

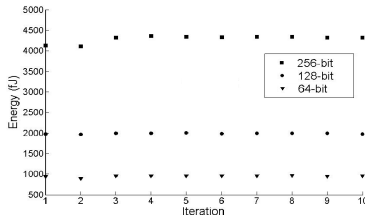
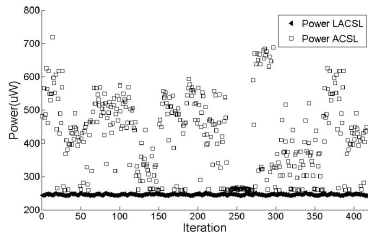
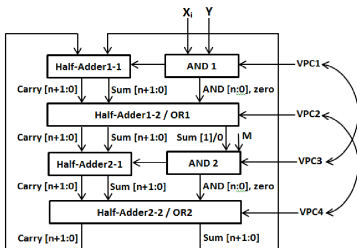
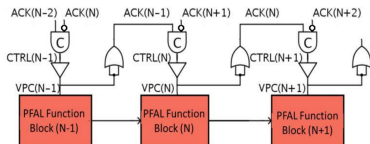


Solution based on validity line and dual-rail coding:



Important overhead: silicon area and local storage (registers)

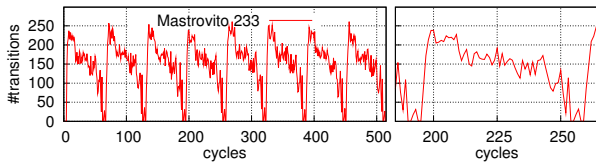
Circuit-Level Protections for Arithmetic Operators



References: [2] and [3]

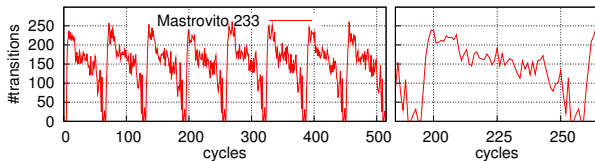
Protected Multipliers

Unprotected



Protected Multipliers

Unprotected



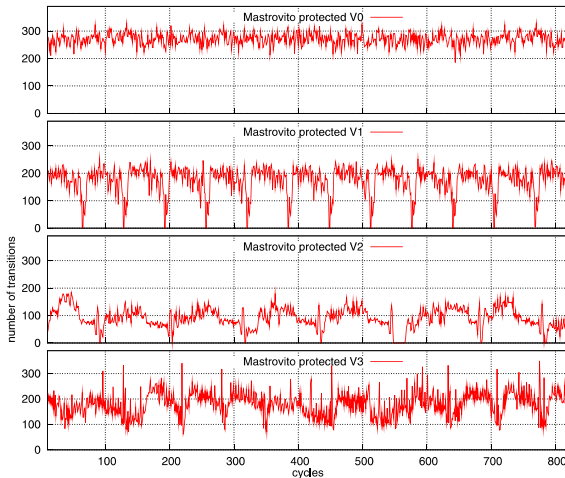
Protected

Overhead:
Area/time < 10 %

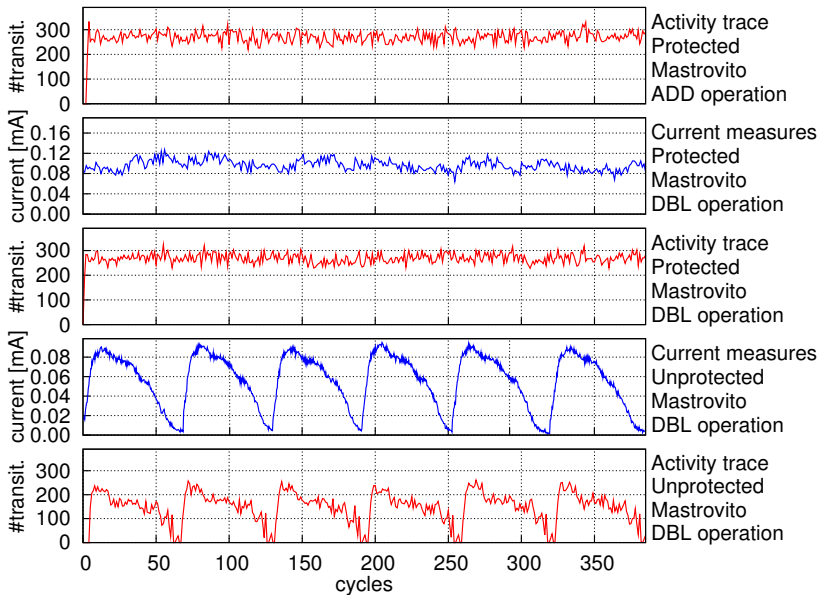
References:

PhD D. Pamula [5]

Articles: [8], [7], [6]



Protected ECC Accelerator



Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

k_{t-1}	k_{t-2}	\cdots	k_2	k_1	k_0
-----------	-----------	----------	-------	-------	-------

t explicit digits

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

k_{t-1}	k_{t-2}	\dots	k_2	k_1	k_0
-----------	-----------	---------	-------	-------	-------

implicit weights $2^{t-1} \ 2^{t-2} \ \dots \ 2^2 \ 2^1 \ 2^0$
 t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \dots, 600\}$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

k_{t-1}	k_{t-2}	\dots	k_2	k_1	k_0
-----------	-----------	---------	-------	-------	-------

2^{t-1} 2^{t-2} \dots 2^2 2^1 2^0 implicit weights
t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \dots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} =$$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

2^{t-1}	2^{t-2}	\dots	2^2	2^1	2^0	k_{t-1}	k_{t-2}	\dots	k_2	k_1	k_0
-----------	-----------	---------	-------	-------	-------	-----------	-----------	---------	-------	-------	-------

implicit weights
t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \dots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} =$$

k_{n-1}	\dots	k_1	k_0
a_{n-1}	\dots	a_1	a_0
b_{n-1}	\dots	b_1	b_0

n (2, 3)-terms
explicit "digits"
explicit ranks

$a_j, b_j \in \mathbb{N}$, $k_j \in \{1\}$ or $k_j \in \{-1, 1\}$, size $n \approx \log t$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i = \begin{array}{|c|c|c|c|c|c|} \hline 2^{t-1} & 2^{t-2} & \dots & 2^2 & 2^1 & 2^0 \\ \hline k_{t-1} & k_{t-2} & \dots & k_2 & k_1 & k_0 \\ \hline \end{array} \begin{array}{l} \text{implicit weights} \\ t \text{ explicit digits} \end{array}$$

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \dots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} = \begin{array}{|c|c|c|c|} \hline k_{n-1} & \dots & k_1 & k_0 \\ \hline a_{n-1} & \dots & a_1 & a_0 \\ \hline b_{n-1} & \dots & b_1 & b_0 \\ \hline \end{array} \begin{array}{l} n \text{ (2,3)-terms} \\ \text{explicit "digits"} \\ \text{explicit ranks} \end{array}$$

$a_j, b_j \in \mathbb{N}$, $k_j \in \{1\}$ or $k_j \in \{-1, 1\}$, size $n \approx \log t$

DBNS is a very **redundant** and **sparse** representation: $1701 = (11010100101)_2$

$$\begin{aligned} 1701 &= 243 + 1458 &= 2^0 3^5 + 2^1 3^6 &= (1, 0, 5), (1, 1, 6) \\ &= 1728 - 27 &= 2^6 3^3 - 2^0 3^3 &= (1, 6, 3), (-1, 0, 3) \\ &= 729 + 972 &= 2^0 3^6 + 2^2 3^5 &= (1, 0, 6), (1, 2, 5) \\ &\dots \end{aligned}$$

Randomized DBNS Recoding of the Scalar k

On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1 + 2 \leftrightsquigarrow 3$ $1 + 3 \leftrightsquigarrow 2^2$ $1 + 2^3 \leftrightsquigarrow 3^2$...
 control number of reductions (\leftarrow) and expansions (\rightarrow)

protocol level

encryption

signature

etc

$[k]P$

curve level

ADD(P, Q)

DBL(P)

TPL(P)

Point tripling operation

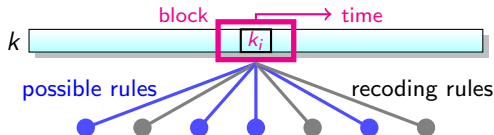
$$Q = \text{TPL}(P) = P + P + P$$

field level

$x \pm y$

$x \times y$

...



Randomized DBNS Recoding of the Scalar k

On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1 + 2 \leftrightsquigarrow 3$ $1 + 3 \leftrightsquigarrow 2^2$ $1 + 2^3 \leftrightsquigarrow 3^2$...
 control number of reductions (\leftarrow) and expansions (\rightarrow)

protocol level

curve level

field level

encryption

signature

etc

$[k]P$

ADD(P, Q)

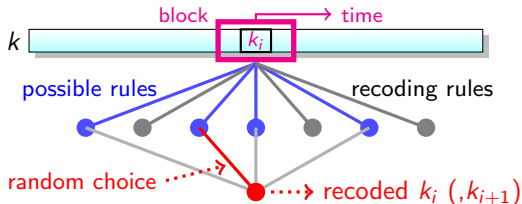
DBL(P)

TPL(P)

$x \pm y$

$x \times y$

...



Point tripling operation

$Q = \text{TPL}(P) = P + P + P$

Randomized DBNS Recoding of the Scalar k

On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1 + 2 \leftrightsquigarrow 3$ $1 + 3 \leftrightsquigarrow 2^2$ $1 + 2^3 \leftrightsquigarrow 3^2$...
control number of reductions (\leftarrow) and expansions (\rightarrow)

encryption

signature

etc

$[k]P$

ADD(P, Q)

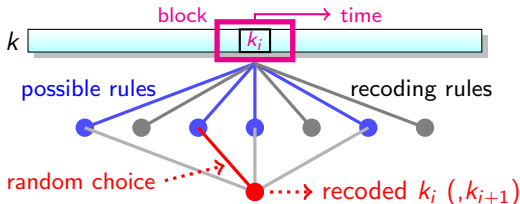
DBL(P)

TPL(P)

$x \pm y$

$x \times y$

...



Point tripling operation

$$Q = \text{TPL}(P) = P + P + P$$

DBNS is redundant \Rightarrow security \nearrow

DBNS is sparse \Rightarrow 20–30 % speed \nearrow

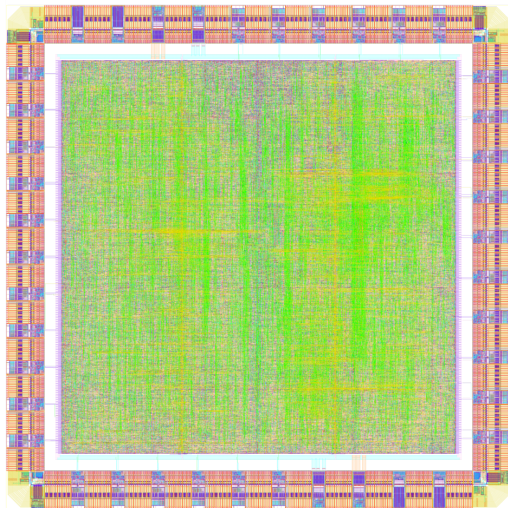
Ref: [1] Chabrier, Pamula & Tisserand.
Asilomar 2009

protocol level

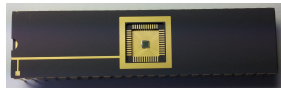
curve level

field level

ANR PAVOIS Integrated Circuit



ECC 256 bits
65 nm CMOS
1.5 mm²



Conclusion

- Side channel and fault attacks are **serious threats**
- **Attacks** are more and more **efficient** (many variants)
- Security analysis is mandatory at **all levels** (specification, algorithm, operation, implementation)
- Security = **trade-off** between performances, robustness and cost
- Security = *func*(secret value, attacker capabilities)
- **security** = **computer science + microelectronics + mathematics**

Conclusion

- Side channel and fault attacks are **serious threats**
- **Attacks** are more and more **efficient** (many variants)
- Security analysis is mandatory at **all levels** (specification, algorithm, operation, implementation)
- Security = **trade-off** between performances, robustness and cost
- Security = *func*(secret value, attacker capabilities)
- **security** = **computer science + microelectronics + mathematics**

Current works examples:

- Methods/tools for automating security analysis
- Circuit reconfiguration (representations, algorithms)
- Circuits with reduced activity variations
- Representation of numbers with error detection/correction “codes”
- Design space exploration
- CAD tools with security improvement capabilities

References I

- [1] T. Chabrier, D. Pamula, and A. Tisserand.
Hardware implementation of DBNS recoding for ECC processor.
In *Proc. 44rd Asilomar Conference on Signals, Systems and Computers*, pages 1129–1133, Pacific Grove, California, U.S.A., November 2010. IEEE.
- [2] J. Chen, A. Tisserand, E. M. Popovici, and S. Cotofana.
Robust sub-powered asynchronous logic.
In J. Becker and M. R. Adrover, editors, *Proc. 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–7, Palma de Mallorca, Spain, September 2014. IEEE.
- [3] J. Chen, A. Tisserand, E. M. Popovici, and S. Cotofana.
Asynchronous charge sharing power consistent Montgomery multiplier.
In J. Sparso and E Yahya, editors, *Proc. 21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 132–138, Mountain View, California, USA, May 2015.
- [4] P. C. Kocher, J. Jaffe, and B. Jun.
Differential power analysis.
In *Proc. Advances in Cryptology (CRYPTO)*, volume 1666 of *LNCS*, pages 388–397. Springer, August 1999.
- [5] D. Pamula.
Arithmetic Operators on $GF(2^m)$ for Cryptographic Applications: Performance - Power Consumption - Security Tradeoffs.
Phd thesis, University of Rennes 1 and Silesian University of Technology, December 2012.

References II

- [6] D. Pamula, E. Hryniewicz, and A. Tisserand.
Analysis of $\text{GF}(2^{233})$ multipliers regarding elliptic curve cryptosystem applications.
In *11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDeS)*, pages 271–276, Brno, Czech Republic, May 2012.
- [7] D. Pamula and A. Tisserand.
 $\text{GF}(2^m)$ finite-field multipliers with reduced activity variations.
In *4th International Workshop on the Arithmetic of Finite Fields*, volume 7369 of *LNCS*, pages 152–167, Bochum, Germany, July 2012. Springer.
- [8] D. Pamula and A. Tisserand.
Fast and secure finite field multipliers.
In *Proc. 18th Euromicro Conference on Digital System Design (DSD)*, pages 653–660, Madeira, Portugal, August 2015.
- [9] R. L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Communications of the ACM, 21(2):120–126, February 1978.

The end, questions ?

Contact:

- <mailto:arnaud.tisserand@univ-ubs.fr>
- <http://www-labsticc.univ-ubs.fr/~tisseran>
- CNRS, Lab-STICC Laboratory
University South Brittany (UBS),
Centre de recherche C. Huygens, rue St Maudé, BP 92116,
56321 Lorient cedex, France

Thank you