



HAL
open science

Practical Analysis of RSA Countermeasures Against Side-Channel Electromagnetic Attacks

Guilherme Perin, Laurent Imbert, Lionel Torres, Philippe Maurine

► **To cite this version:**

Guilherme Perin, Laurent Imbert, Lionel Torres, Philippe Maurine. Practical Analysis of RSA Countermeasures Against Side-Channel Electromagnetic Attacks. *CARDIS: Smart Card Research and Advanced Applications*, Nov 2013, Berlin, Germany. pp.200-215, 10.1007/978-3-319-08302-5_14. lirmm-01096070

HAL Id: lirmm-01096070

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01096070>

Submitted on 16 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Practical Analysis of RSA Countermeasures Against Side-Channel Electromagnetic Attacks

[Guilherme Perin](#), Laurent Imbert, Lionel Torres and Philippe Maurine

November 28th, 2013

CARDIS 2013 – 12th Smart Card Research and Advanced Application Conference



Motivation

- RSA is a continuing subject of many side-channel attacks
- Is there a combination of countermeasures which provides sufficient protection against most advanced side-channel attacks?
 - Simple and Collisions-based Attacks
 - Differential and Correlation Analyses
 - Single Execution Attacks on Exponentiations
- Different levels of countermeasures

Agenda



- Countermeasures
- RNS-based RSA
- The Proposed Hardware
- Robustness Against Electromagnetic Analysis:
 - Collision-based attacks
 - Correlation Analyses
 - EM Analysis vs Hardware Countermeasures



RSA: Countermeasures

1. Algorithmic: Blinded Exponentiation

$$N = p \times q$$

$$\phi(N) = (p - 1)(q - 1)$$

$$c = m^e \bmod N$$

$$er = e + r \cdot \phi(N)$$

Exponent Blinding

$$A_0 = 1 + r_1 \cdot n \bmod r_2 \cdot n$$

$$A_1 = m + r_1 \cdot n \bmod r_2 \cdot n$$

Additive Message Blinding

for $i = t - 1 : 0$

$$A_{er_i} = A_0 \cdot A_1 \bmod N$$

$$A_{er_i} = A_{er_i} \cdot A_{er_i} \bmod N$$

Regular Exponentiation:
Montgomery Ladder

end for



RSA: Countermeasures

2. Hardware

- Minimize the Signal-to-Noise Ratio (SNR)
 - Variable location (localized EM analyses)
 - Clock jitter
 - Dummy cycles
 - Frequency dividers

Single Execution (Trace) Attacks on Exponentiation:

- Horizontal Attacks
- Supervised and Unsupervised Template Attacks



RSA: Countermeasures

3. Arithmetic: The Leak Resistant Arithmetic*

- LRA is a derivative of RNS arithmetic for PKC algorithms;
- RNS is a fast, parallel and natural msg blinding arithmetic;
- Immune to collision, differential and (vertical/horizontal) correlation attacks.
- $C_k^{2k} \approx 2^{2k} / \sqrt{\pi k}$ different representations (k = number of moduli).

All variables are randomized during the exponentiation:

- *Moduli* could be recovered during the *Radix to RNS Conversion*
- For 32 *moduli*: $\text{Prob}[\text{moduli guessed} = \text{moduli hardware}] = 1.65 \cdot 10^{-9}$
- **Preliminar conclusion:** vulnerabilities will be only related to RAM and CPU executions (conditional tests, addressing, etc.)

* J.-C. Bajard, L. Imbert, P.-Y. Liardet, and Y. Teglia, "Leak resistant arithmetic," in *CHES'04, ser. LNCS, vol. 3156*. Springer, 2004, pp. 62–75.

Agenda



- Countermeasures
- **RNS-based RSA**
- The Proposed Hardware
- Robustness Against Electromagnetic Analysis:
 - Collision-based attacks
 - Correlation Analyses
 - EM Analysis vs Hardware Countermeasures



Residue Number System

A integer X is represented according to a base $\mathcal{B} = (b_1, b_2, \dots, b_n)$ of relatively prime integers (*moduli*). Then:

$$\langle X \rangle_{\mathcal{B}} = (x_1, x_2, \dots, x_k)$$

where $x_i = X \bmod b_i$. Then, operations $+$, $-$, \cdot are performed modulo b_i :

$$x_i + y_i \bmod b_i$$

$$x_i - y_i \bmod b_i$$

$$x_i \cdot y_i \bmod b_i$$

Notation: $|X|_{b_i} = X \bmod b_i$



RNS Montgomery Ladder

Data: x in $\mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = (a_1, a_2, \dots, a_k)$, $\mathcal{B} = (b_1, b_2, \dots, b_k)$, $A = \prod_{i=1}^k a_i$,
 $B = \prod_{i=1}^k b_i$, $\gcd(A, B) = 1$, $\gcd(B, N) = 1$ and $e = (e_{n-1} \dots e_1 e_0)_2$.

Result: $z = x^e \pmod N$ in $\mathcal{A} \cup \mathcal{B}$

Pre-Computations: $|AB \pmod N|_{\mathcal{A} \cup \mathcal{B}}$

$$A_0 = MM(1, AB \pmod N, N, \mathcal{A}, \mathcal{B}) \quad (\text{in } \mathcal{A} \cup \mathcal{B})$$

$$A_1 = MM(x, AB \pmod N, N, \mathcal{A}, \mathcal{B}) \quad (\text{in } \mathcal{A} \cup \mathcal{B})$$

for $i = n - 1$ **to** 0 **do**

$$\quad | \quad A_{\overline{e_i}} = MM(A_{\overline{e_i}}, A_{e_i}, N, \mathcal{B}, \mathcal{A}) \quad (\text{in } \mathcal{A} \cup \mathcal{B})$$

$$\quad | \quad A_{e_i} = MM(A_{e_i}, A_{e_i}, N, \mathcal{B}, \mathcal{A}) \quad (\text{in } \mathcal{A} \cup \mathcal{B})$$

end

$$A_0 = MM(A_0, 1, N, \mathcal{B}, \mathcal{A}) \quad (\text{in } \mathcal{A} \cup \mathcal{B})$$

→ Transform the input data $(1, x)$ into the Montgomery domain by inverting \mathcal{A} and \mathcal{B}
In the two calls of MM:

➤ $1.AB.A^{-1} \pmod N = 1.A^2B \pmod N = B \pmod N$

➤ $x.AB.A^{-1} \pmod N = x.A^2B \pmod N = x.B \pmod N$



Montgomery Multiplication

Classical arithmetic: (Montgomery Constant $R=2^k$, k is the bit-length)

$$q = x.y.(-N^{-1}) \bmod R$$

$$s = \frac{x.y+q.N}{R}$$

Return $x.y.R^{-1} \bmod N$

Residue Number System: (Montgomery Constant $B = \prod_{i=1}^k b_i$, k is the number of moduli in base $\mathcal{B} = (b_1, \dots, b_i)$)

Base \mathcal{A}

$$q_{\mathcal{A}}$$

$$w_{\mathcal{A}} = (x_{\mathcal{A}}.y_{\mathcal{A}} + q_{\mathcal{A}}.N_{\mathcal{A}})/B$$

Base Extension

←

→

Base \mathcal{B}

$$q_{\mathcal{B}} = x_{\mathcal{B}}.y_{\mathcal{B}}.|-N^{-1}|_{\mathcal{B}}$$

$$w_{\mathcal{B}}$$

Return $x.y.B^{-1} \bmod N$

RNS Montgomery Multiplication



$$s_B = x_B \cdot y_B$$

$$s_A = x_A \cdot y_A$$

$$q_B = s_B \cdot | - N^{-1} |_B$$

$$q_A \leftarrow q_B$$

BE1

$$w_A = (s_A + q_A \cdot N_A) \cdot B^{-1}$$

$$w_B \leftarrow w_A$$

BE2

$$f = \left\lfloor \left(2^{m-1} + \sum_{i=1}^k |w \cdot A_i^{-1}|_{a_i} \right) / 2^m \right\rfloor$$

$$w_B = \left| \sum_{i=1}^k |w|_{a_i} \cdot A_i |_B - |f \cdot A|_B \right|$$

Fast Approximation Base Extension (CRT):

$$X = \sum_{i=1}^k B_i |x_i B_i^{-1}|_{b_i} - f \cdot B \quad B_i = \frac{B}{b_i}$$

$$|X|_A = \left| \sum_{i=1}^k B_i |x_i B_i^{-1}|_{b_i} \right|_{a_i} - f \cdot |B|_{a_i}$$

$$f = \left\lfloor \left(\sum_{i=1}^k |q \cdot B_i^{-1}|_{b_i} \right) / 2^m \right\rfloor$$

$$q_A = \left| \sum_{i=1}^k |q|_{b_i} \cdot B_i |_A - |f \cdot B|_A \right|$$

RNS Montgomery Multiplication

Improved Version [*]



$$s_B = x_B \cdot y_B$$

$$s_A = x_A \cdot y_A$$

$$q_B = |s_B \cdot B_i^{-1} - N^{-1}|_B$$

$$f = \left\lfloor \left(\sum_{i=1}^k |q|_{b_i} \right) / 2^m \right\rfloor$$

$$w_A = s_A \cdot B^{-1} + \sum_{i=1}^k |q|_{b_i} \cdot B_i \cdot N \cdot B^{-1} |_{\mathcal{A}} - |f \cdot B \cdot N \cdot B^{-1}|_{\mathcal{A}}$$

$$q_A = |w \cdot A_i^{-1}|_{\mathcal{A}}$$

$$f = \left\lfloor \left(2^{m-1} + \sum_{i=1}^k |q|_{a_i} \right) / 2^m \right\rfloor$$

$$w_B = \left| \sum_{i=1}^k |w|_{a_i} \cdot A_i \right|_B - |f \cdot A|_B$$

	RNS MM	RNS MM Improved
Pre-computations	$2k^2 + 7k$	$2k^2 + 5k$
RNS multiplications	$2k^2 + 7k$	$2k^2 + 5k$

* F. Gandino, F. Lamberti, P. Montuschi, and J.-C. Bajard, "A general approach for improving RNS montgomery exponentiation using pre-processing," in *ARITH20*. IEEE Computer Society, 2011, pp. 195–204.

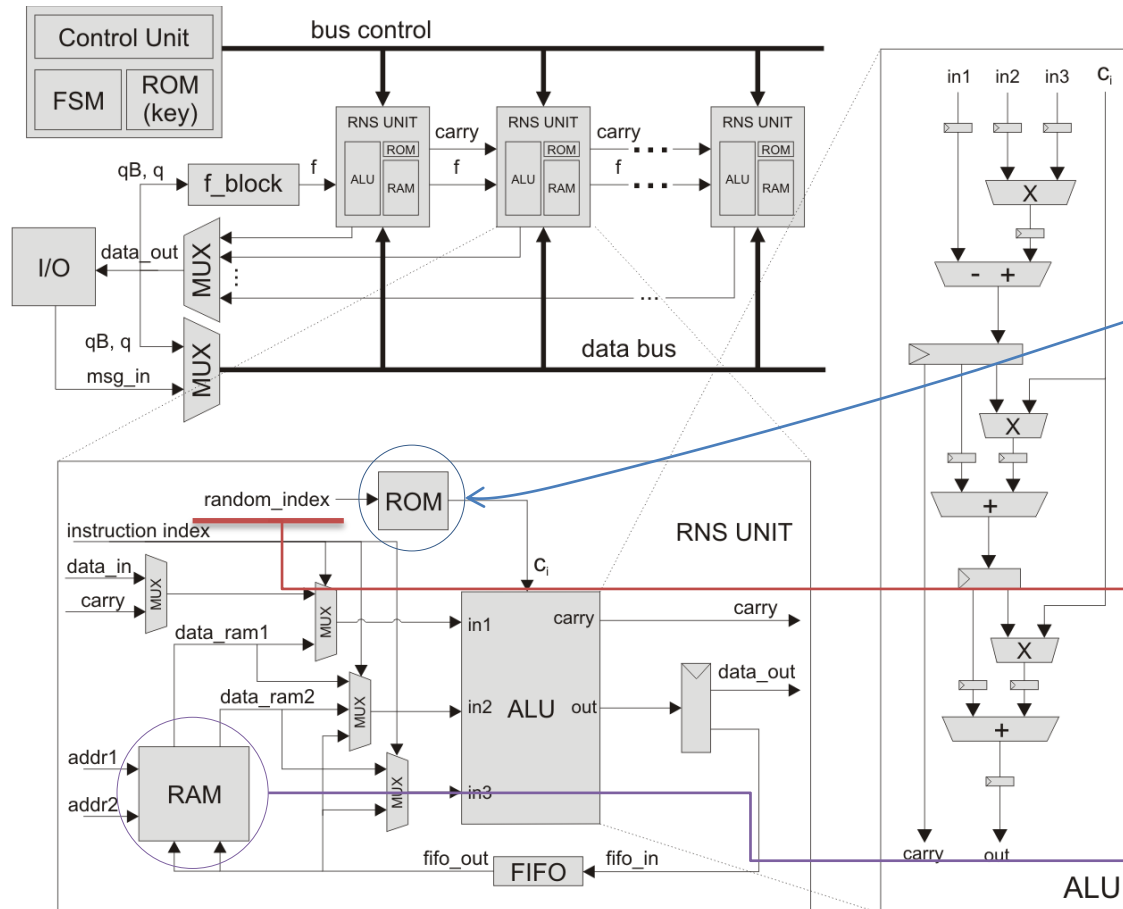
Agenda



- Countermeasures
- RNS-based RSA
- **The Proposed Hardware**
- Robustness Against Electromagnetic Analysis:
 - Collision-based attacks
 - Correlation Analyses
 - EM Analysis vs Hardware Countermeasures



Proposed and Evaluated Hardware



Moduli as Pseudo-Mersenne Number:

$$b_i = 2^w - c_i$$

$$x \leftarrow (x \bmod 2^w) + c_i \cdot (x/2^w)$$

(two times)

All RNS Units operate in all RNS moduli

All RNS Units store All pre-computed values

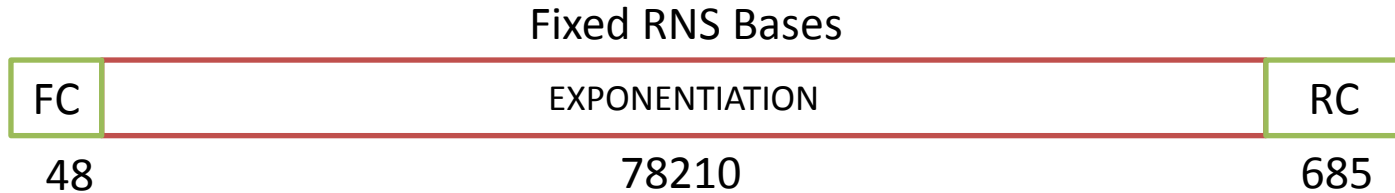
With Fixed Bases (32 moduli, 32 bits): pre-computations need **8.5 kB**

With Randomized Bases (32 moduli, 32 bits): pre-computations need **118 kB**

LRA Precomputations



- RNS Bases are randomized once before each exponentiation.
- Clock cycles (512 bits):



FC = Radix to RNS
RC = RNS to Radix

Clock Cycles Overhead: 1%

Memory Overhead: 92%

Agenda



- Countermeasures
- RNS-based RSA
- The Proposed Hardware
- **Robustness Against Electromagnetic Analysis:**
 - Collision-based attacks
 - Correlation Analyses
 - EM Analysis vs Hardware Countermeasures



Collision Attacks

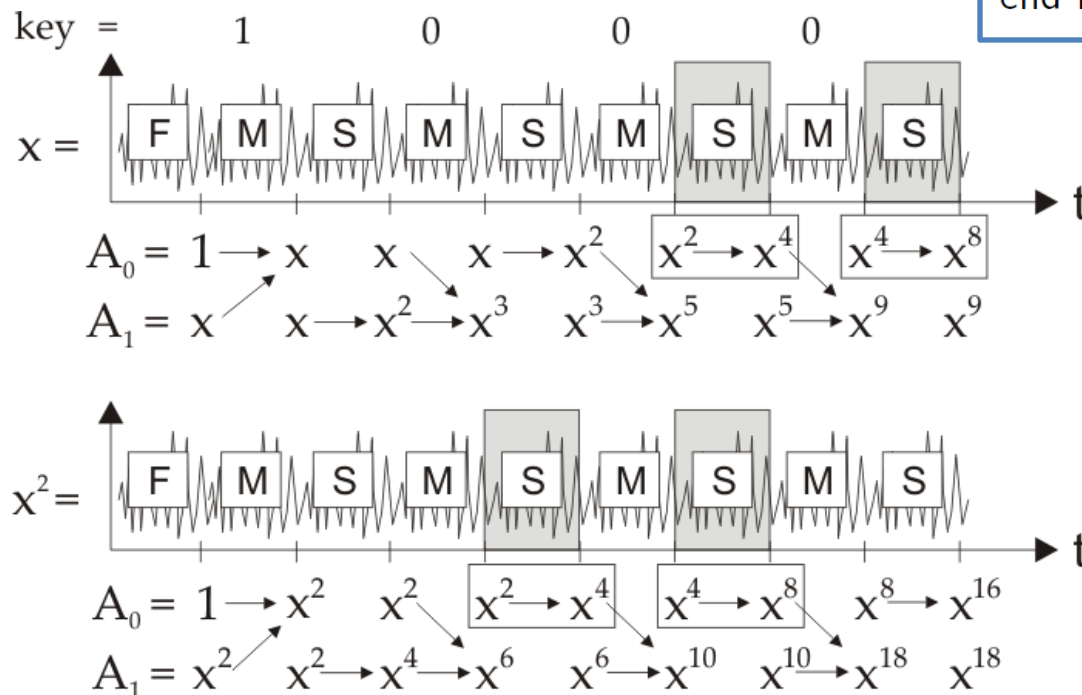
- Identify redundant operations by collecting two (averaged or not) traces for different chosen-message pairs:
 - (x, x^2) : Doubling Attack
 - $(x, -x)$: Yen's et al Attack
 - (x^α, y^β) : Homma's et al Attack

for $i = t - 1 : 0$

$$A_{\overline{er}_i} = A_0 \cdot A_1 \text{ mod } N$$

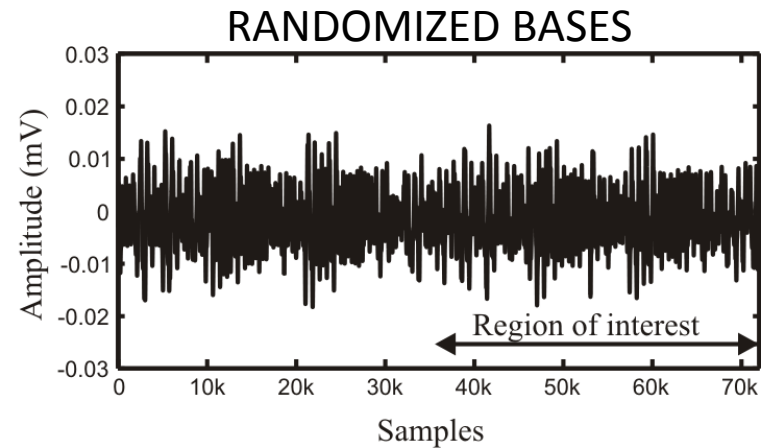
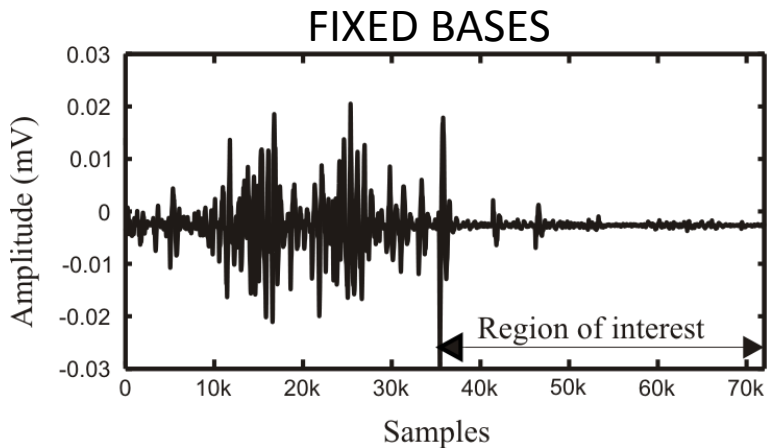
$$A_{er_i} = A_{er_i} \cdot A_{er_i} \text{ mod } N$$

end for





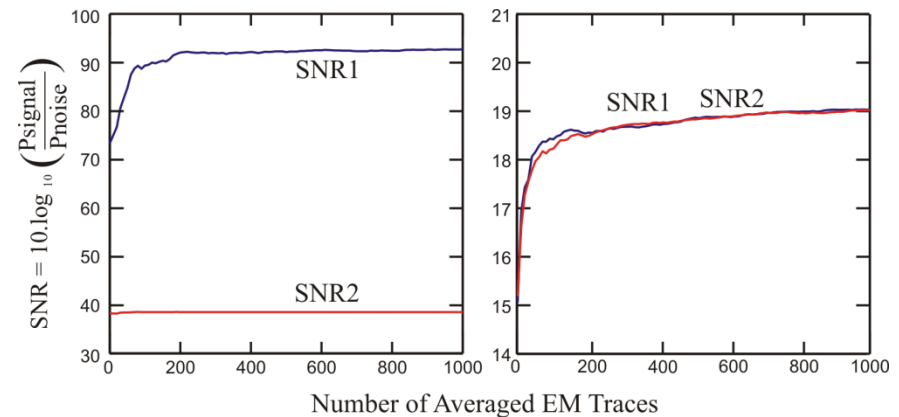
LRA vs Collision Attacks



$EM(T_S, x, e_i)$ = squaring EM trace at e_i
 $EM(T_S, x, e_{i-1})$ = squaring EM trace at e_{i-1}

$$SNR = 20 \cdot \log_{10} \frac{P_{signal}}{P_{noise}} =$$

$$= 20 \cdot \log_{10} \frac{\sigma^2(EM(T_S, x, e_{i-1}))}{\sigma^2(EM(T_S, x, e_{i-1}) - EM(T_S, x^2, e_i))}$$





Correlation Attacks

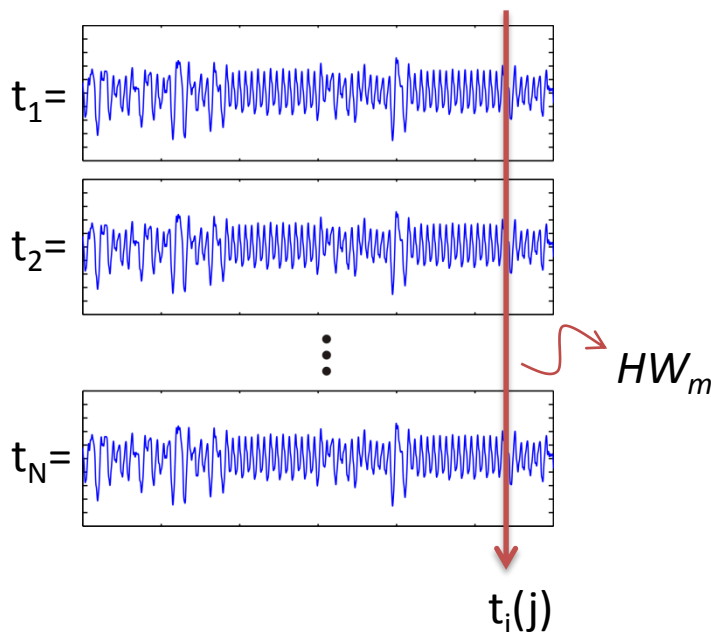
HW_m = Hamming Weight of a Data m

$t_i(j)$ = sample j of a trace i

$$\rho(HW_m, t_i(j)) = \frac{\text{cov}(HW_m, t_i(j))}{\sqrt{\text{var}(HW_m)\text{var}(t_i(j))}}$$

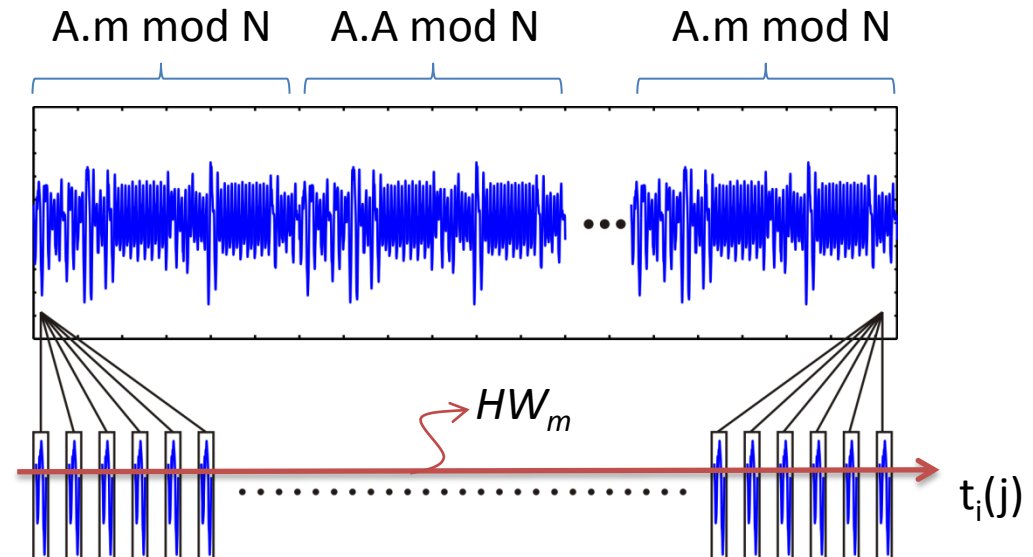
Vertical:

- Correlate HW x Trace



Horizontal (Immune to Exponent Blinding):

- Correlate HW x Trace
- Correlate Trace x Trace*



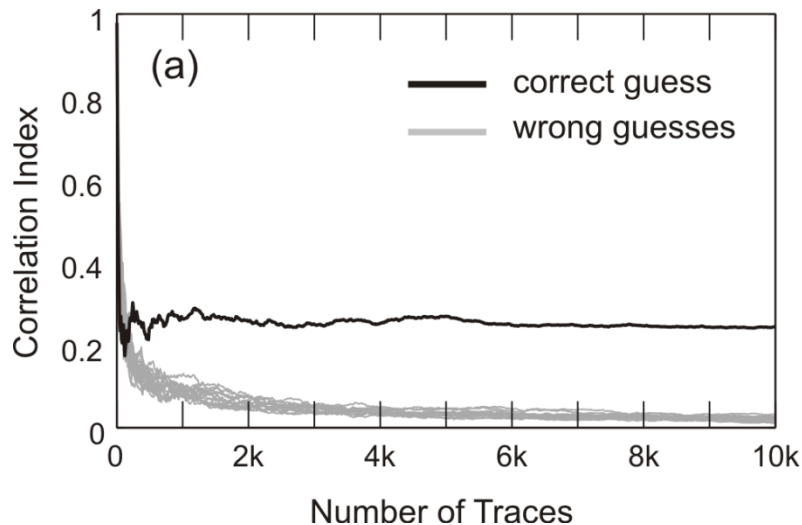
*C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Rousselet and V. Verneuil, "ROSETTA for Single Trace Analysis," in *INDOCRYPT 2012*;



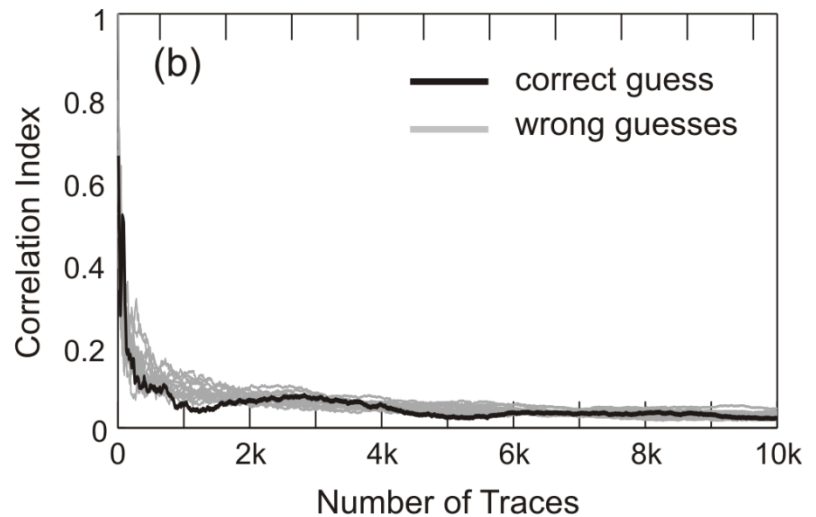
LRA vs Correlation Attacks

Vertical:

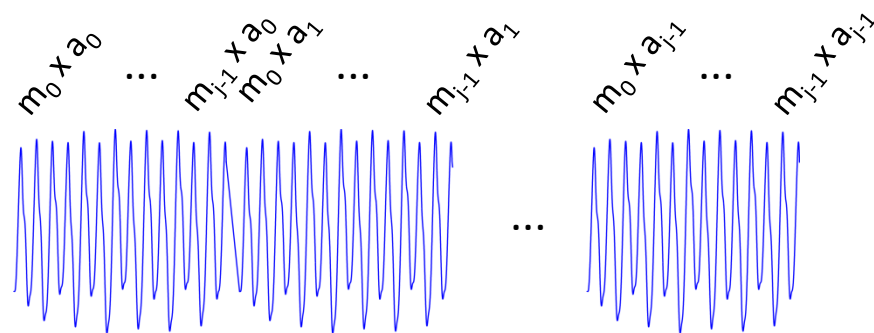
FIXED BASES



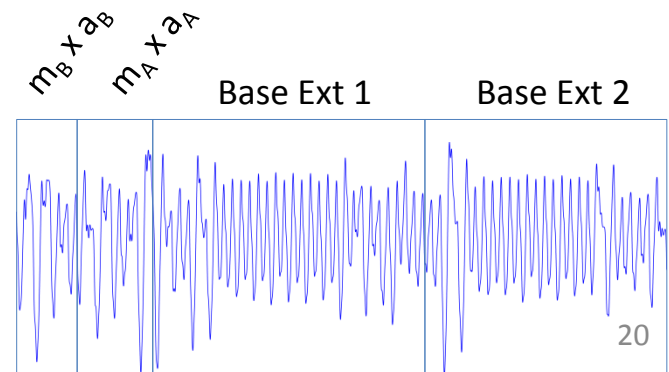
RANDOMIZED BASES



Horizontal: Proposed for Long-Integer Multiplications



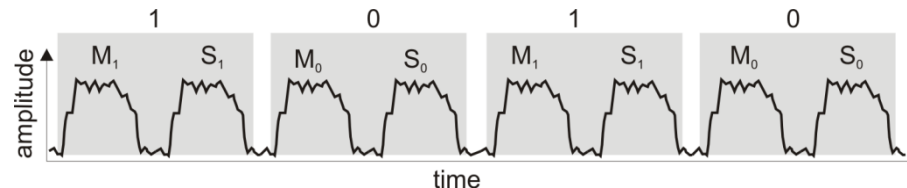
Horizontal: RNS Multiplications





Single Execution Attacks

- Why? Exponentiation is randomized.
 - Exponent: $er = e + r.\phi(N)$
 - Message: Leak Resistant Arithmetic
- Which attacks?
 - Horizontal attacks;
 - Supervised, semi-supervised and unsupervised template attacks:



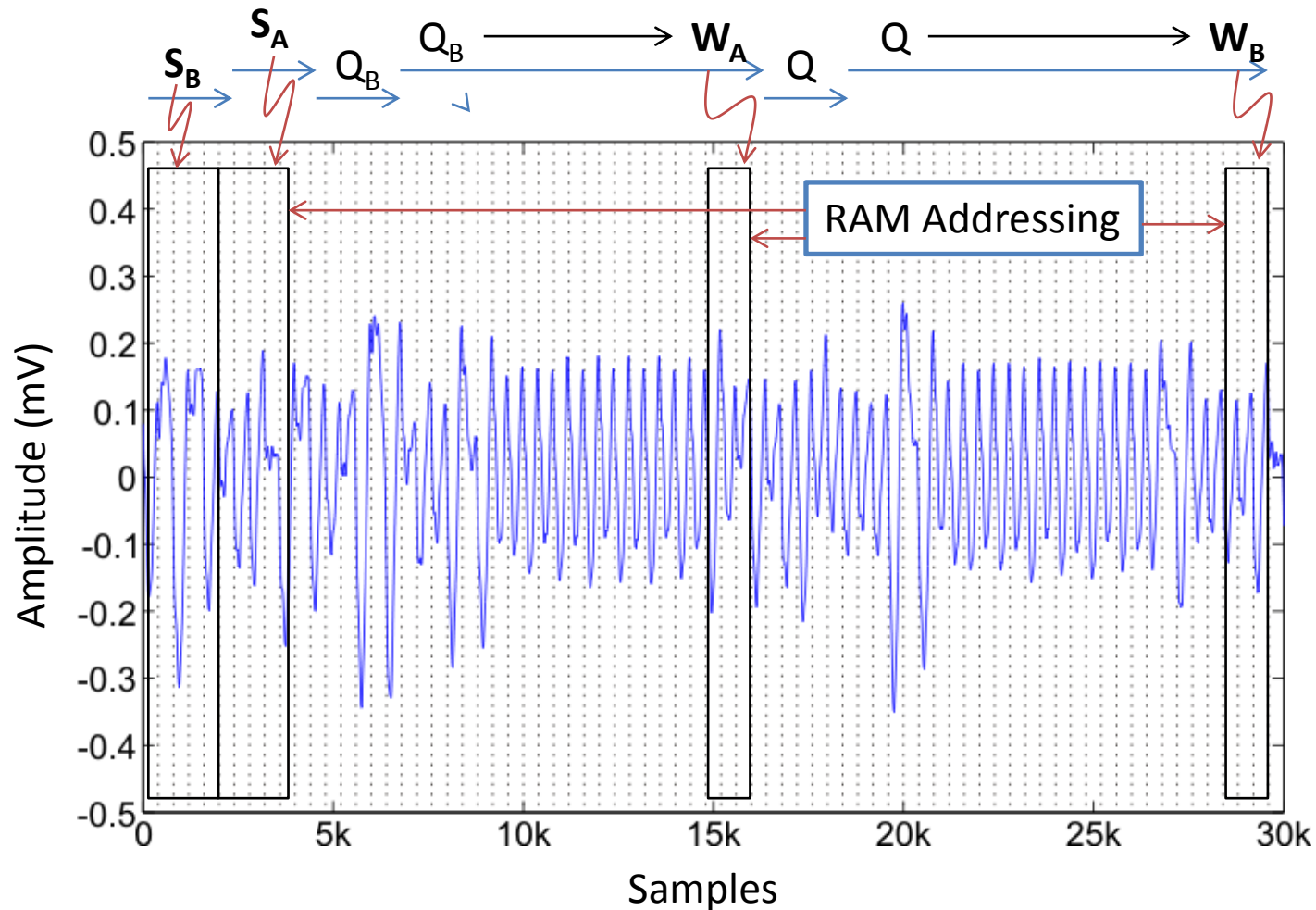
□ Montgomery Ladder -> Find the means (μ) and std dev (σ) of two classes:

- $N(\mu_{(m0)}, \sigma_{(m0)})$: mean and std dev of a **multiplication** when exponent bit is **0**
- $N(\mu_{(m1)}, \sigma_{(m1)})$: mean and std dev of a **multiplication** when exponent bit is **1**
- $N(\mu_{(s0)}, \sigma_{(m0)})$: mean and std dev of a **squaring** when exponent bit is **0**
- $N(\mu_{(s1)}, \sigma_{(m1)})$: mean and std dev of a **squaring** when exponent bit is **1**



Single Execution Attacks on RNS Exponentiation

RAM, CPU: **exponent-dependent activities**





What are the RAM leakages?

- Fixed Exponent:
 - Averaged EM traces: **remove the data dependency**

$$\overline{m_0} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} m_i(0)$$

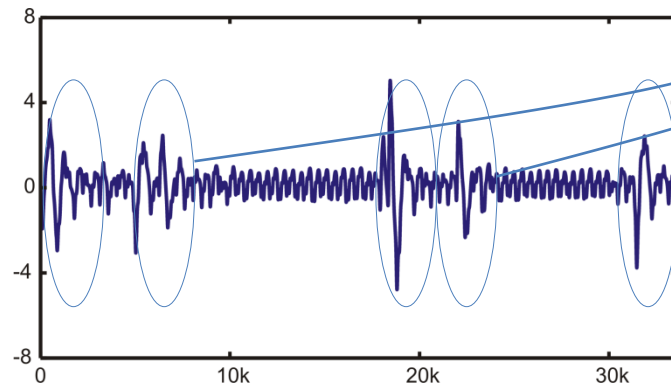
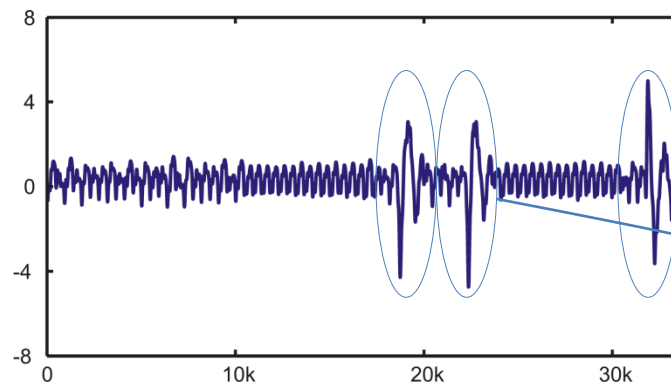
$$\overline{m_1} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} m_i(1)$$

$$\overline{s_0} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} s_i(0)$$

$$\overline{s_1} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} s_i(1)$$

$$\overline{m_0} - \overline{m_1} =$$

$$\overline{s_0} - \overline{s_1} =$$



Conditional Tests
RAM Addressing



RAM Addressing Randomization

Intermediate results are never stored in same positions:

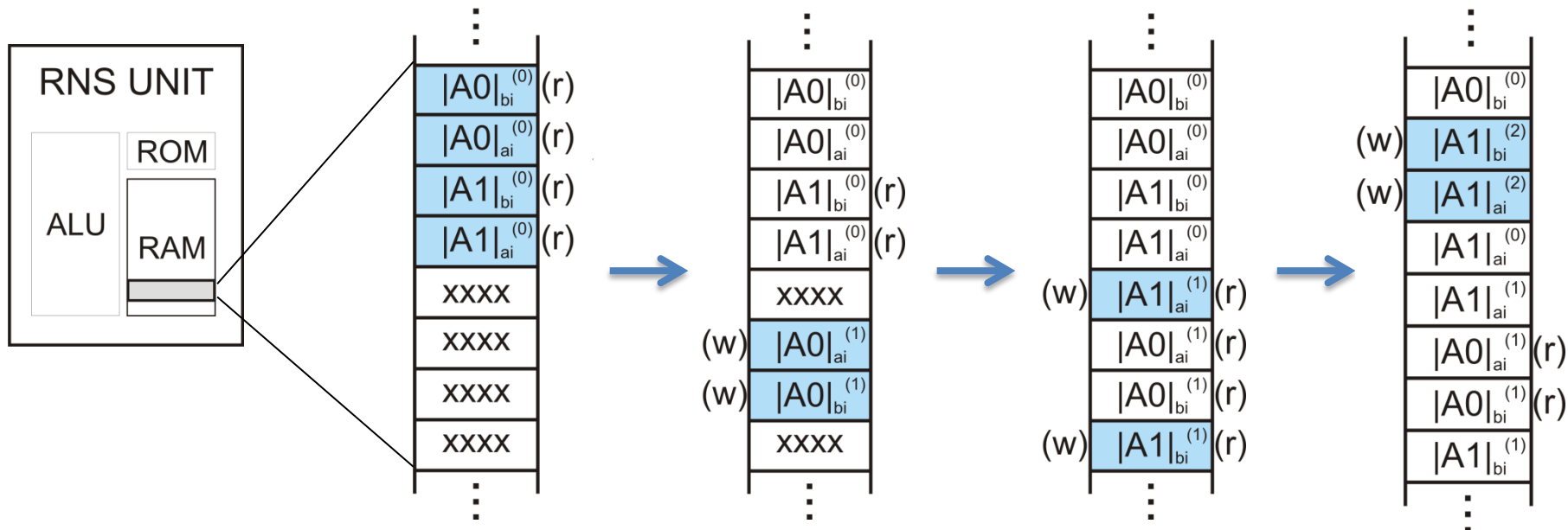
```

for  $i = t - 1 : 0$ 
   $A_{er_i} = A_0.A_1 \text{ mod } N$ 
   $A_{er_i} = A_{er_i}.A_{er_i} \text{ mod } N$ 
end for
  
```

Multiplication(1):
 $A0 = A0.A1 \text{ mod } N$
read(A0, A1)
write(A0)

Squaring(1):
 $A1 = A1.A1 \text{ mod } N$
read(A1)
write(A1)

Multiplication(0):
 $A1 = A0.A1 \text{ mod } N$
read(A1)
write(A1)

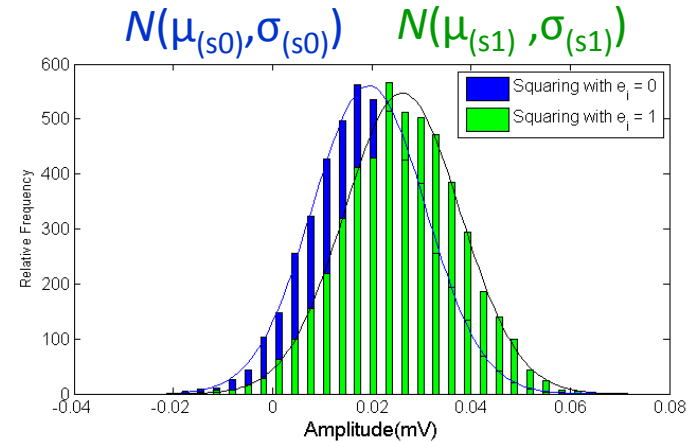
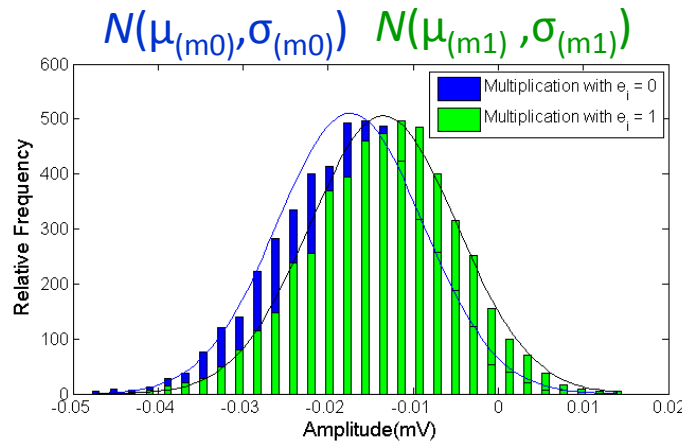




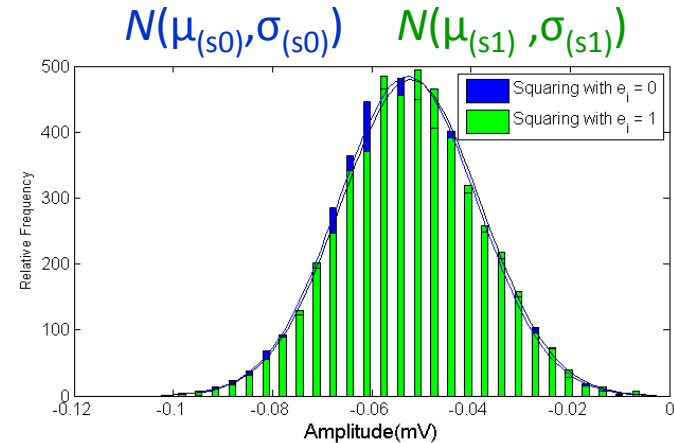
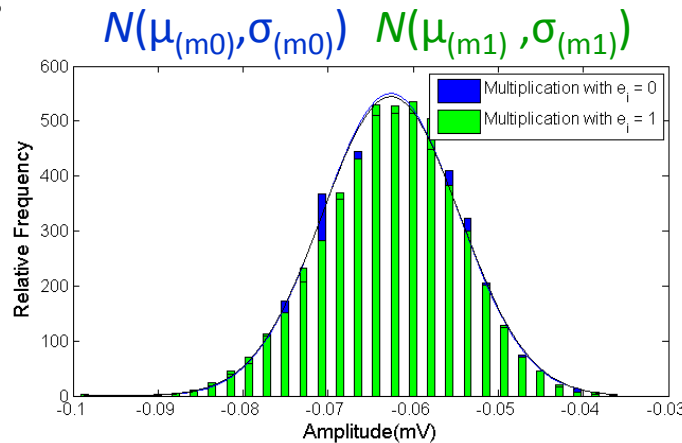
RAM Addressing Randomization

We took a fixed sample point t_i representing the RAM addressing (writing):

Unprotected:



Protected:





Conclusions

- We evaluated the combination of **Algorithmic + Arithmetic + Hardware** countermeasures against side-channel EM Analyses.
- LRA is a robust solution against simple, collisions, correlation and horizontal analyses (HW vs Trace).
- The major impact of LRA countermeasure is given in terms of memory (92%), not time (1%).
- Hardware countermeasures reduce the efficiency of single executions (trace) analysis on exponentiations (reduce the SNR).

Future Works:

- We will evaluate the effect of **Algorithmic + Arithmetic + Hardware** countermeasures against supervised and unsupervised template attacks.



Thank you for your attention!

QUESTIONS?

perin@lirmm.fr