

Smart cards attacks and protections

Louis Goubin

Université de Versailles-St-Quentin-en-Yvelines

Université Paris-Saclay

M1 informatique – Site de Versailles

UE « Calcul Sécurisé »

1^{ère} partie

La sécurité logique est une abstraction

○ L'attaquant échange des messages avec le système :

- Messages connus: adversaire **passif** (écoute)
- Messages choisis (de façon adaptative) : adversaire **actif**

○ Puis utilise ces messages pour mettre en défaut des objectifs de sécurité :

- **Confidentialité** : e-mails, numéros de cartes, voix, ...
- **Intégrité** : téléchargement de logiciels, ...
- **Authenticité** : contrôle d'accès, signature électronique, ...
- **Anonymat** : paiement anonyme, vote électronique, ...
- ...

Sécurité au niveau cryptographique

○ Algorithmes cryptographiques = composants de base de la sécurité

- Chiffrement, signature, authentification, ...
- S'appuient fortement sur les mathématiques : probabilités, combinatoire, théorie des codes, théorie de nombres, réseaux euclidiens, corps finis, courbes (hyper)elliptiques, géométrie algébrique, graphes, ...
- Secret = Clé (*principe de Kerckhoff, 1883*)



- Les attaques utilisent des techniques de cryptanalyse
- Preuves of sécurité (partielles), sous l'hypothèse de la difficulté d'un certain problème mathématique (théorie de la complexité)

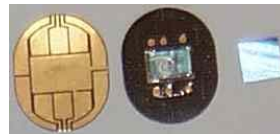
Securité au niveau des protocoles

○ Procotoles = Sur un réseau, supposé hostile

- Des **intrus** peuvent lire, modifier et effacer le trafic, peuvent prendre le contrôle d'un ou plusieurs éléments du réseau.
- Des **attacks** souvent non-intuitives :
 - Attaques de base : utilisent les fonctionnalités de base, dans un ordre arbitraire.
 - Attaques plus complexes : utilisent également des propriétés subtiles des algorithmes cryptographiques, ou l'analyse statistique du trafic...
- **Preuves de sécurité** pour les protocoles :
 - **Modèle** mathématique/logique du système & des objectifs de sécurité
 - Procédure effective pour vérifier la preuve (**méthodes formelles**).

Sécurité physique

- Modèle de sécurité plus général : utilise les aspects physiques du calcul.
- Menace potentielle pour tout dispositif portable/embarqué (spécialement les cartes à puce)
- Attaques invasives vs non invasives
 - Attaques **invasives** → « dépackager » le circuit pour avoir un accès direct à ses composants (e.g. connecter un fil sur un bus de données pour écouter les données transférées)
 - Attaques **non-invasives** → utilisent uniquement les informations disponibles de façon externe (temps de calcul, consommation de courant, ...)



The RSA Algorithm



RSA Cryptosystem (1977)

de facto standard of public-key cryptosystems

p, q : primes, $n = pq$, $ed = 1 \bmod (p-1)(q-1)$,

e, n : public key, d : secret key, (factoring, n : 1024 bits)

M : message, $M \in \{0, 1, 2, \dots, n-1\}$.

Encryption: $C = M^e \bmod n$ e : small ($2^{16}+1$)

Decryption: $M = C^d \bmod n$ d : large ($d > n^{1/2}$)

Fast Exponentiation

The binary representation of $d = d[k-1]2^{k-1} + d[k-2]2^{k-2} + \dots + d[1]2^1 + d[0]2^0$, where $d[k-1]=1$.

Left-to-right binary method

Input C, n, d

Output $C^d \bmod n$

$X = C;$

For $i=k-2$ to 0

$X = X^2 \bmod n;$

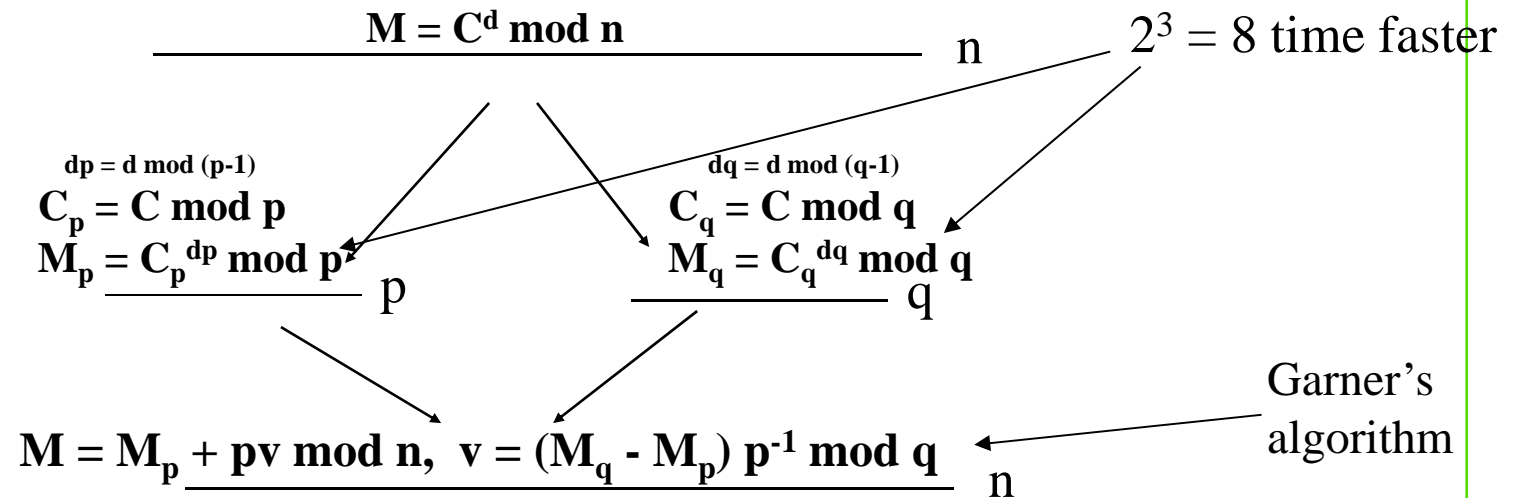
if $d[i]=1$, then $X = X * C \bmod n;$

Return X

cubic complexity $O((\log n)^3)$.
- we need about 1500 modular multiplications for 1024-bit n, d on average.

$d =$ 179769313486231590772930519078902473361797697894230657273430081157732639445209167262771634937140456477800995856
4863673560357494227785840418926558467439899258695049140360821770965996851973903412635215659390188627764072341203
1668285970266526289737711820513944871376325649575655785893257302729658745304709432808

RSA Decryption using Chinese Remainder Theorem



RSA decryption using the CRT can be computed about **4 times faster** than the original decryption.

RSA with CRT

Algorithm RSA_Decryption_CRT ($n=pq$)

Input $C, n, p, q, dp, dq, p_inv_q$

Output M

Pre-computation
avoiding inversion

```
1:   $M_p = C^{dp} \bmod p$ ;  
2:   $M_q = C^{dq} \bmod q$ ;  
3:   $v = (M_q - M_p) p\_inv\_q \bmod q$ ;  
4:   $M = M_p + pv$ ;  
5:  Return  $M$ 
```

PKCS #1, <http://www.rsasecurity.com/rsalabs/pkcs/>

PKCS #1 - RSA Cryptography Standard

This document provides recommendations for the implementation of public-key cryptography based on the RSA algorithm, covering the following aspects: cryptographic primitives; encryption schemes; signature schemes with appendix; ASN.1 syntax for representing keys and for identifying the schemes.

Version 2.1

- PKCS #1: RSA Cryptography Standard: [MS-Word](#), [Acrobat PDF](#).
- [ASN Module for PKCS #1 v2.1](#)
- Errata for PKCS #1 v2.1 ([txt](#))
- NOTE: A new OID has been defined for the combination of the v1.5 signature scheme and the SHA-224 hash function:

```
sha224WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 14 }
```

Like the other sha*WithRSAEncryption OIDs in PKCS #1 v2.1, this OID has NULL parameters. The DigestInfo encoding for SHA-224 (see Section 9.2, Note 1) is:

```
(0x)30 2d 30 0d 06 09 60 86 48 01 65 03 04 02 04 05 00 04 1c || H
```

Version 2.0

- PKCS #1 RSA Cryptography Standard: [MS-Word](#), [ASCII](#). View [changes to previous draft](#).
- PKCS #1 Amd. 1: Multi-Prime RSA: [MS-Word](#), [Adobe Acrobat](#), [PostScript](#).

Version 1.5

- RSA Encryption Standard: [ASCII](#), [MS-Word](#), [PostScript](#) and [Gzip PostScript](#)

Related Documents

- Corrected: [ASN.1 module for PKCS #1 v2.0](#)
- Presentation of v 2.0 ([PowerPoint](#)) from the ['98 Workshop](#)
- PKCS #1 Informational RFC (3447): [ASCII](#)

Test Vectors

- [RSA-OAEP and RSA-PSS test vectors \(.zip file\)](#)

[PKCS Home](#) | [PKCS Mailing Lists](#)

[#1](#) | [#3](#) | [#5](#) | [#6](#) | [#7](#) | [#8](#) | [#9](#) | [#10](#) | [#11](#) | [#12](#) | [#13](#) | [#15](#)

Questions and comments can be submitted via our [contact form](#).

[Crypto FAQ](#)
[CryptoBytes](#)
[RSA Algorithm](#)
[PKCS](#)
[RFID](#)
[Tech Notes](#)
[Staff & Associates](#)
[Bulletins](#)
[Challenges](#)

Security Analysis of RSA Cryptosystem

SECURE RSA!

Klima-Rosa attack against PGP

← Programming or
Coding failure

Side Channel
Attack (SCA)

Timing
Attack

Differential Fault
Attack (DFA)

← Implementation
failure

Broad cast attack

Common modulus

Bleichenbacher Attack (PKCS1)

← Padding failure

Chosen Ciphertext Attack (Simmons)

← Protocol failure

Strong prime, Cycling attack, low exponent attack

← Other parameters

Factoring $n = pq$

← Number theoretic
Problems (key size)

Timing Attacks



What are Timing Attacks ?

- The term “Timing Attack” was first introduced at CRYPTO'96 in Paul Kocher's paper
- Few other theoretical approaches without practical experiments up to the end of 97'
- Theory was put into practice in early 98'
- Timing attacks belong to the large family of "side channel" attacks



What are Timing Attacks ?

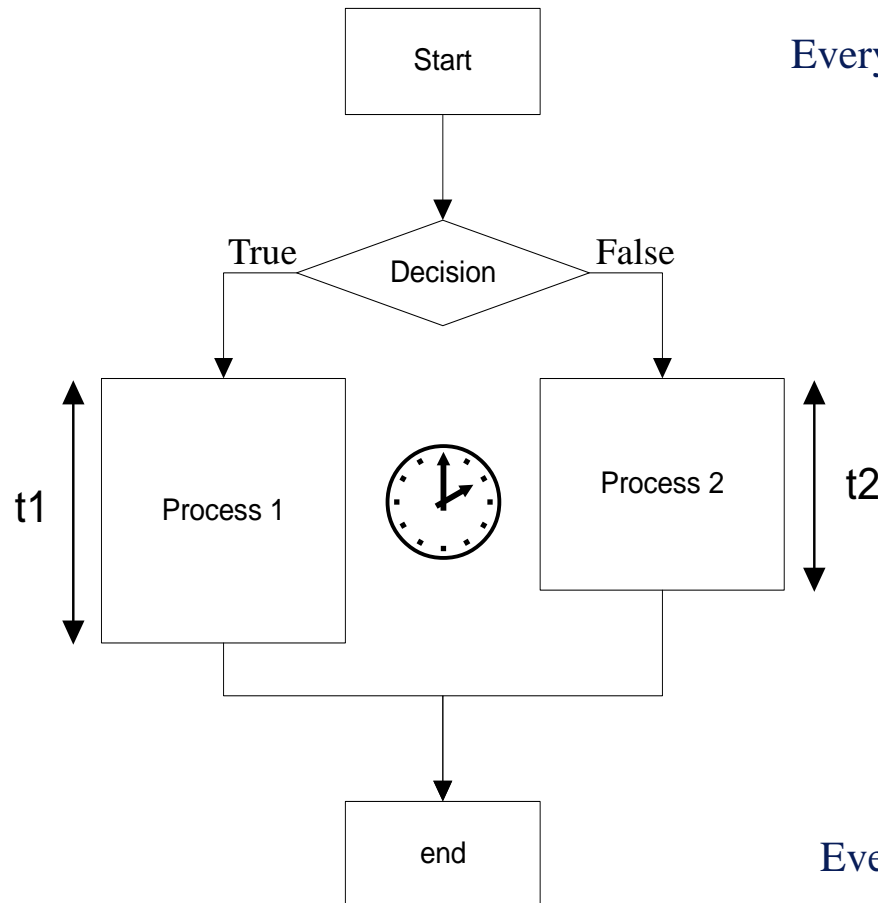
○ Principle of Timing Attacks :

- Secret data are processed in the card
- Processing time
 - depends on the value of the secret data
 - leaks information about the secret data
 - can be measured (or at least their differences)

○ Practical attack conditions

- Possibility to monitor the processing of the secret data
- Have a way to record processing times
- Have basic computational & statistical tools
- Have some knowledge of the implementation

What are Timing Attacks ?



Everything performed unconditionally before the test

A test based on secret data is performed that leads to a boolean decision

Depending on the boolean condition, the process may be long ($t1$) or short ($t2$)

Everything performed unconditionally after the test

Timing attack on RSA

- **Timing Attacks:** by precisely measuring the time it takes the smartcard to perform an decryption, Marvin can discover d .
- “repeated squaring algorithm”, compute $C=M^d \bmod N$.
 $d=d_n d_{n-1} \dots d_0$
 - Set z equals to M and $C=1$. For $i=0, \dots, n$ do:
 - if $d_i=1$ set $C=C*z \bmod N$
 - set z equal to $z^2 \bmod N$At the end, C has the value $M^d \bmod N$
- To mount attack, Marvin asks the smartcard to generate signatures on a large number of random messages $M_1, M_2, \dots, M_k \in \mathbb{Z}_N^*$ and measure the time T_i it takes to generate each signature.

Timing attack on RSA

○ Timing Attack

- If $d_1=1$, smartcard computes $Cz=MM^2 \bmod N$ and, Otherwise it does not. Let t_i be the time it takes the smart card to compute $M_i M_i^2 \bmod N$. The t_i 's differ from each other and depends on M_i . Marvin measures them offline.
- When $d_1=1$, the two ensembles $\{t_i\}$ and $\{T_i\}$ are correlated. when $d_1=0$, they behave as independent random variables. By measuring the correlation, Marvin can determine $d_1=1$ or 0.
- Continuing in this way, he can discover $d_2, d_3 \dots$ and so on.
- Solutions: 1) add appropriate delay s.t. modular exponentiation always takes a fixed amount of time. 2) Rivest's blinding trick.
- Kocher's Power cryptanalysis?

Power Analysis Attacks



Power Analysis: Basic Principles

○ ICC's Power Consumption leaks information about data processing

- Power Consumption = $f(\text{processing}, \text{data})$

○ Deduce information about secret data and processing

- empirical methods
- statistical treatment

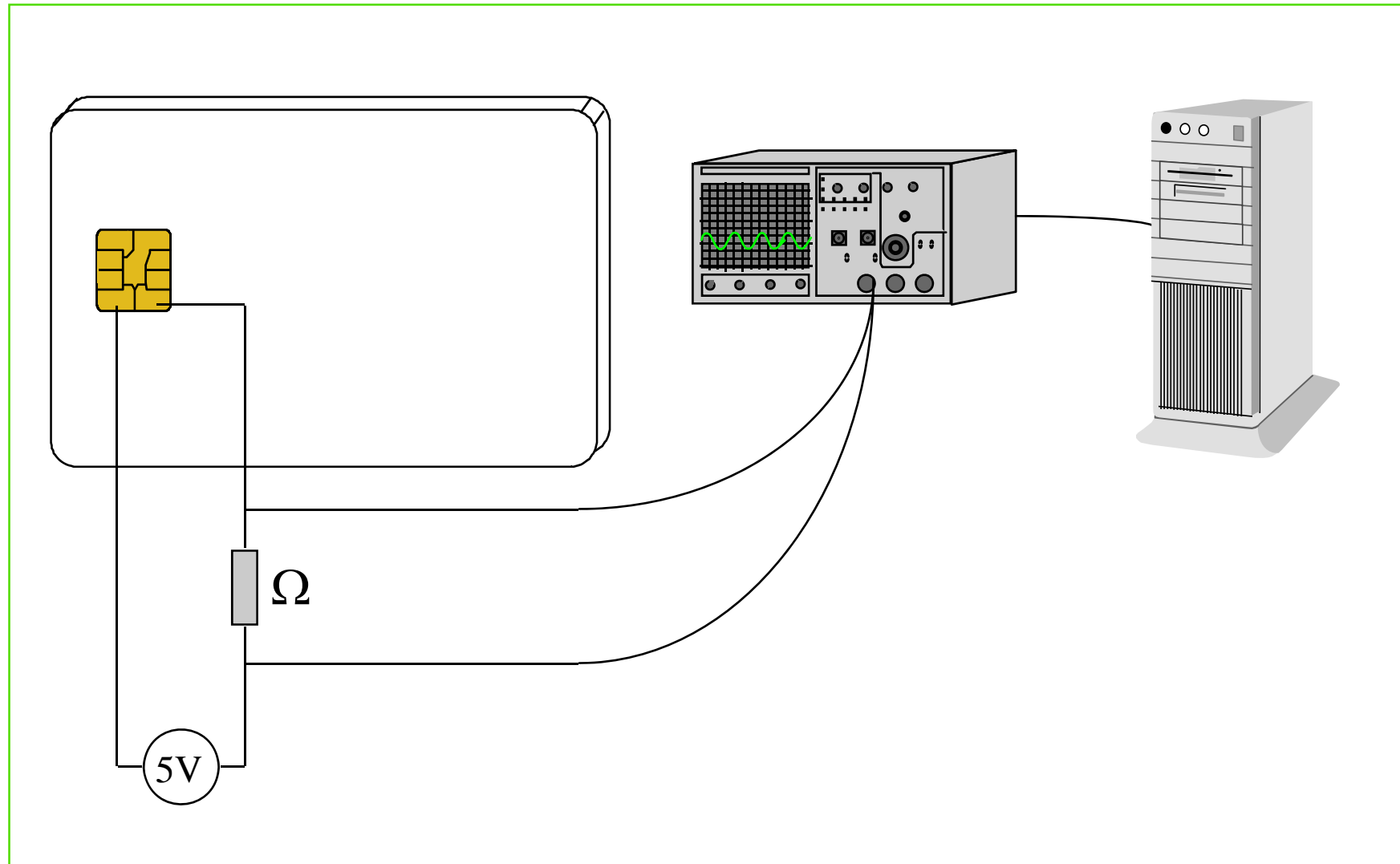
○ Example : reverse engineering of an algorithm

- The algorithm structure
- Electrical signatures

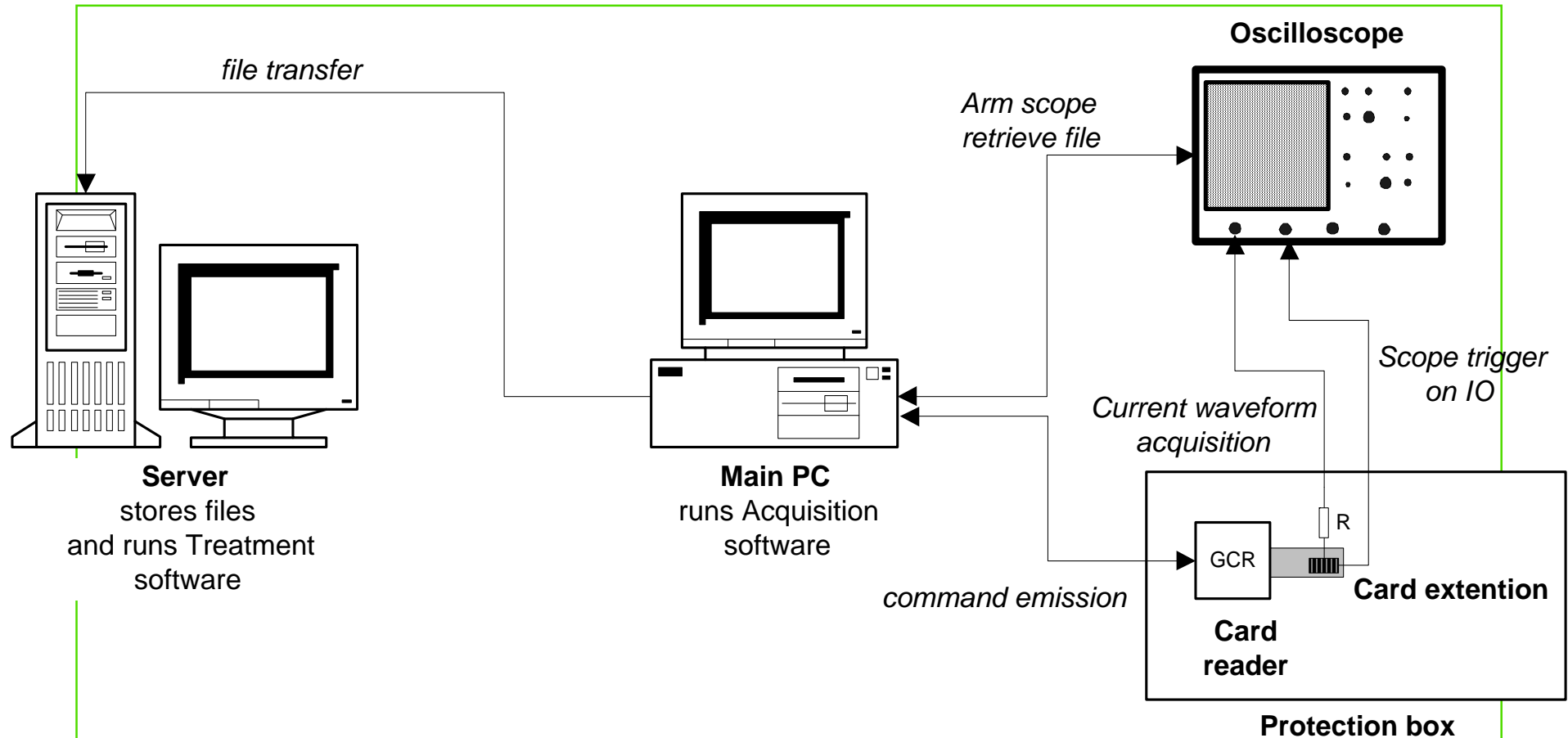
○ Single Power Analysis (SPA)

- Attack against the DES key schedule
- Attack against RSA

Power Analysis Tools



Experimental equipment



Devices for monitoring the current consumption of a

Information leakage

○ The power consumption of a chip depends on

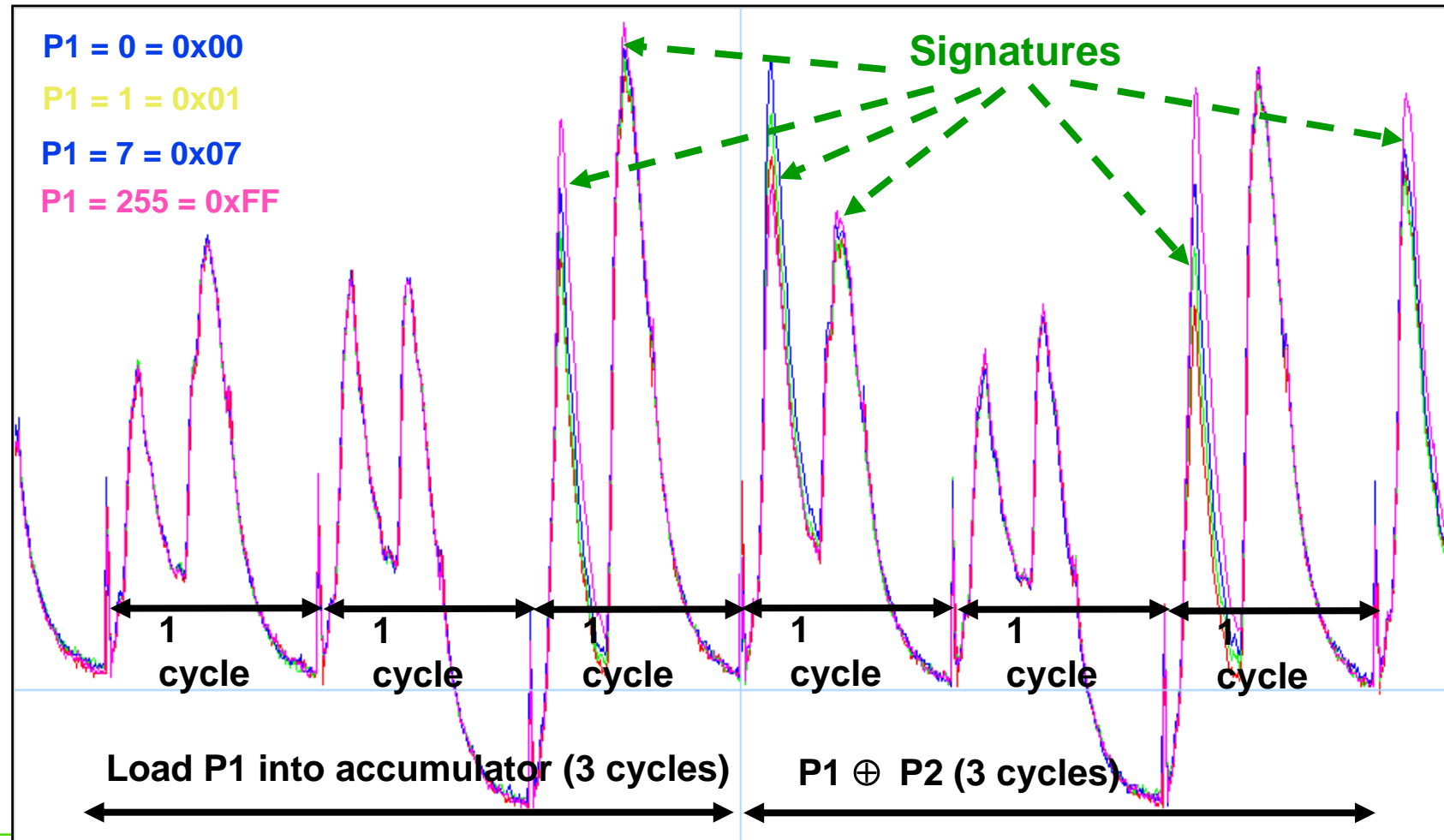
- the manipulated data
- the executed instruction

○ Leakage models

- Hamming Weight of the data, address, code Op
 - $HW(0) = 0$
 - $HW(1) = HW(2) = HW(4) = HW(2^n) = 1$
 - $HW(3) = HW(5) = HW(6) = HW(9) = 2$
 - ...
 - $HW(255) = HW(0xFF) = 8$
- Transitions weight (flipping bits on a bus state) :
 - $HW(state_i \oplus state_{i-1})$
- Other models, chips & technologies ...

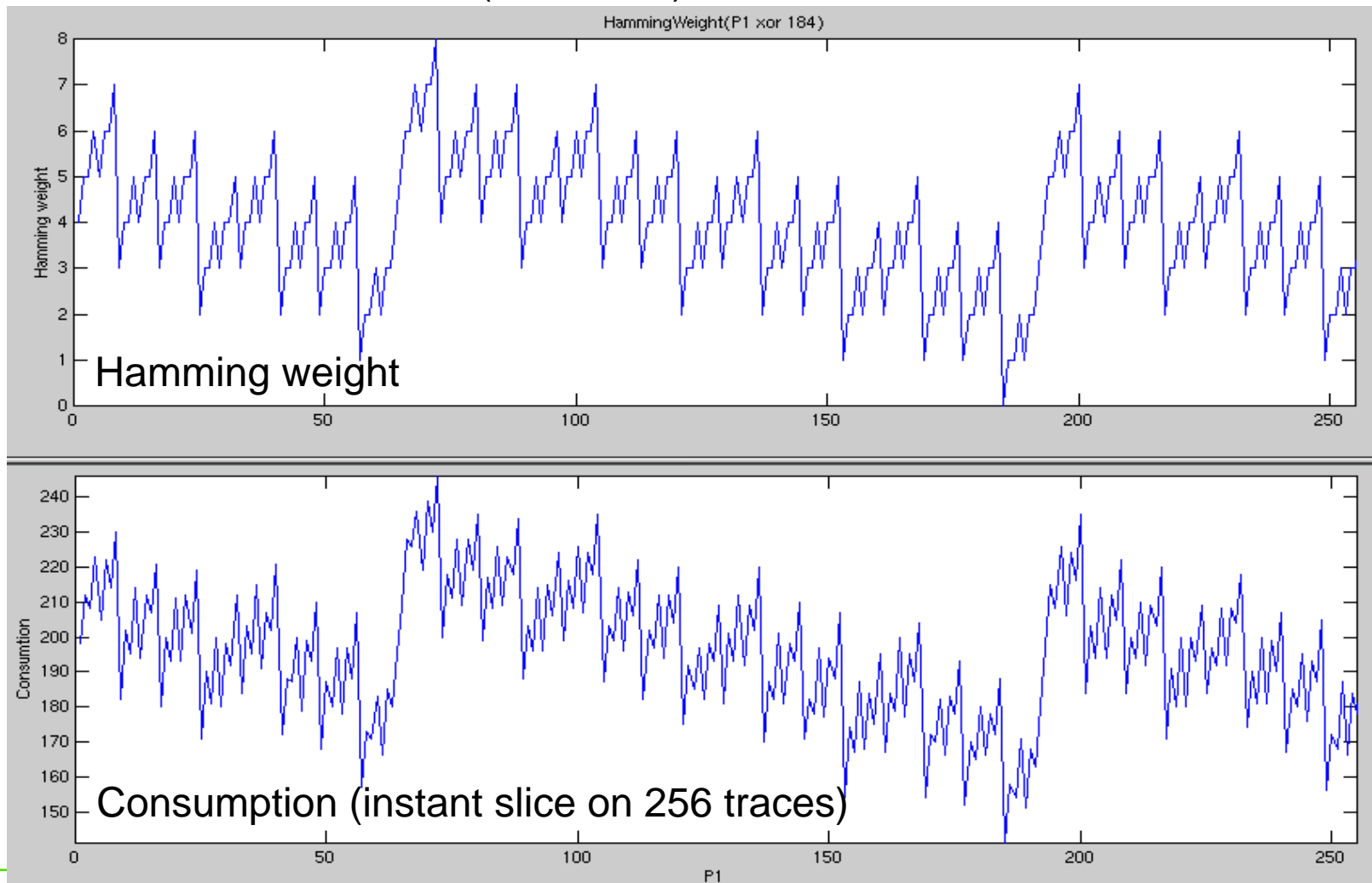
Information leakage

Load P1 and XOR with P2 = 0 ($P1 \oplus P2$ with $P1 = 0, 1, 7, 255$)

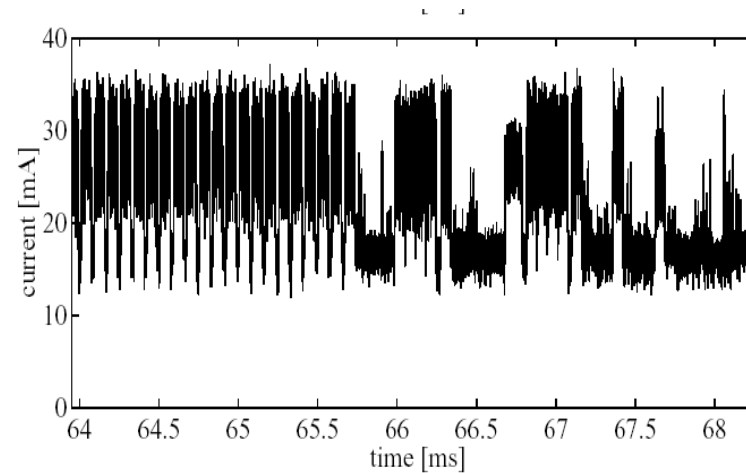
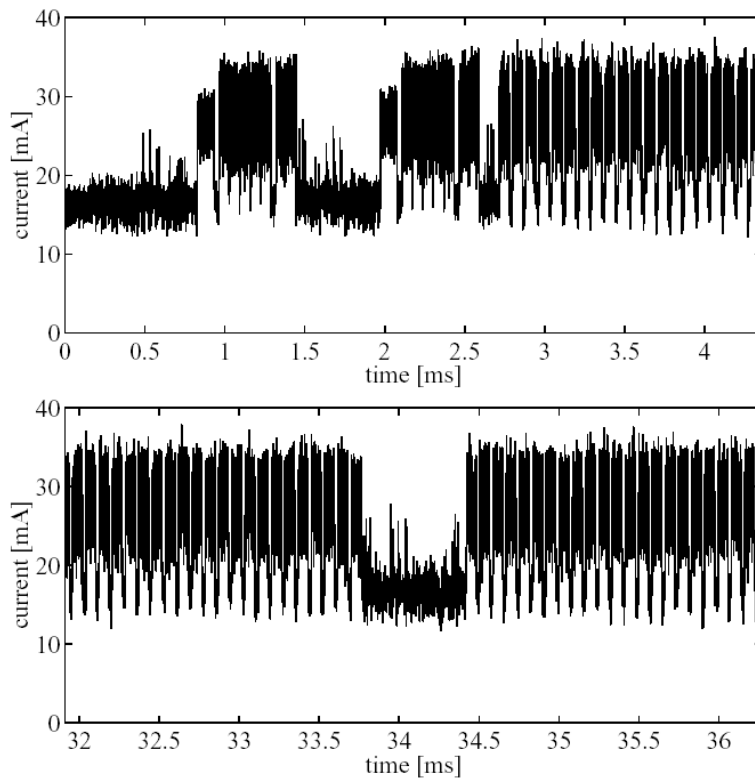


Information leakage

HW ($P1 \oplus 184$), for $P1 = 0, 255$



Power Consumption of RSA-CRT



Cited from the paper: R.Novak, "SPA-Based Adaptive Chosen Ciphertext Attack on RSA Implementation," PKC 2002, LNCS 2274, pp.252-262, 2002.

Side Channel Attacks

Left-to-right binary method

Input M, n, d

Output $M^d \bmod n$

$X = M;$

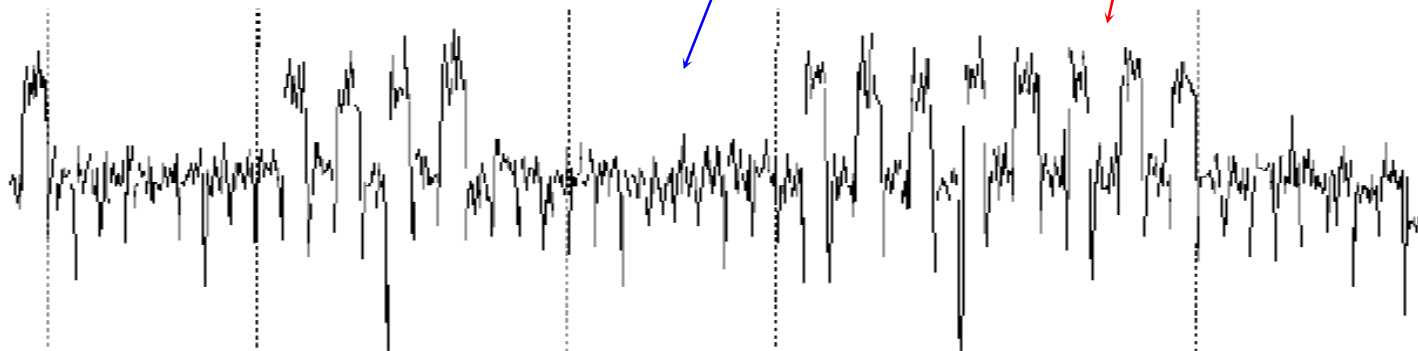
For $i=k-2$ to 0

$X = X * X \bmod n;$

if $d[i]=1$, then $X = X * M \bmod n;$

Return X

The time or the power to execute
Squaring and **Multiplication** are different
(side-channel information).



Simple Power Analysis

○ Simple (Single) Power Analysis context

- Find out a secret or private key
- Known algorithm
- Unknown implementation (*background culture recommended*)

○ Conditions

- 1 card available
- Learning phase required (*signature location*)
- Key inference on a single curve (*with relevant height of view*)
- Possibly known plain or ciphertext

SPA attack on RSA

○ SPA against RSA private exponentiation

$$s = m^d \bmod n$$

- n large modulus, say 1024 bits ($n = p * q$, with p & q large primes)
- m message : slightly smaller than n (say 1023 bits)
- s signature
- d private exponent such that : $e * d \equiv 1 \bmod (p-1)(q-1)$, with e public exponent

○ The attacker aims at retrieving d

SPA attack on RSA

- basic “square and multiply” algorithm
- exponent bits scanned from MSB to LSB (left to right)

Let $k = \text{bitsize of } d \text{ (say 1024)}$

Let $s = m$

For $i = k-2$ down to 0

Let $s = s * s \bmod n$ (*SQUARE*)

If (bit i of d) is 1 then

Let $s = s * m \bmod n$ (*MULTIPLY*)

End if

End for

Example : $s = m^9 = m^{1001b}$

init (MSB 1) $s = m$

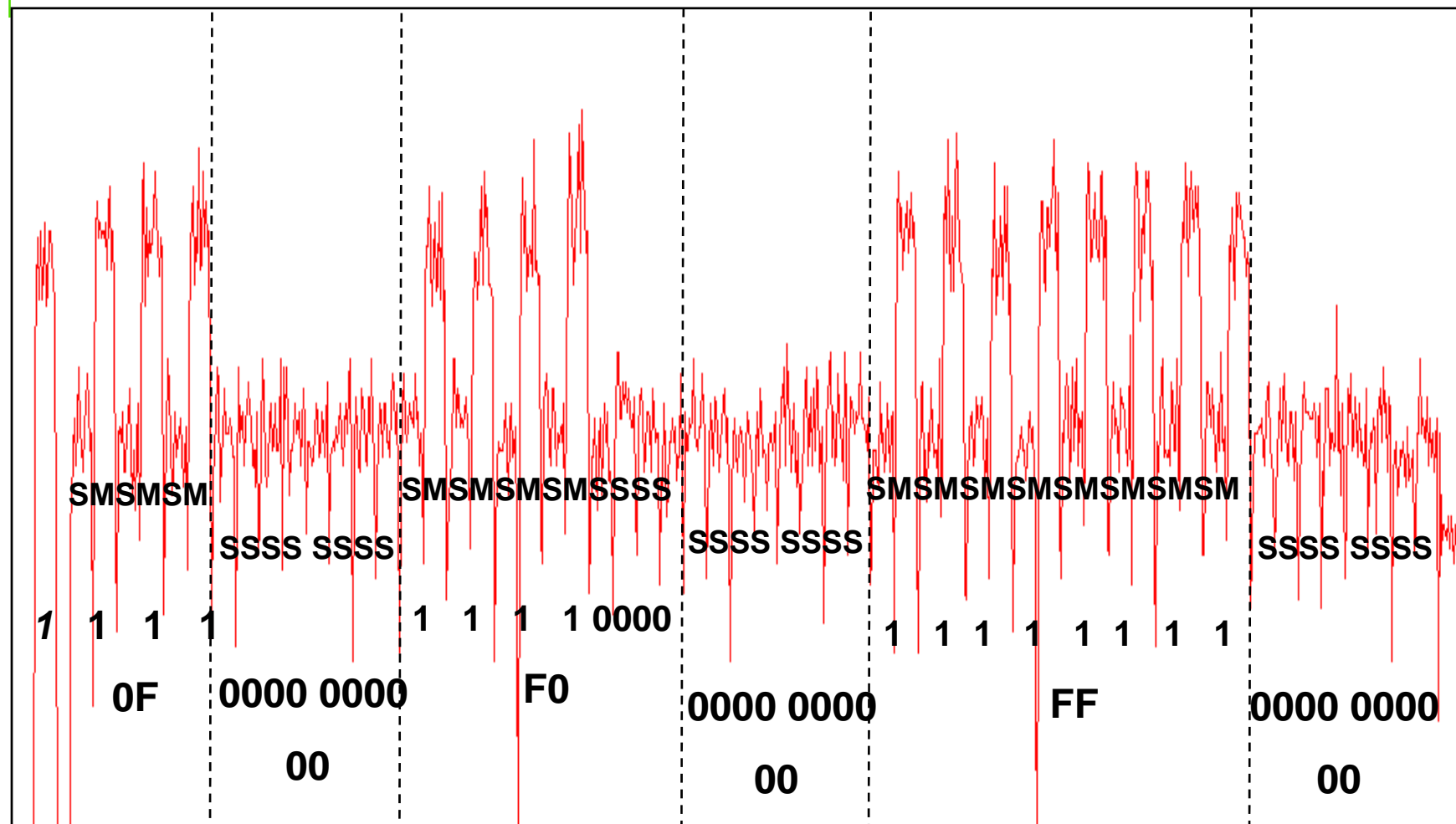
round 2 (bit 0) $s = m^2$

round 1 (bit 0) $s = (m^2)^2 = m^4$

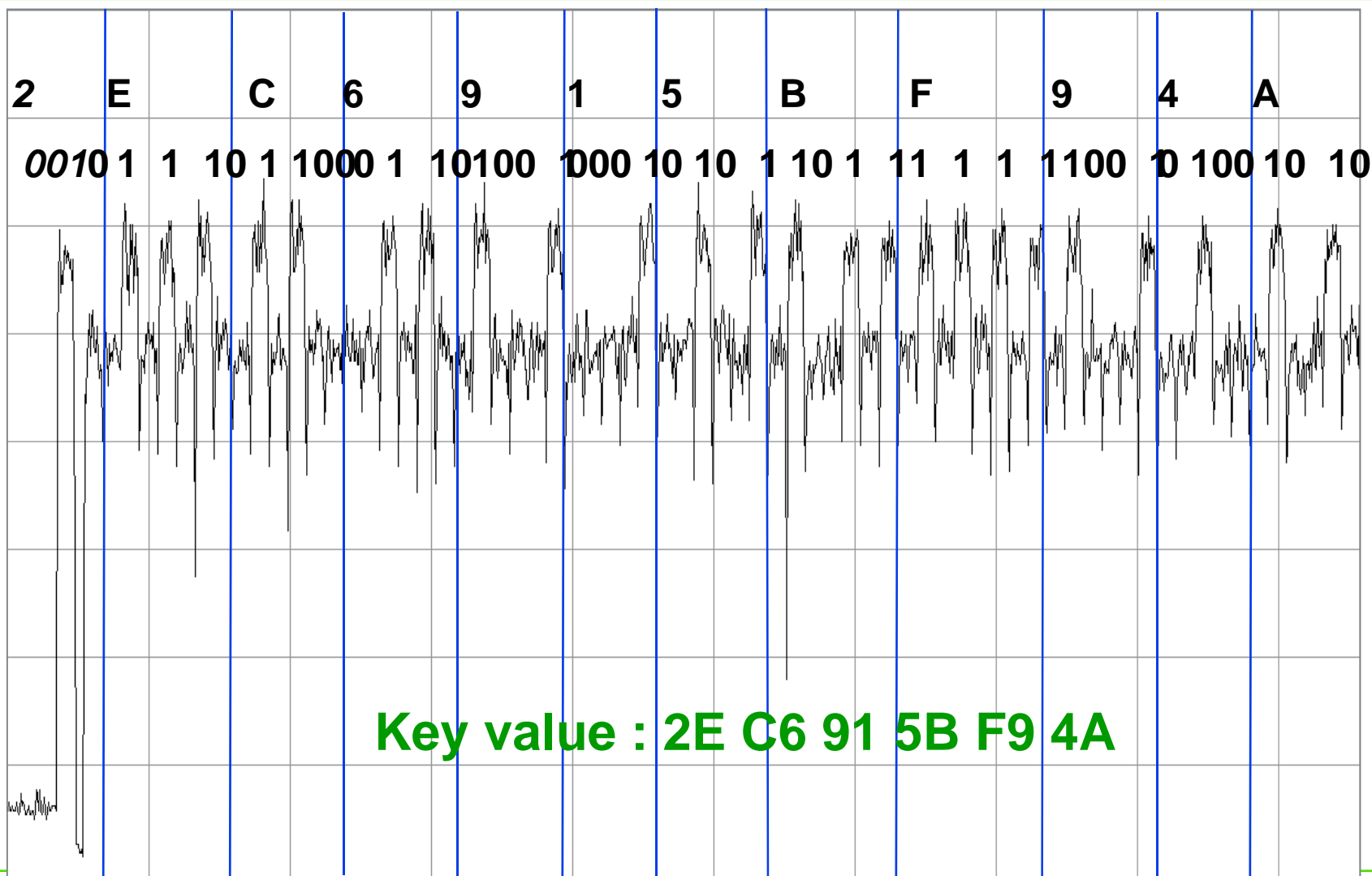
round 0 (bit 1) $s = (m^4)^2 * m = m^9$

SPA attack on RSA

Test key value : 0F 00 F0 00 FF 00



SPA attack on RSA



What you can do with SPA

○ SPA uses implementation related patterns

○ SPA strategy

- algorithm knowledge
- reverse engineering phase (*signature location*)
- representation tuning (*height of view, zoom, visualisation*)
- then play with implementation assumptions...

○ SPA is always specific due to

- the algorithm implementation
- the applicative constraints
- the chip's technology (*electrical properties*)
- possible counter-measures...

Counter-measures

○ Counter-measure : anything that foils the attack !

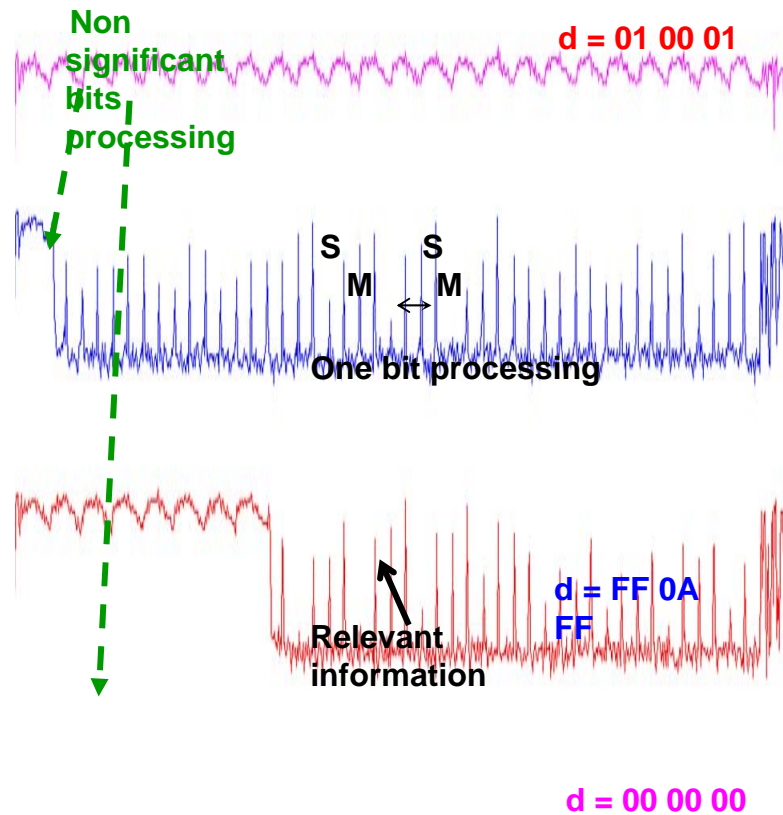
○ Trivial counter-measure

- prohibit code branches conditioned by the secret bits

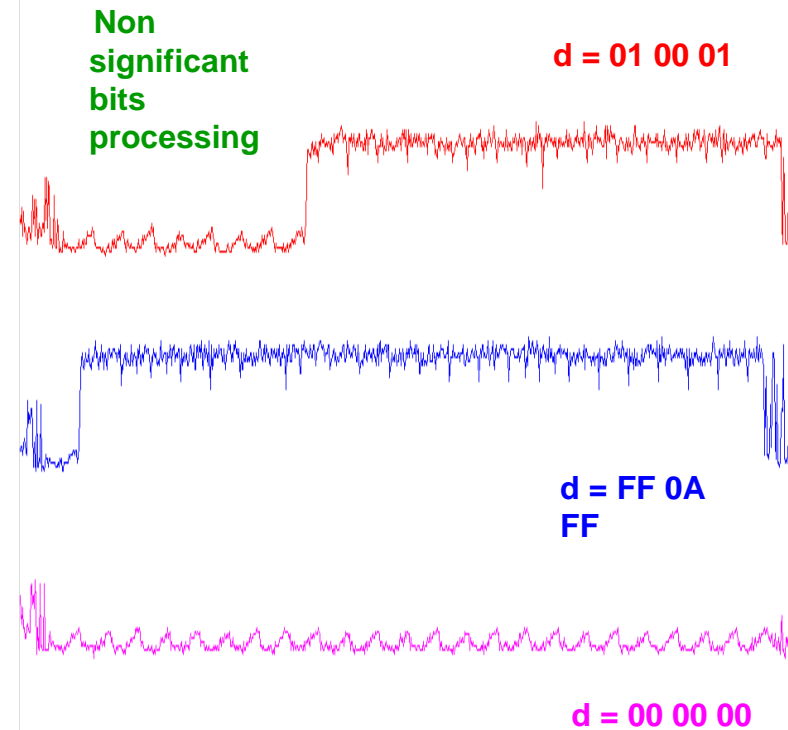
○ Advanced counter-measures

- algorithm specification refinement
 - code structure
 - data whitening
- implementation design based on the chip's resources
 - play with instructions set
 - hardware electrical behaviour (current scrambler, desynchronisation, cryptoprocessor...)

Effects of Counter-measures



SPA possible



SPA resistant !

**Although same specified
RSA !**

Differential Power Analysis

○required number of acquisitions : 500 to 10,000

○prerequisite

- physical access to the card under attack
- access to either plaintext M or ciphertext C
- varying plaintext and constant key
- algorithm specifications (MANDATORY)

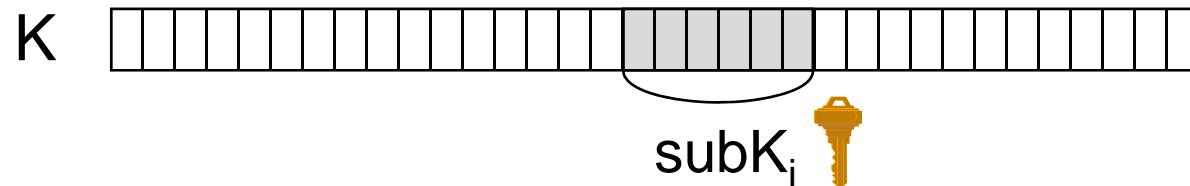
○cost

- A few dollars (to a few thousands)
- A few days training
- Average good level of expertise
- Chip and implementation independent

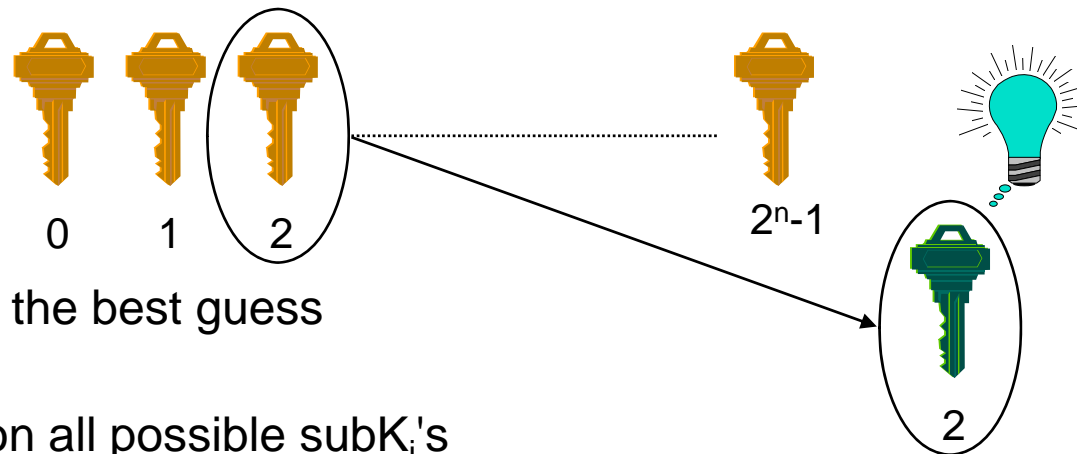
Differential Power Analysis

description :

- choose a subset (subK_i) of n bits of K_1



- perform a statistical test for each possible value of a subK_i

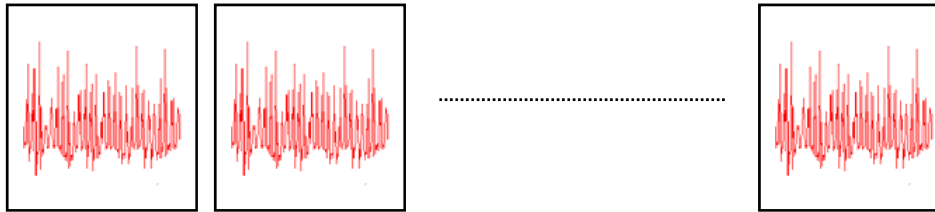


- Choose the best guess
- Iterate on all possible subK_i 's

Differential Power Analysis

○ DPA statistical test :

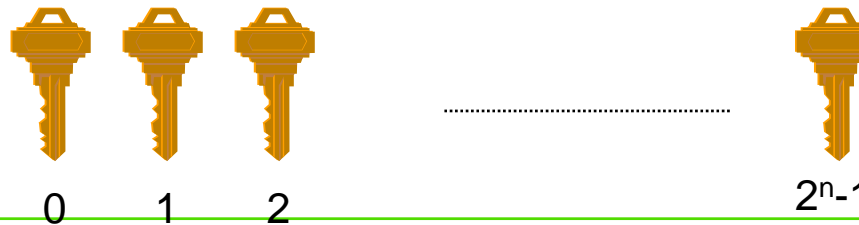
- a batch of data acquisitions for various messages M_k



- the corresponding plaintext M_k or the ciphertext C_k

dfdsffb
fdgcxv
.....
lkkljsdq
 M_0 M_1 M_k

- the values of the sub K_i

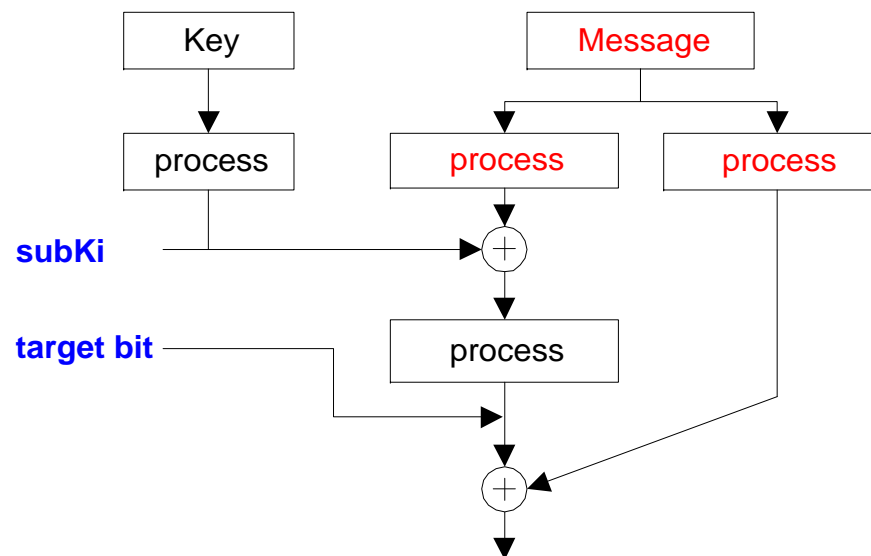


Differential Power Analysis

ODPA statistical test :

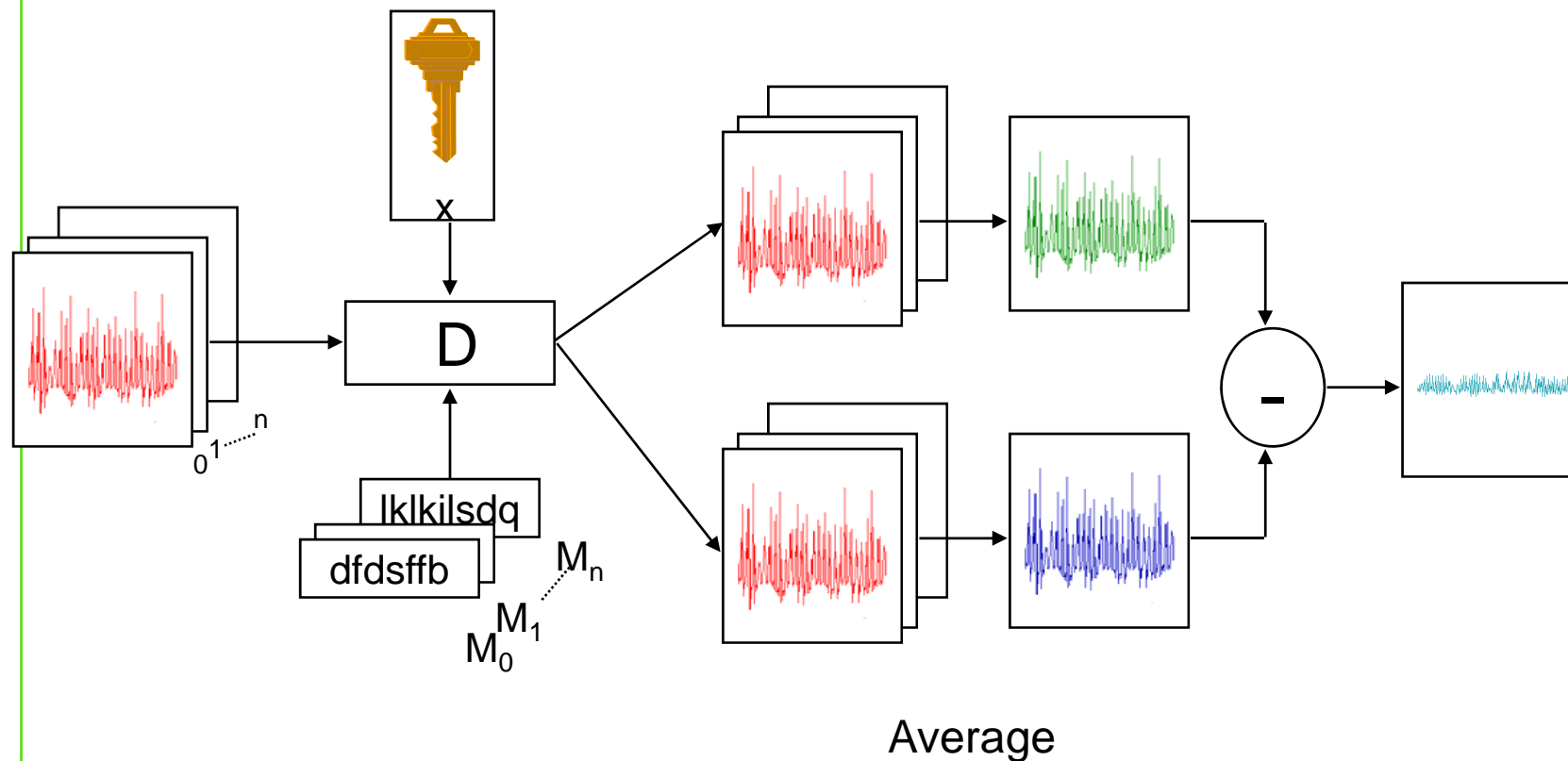
– selection function D :

- sort curves according to M_k or C_k for each value of a $\text{sub}K_i$
- output = image of a target bit of the algorithm



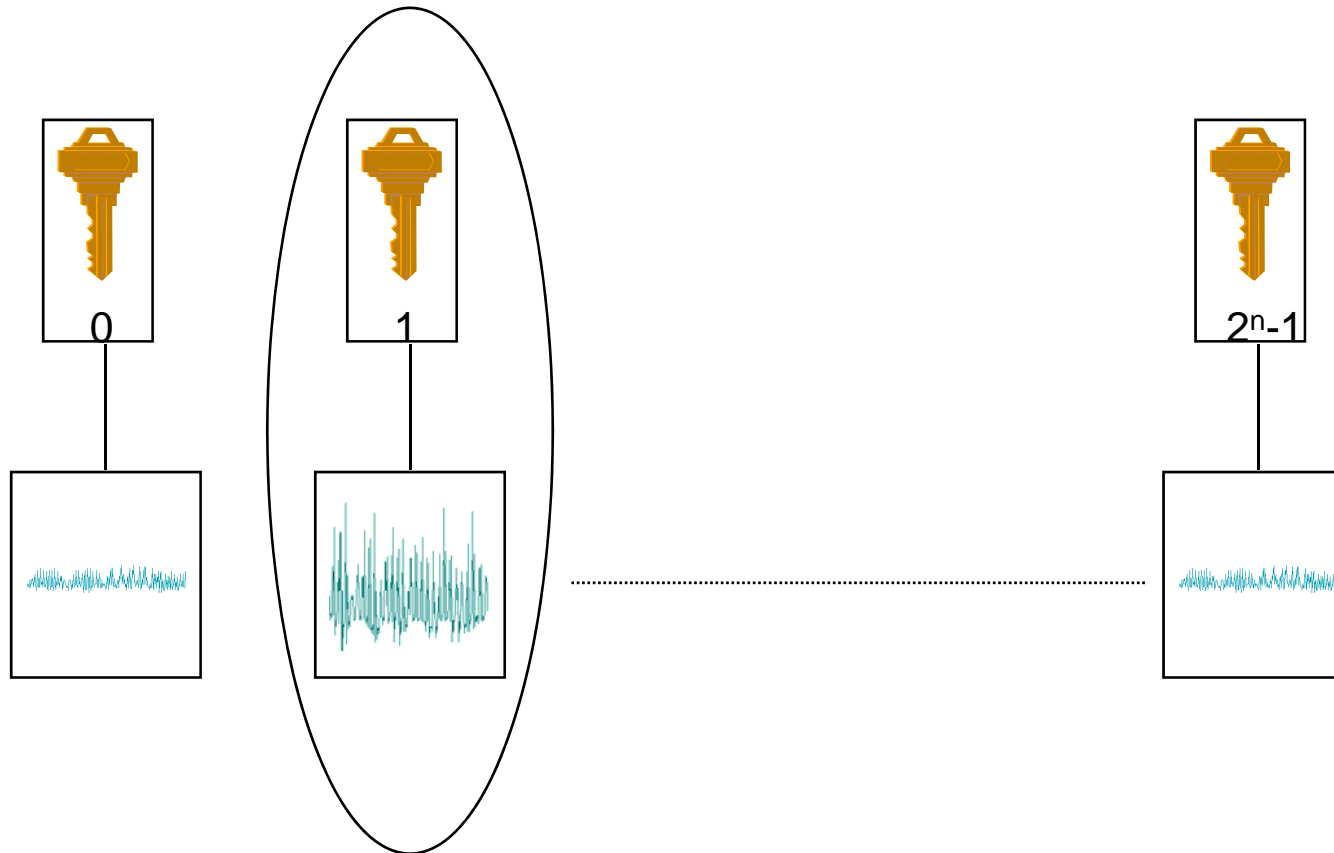
Differential Power Analysis

○ data processing for a value x of a sub K_i :



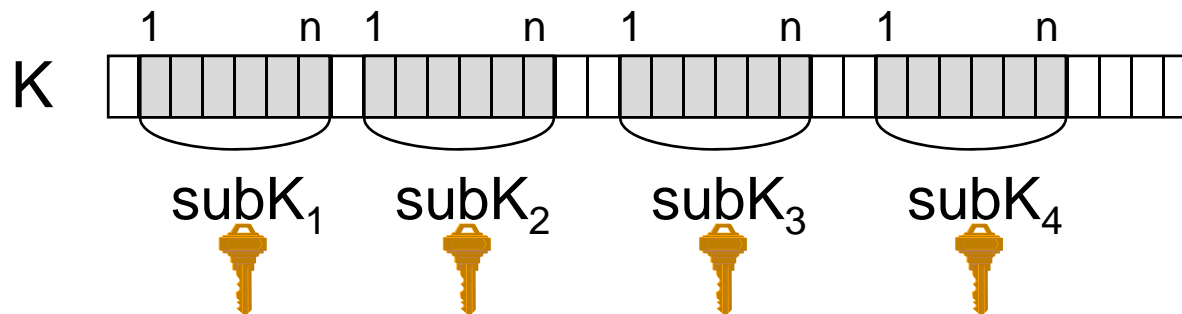
Differential Power Analysis

○ Choosing the right guess



Differential Power Analysis

○ Iterate on all possible sub-keys :

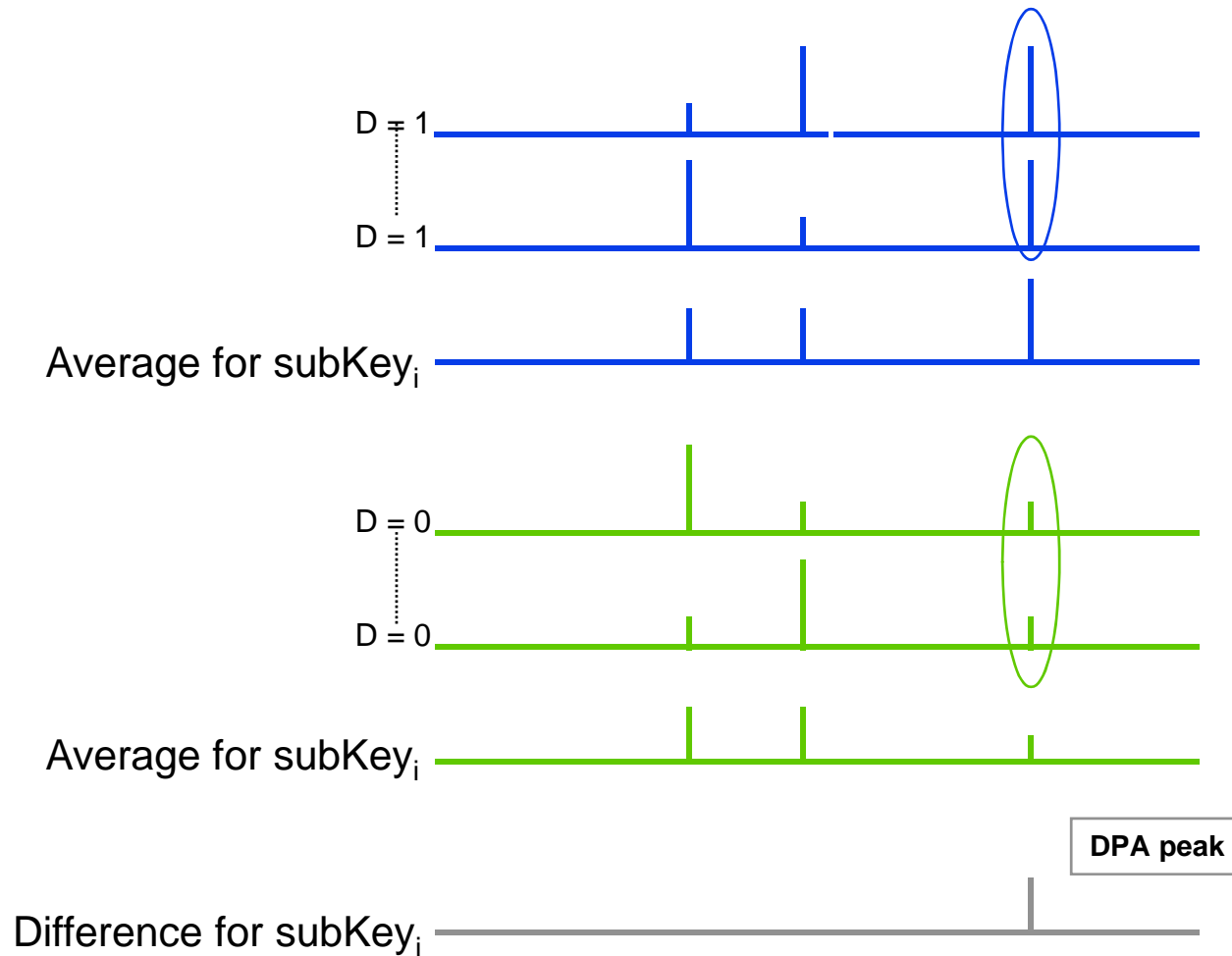


○ find the remaining bits through exhaustive search



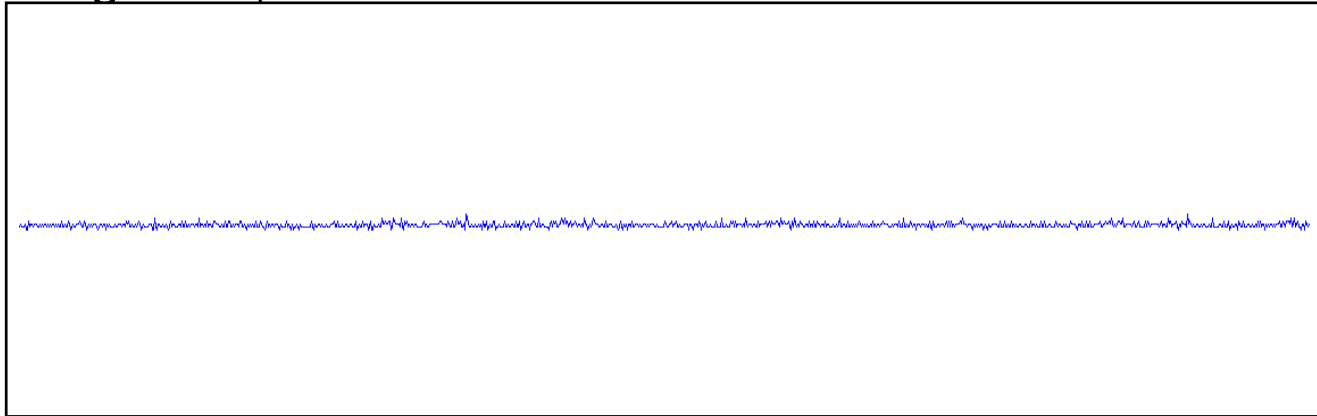
Differential Power Analysis

○ How does it work ?

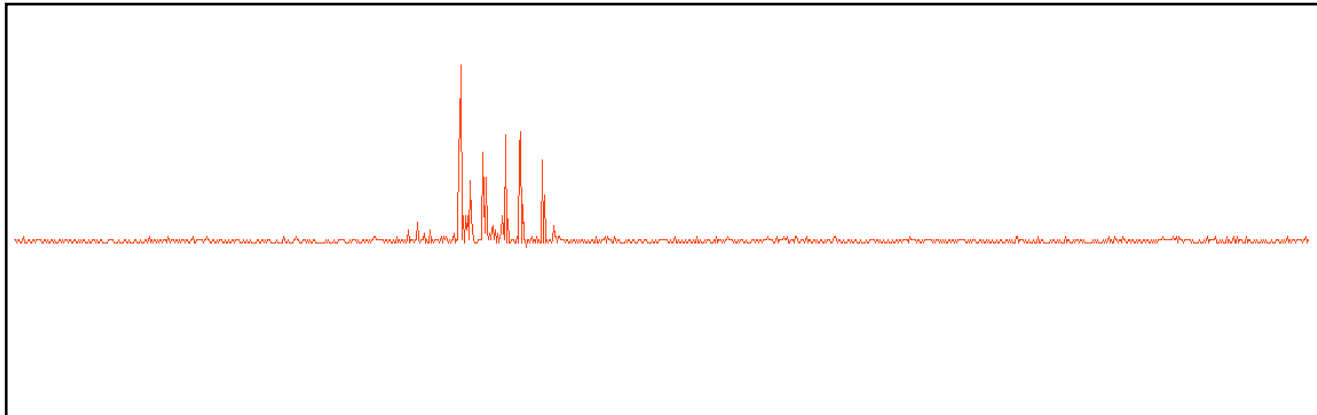


Differential Power Analysis

– wrong subK_i



– right subK_i



Counter-measures

- Add noise
- Scramble power consumption or stabilize it
- Randomize all sensitive data variables with a fresh mask for every execution of an algorithm

- Randomize, randomize, randomize ...
 - Secret keys
 - Messages
 - Private exponents
 - Bases
 - Moduli

Example: DPA on RSA (1)

○ DPA attacks on Modular exponentiation techniques
[CHES'99 MDS]

○ DPA - monobit exponentiation (« Square and Multiply » exponentiation algorithm 2):

○ $A(0) = 1$

○ for $i = 0$ to $m-1$

- $A(0) = A(0)^2 \bmod N$
- $A(1) = A(0) * X \bmod N$
- $A(0) = A(d_i)$

○ next

○ return $A(0)$

Example: DPA on RSA (2)

○Hypotheses:

- – attacker has control of inputs to target device.
- – attacker has some knowledge of the implementation hence can compute intermediate results using an offline simulation

○Secret exponent extracted bit by bit

○for $i = 0$ to m

- – set $d' = d_0 \dots d_{i-1} || 1$ and $d' = d_0 \dots d_{i-1} || 0$
- – accept d' which produces best correlation/match with the offline calculation.

Conclusion on Power Analysis Attacks

- Naïve smartcard implementations of cryptosystems can leak secret data.
- Power Analysis Attacks
 - – target symmetric and asymmetric cryptosystems
 - – practical, 'fast' and cheap
 - – difficult to circumvent
 - – countermeasures may impact efficiency.

First DPA Countermeasures

- Remove data dependent processing.
- Introduce 'noise' to reduce signal to noise ratio.
 - – Vertical noise in the CPU processing
 - – Horizontal noise - insertion of dummy cycles or instructions in the CPU processing may make synchronisation difficult.
 - – Concurrent processing.
 - – Hamming weight balancing.
- All of these can be circumvented given enough processing examples.

DPA Countermeasures for RSA

○ Algebraic approaches (power consumption is not directly correlated with sensitive data).

– – Exponent Blinding

- 'add' a random number, r , to the exponent.
 $d \rightarrow d + r * \phi(N)$

– – Exponent splitting

- E.g. represent $d = d_1 + d_2$ where $d_1 = r$, $d_2 = d - r$ and r is random.

– – Message Blinding

- $M \rightarrow r_1 * M \bmod N$ (r_1 random).

– – Modulus Blinding

- $N \rightarrow r_2 * N$ (r_2 random).

– – Randomised exponentiation

- numerous approaches.

(Differential) Fault Attacks



Introduction

- Fault Attacks were first published as a way of jeopardising computations of cryptographic algorithms (RSA, DSA, DES).
- However, you can imagine to implement fault attacks on other processes inside a microprocessor.
- Fault attacks are real industrial security concern:
 - To pass some certification, like FIPS140-1 level 3 (US government security certification), you should prove that your system resists to fault attacks.

Different types of faults

○ Transients Faults.

- Appear randomly in a system, have various unpredictable causes.

○ Latent or Internal Faults.

- Are the result of hardware or low level software default (floating point unit on Pentium chips,...).
- Rarely controllable

○ Induced Faults.

- Appear after intentional stress (E^2) or hardware "mutilation", can be transient or permanent.
- Sometimes controllable with knowledge of the physical/chemical/electrical behaviour of the chip.

The DFA crisis: 1996

○ September 96

- Attack on RSA CRT by Bellcore (*EuroCrypt'97*)
- Attack improvements by Lenstra

○ October 96

- 18: DFA on DES by Biham et Shamir
- 29: Attack on RSA and ElGamal
- 30: DFA on unknown cryptosystems by Biham & Shamir.
«*Differential Fault Analysis of Secret Key Cryptosystems*»
(*Crypto'97*)

○ November 96

- Attack of CRT on LUC and Demytko by Marc Joye and JJ Quisquater

Attack on standard signature

○ Hypothesis:

- the message m and its signature $s = m^d$ are known
- a fault is injected on one bit i of d
- this results in a wrong signature $s' = m^{d'}$

○ Then :

○ $s'/s = m^{d'-d} = m^{2^i} \mod n$ if bit i was 0

○ or

○ $s'/s = m^{d'-d} = m^{-2^i} \mod n$ if bit i was 1

○ One bit in random position of the secret exponent is discovered every round.

Attack on standard signature

- Faults can be induced on more than one bit, making analysis slightly more difficult.
- This attack is compatible with transient or permanent faults.



Recall on CRT

○ The Chinese Remainder Theorem is used in RSA in order to speed up exponentiation by a factor of 4.

○ Exponentiation is performed in three steps

- $s_p = m^d \bmod p$ is computed (in fact, d_p is used)
- $s_q = m^d \bmod q$ is computed (in fact, d_q is used)
- the signature is recombined with CRT as

$$s = a.s_p + b.s_q \bmod n,$$

○ The constants a and b are precomputed such that

$$\begin{aligned} a &= 1 \bmod p, & b &= 0 \bmod p, \\ a &= 0 \bmod q, & b &= 1 \bmod q. \end{aligned}$$

Attack on CRT exponentiation

○ This attack was first published by Lenstra.

○ Hypothesis:

- s , signature of a message m is known.
- a fault is injected in the exponentiation mod p .

○ Due to error injection, s_p becomes s_p'

$$s' = a.s_p' + b.s_q \text{ mod } n,$$

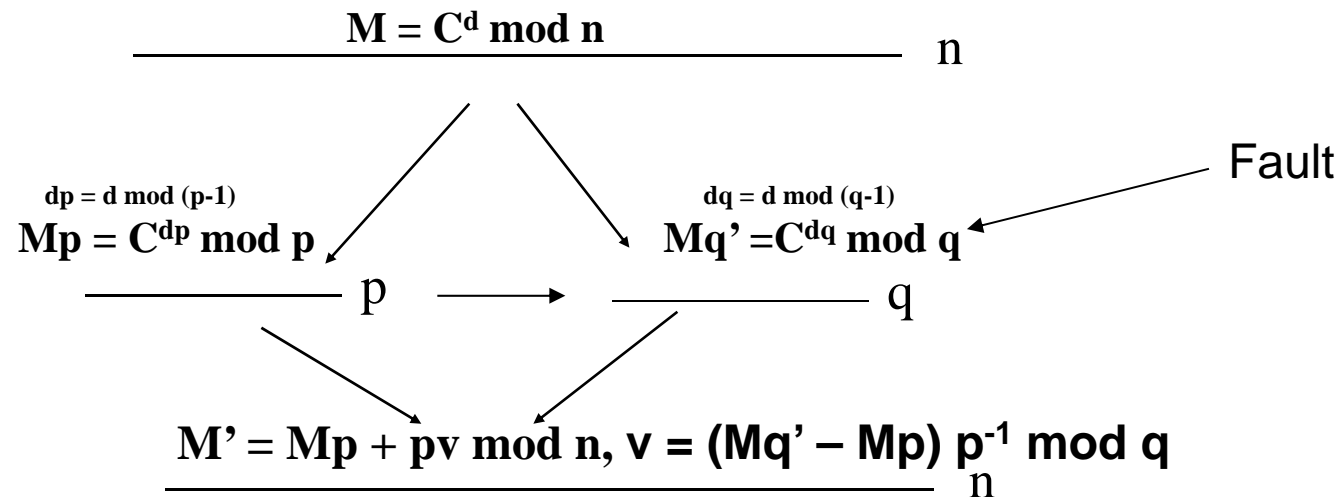
○ $s' - s = (a.s_p' + b.s_q) - (a.s_p + b.s_q) \text{ mod } n$

○ $s' - s = a.(s_p' - s_p) \text{ mod } n$

○ the prime q divides a and can be retrieved by Gcd.

Summary of the Differential Fault Attack (DFA)

An attacker obtains a decryption which is computed in a wrong way.



In the RSA using the CRT, if an attacker can cause a fault for the computation of Mq , then n can be factored by $\gcd(M - M', n) = p$.

Counter-measures on RSA

○ Applicative counter-measures

- Use a random padding with sufficient variability
- Compute the result twice and compare
- Verify that $s^e = m \bmod n$ when e is known
 - e is usually a small number, verification is very fast

○ Algorithmic counter-measures

- One possible counter-measure on RSA-CRT intends to protect both half-exponentiations by:
 - choosing a small random number r
 - computing $s_{pr} = m^d \bmod pr$ et $s_r = m^d \bmod r$
 - checking whether $s_{pr} \bmod r = s_r$

Hardware aspects of fault attacks

○ Flaw injection in this case is a hardly controllable and reproducible process.

○ Implementation is not that easy:

- Stress on memories, during read/writing/retention
 - By means of variations on power supply, frequency
 - Through various types of radiation
- Modifications on hardware mechanism
 - Using probing or FIB
 - Buses
 - Random generators
 - Crypto-coprocessors
 - Hardware DES

Fault attacks and smart cards

- Need expertise in measurements and hardware to implement efficient fault attacks.
- Smart cards give many tools to defeat fault attacks:
 - Tamper-evidence
 - Security sensors, security mechanisms
 - Software counter-measures
 - On algorithms, on secret storage