Practical Analysis of RSA Countermeasures Against Side-Channel Electromagnetic Attacks

Guilherme Perin, Laurent Imbert, Lionel Torres, Philippe Maurine

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

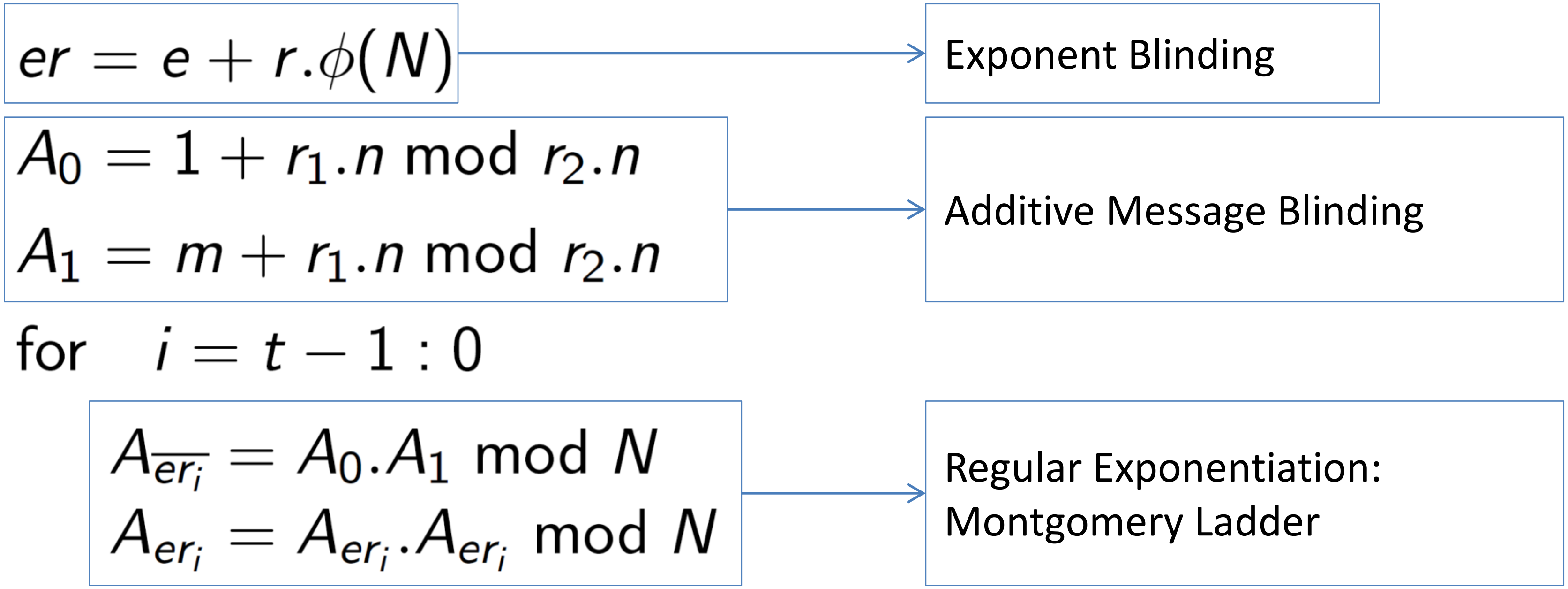
RSA: Countermeasures

1. Algorithmic: Blinded Exponentiation

$$N = p \times q$$

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

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



```
$er = e + r.\phi(N)$
$A_0 = 1 + r_1.n \bmod r_2.n$
$A_1 = m + r_1.n \bmod r_2.n$
for    $i = t-1 : 0$
    $A_{\overline{er_i}} = A_0.A_1 \bmod N$
    $A_{er_i} = A_{er_i}.A_{er_i} \bmod N$
end for
```

Exponent Blinding

Additive Message Blinding

Regular Exponentiation: Montgomery Ladder

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*: Prob[*moduli guessed = moduli hardware*] = $1.65.10^{-9}$
- **Preliminar conclusion:** vulnerabilites 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 $\mathcal{B} = (b_1, b_2, ..., b_n)$ of relatively prime integers (*moduli*). Then:

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

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

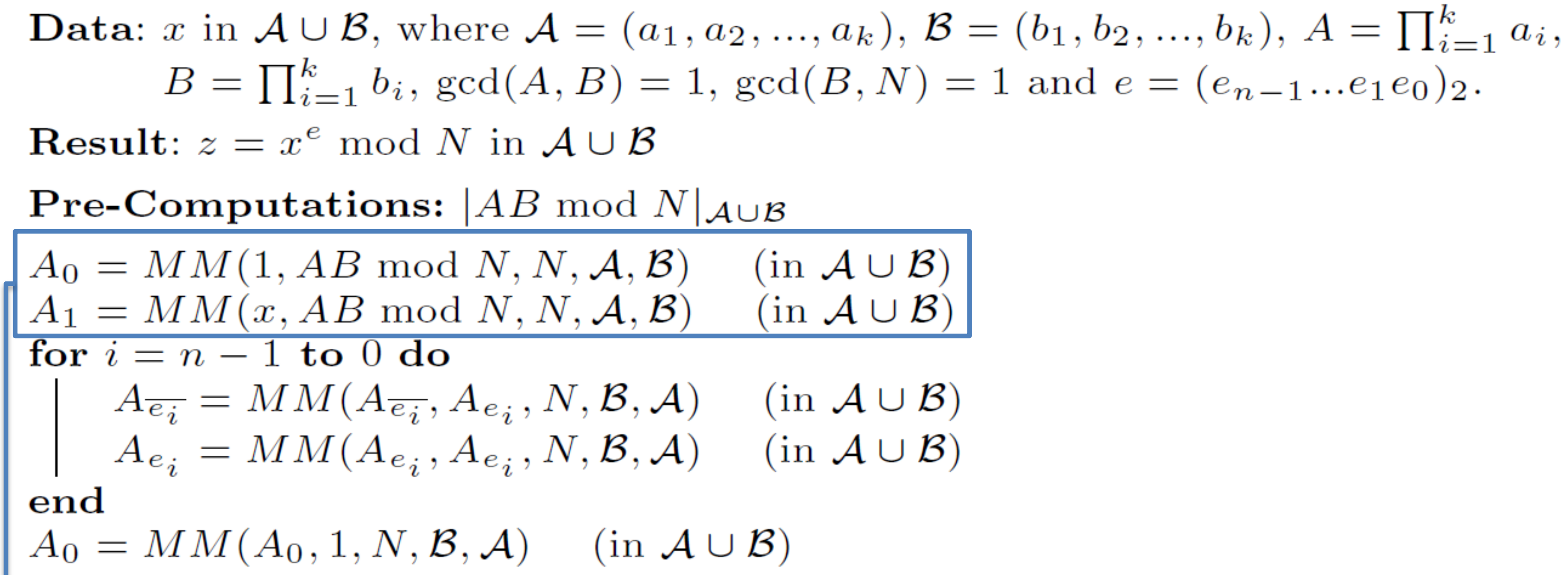
$$x_i + y_i \mod b_i$$

$$x_i - y_i \mod b_i$$

$$x_i . y_i \mod b_i$$

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

RNS Montgomery Ladder



```
Data: $x$ in $\mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = (a_1, a_2, \ldots, a_k)$, $\mathcal{B} = (b_1, b_2, \ldots, b_k)$, $A = \prod_{i=1}^{k} a_i$,
      $B = \prod_{i=1}^{k} b_i$, $\gcd(A, B) = 1$, $\gcd(B, N) = 1$ and $e = (e_{n-1} \ldots e_1 e_0)_2$.
Result: $z = x^e \bmod N$ in $\mathcal{A} \cup \mathcal{B}$
Pre-Computations: $|AB \bmod N|_{\mathcal{A} \cup \mathcal{B}}$
$A_0 = MM(1, AB \bmod N, N, \mathcal{A}, \mathcal{B})$    (in $\mathcal{A} \cup \mathcal{B}$)
$A_1 = MM(x, AB \bmod N, N, \mathcal{A}, \mathcal{B})$    (in $\mathcal{A} \cup \mathcal{B}$)
for $i = n - 1$ to $0$ do
    $A_{\overline{e_i}} = MM(A_{\overline{e_i}}, A_{e_i}, N, \mathcal{B}, \mathcal{A})$    (in $\mathcal{A} \cup \mathcal{B}$)
    $A_{e_i} = MM(A_{e_i}, A_{e_i}, N, \mathcal{B}, \mathcal{A})$    (in $\mathcal{A} \cup \mathcal{B}$)
end
$A_0 = MM(A_0, 1, N, \mathcal{B}, \mathcal{A})$    (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} mod N= 1.A^{2}B mod N = B mod N
- x.AB.A^{-1} mod N= x.A^{2}B mod N = x.B mod 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} mod N

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

Base $\mathcal{A}$	Base Extension	Base $\mathcal{B}$
$q_\mathcal{A}$	$\leftarrow$	$q_\mathcal{B} = x_\mathcal{B}.y_\mathcal{B}.\vert -N^{-1}\vert_\mathcal{B}$
$w_\mathcal{A} = (x_\mathcal{A}.y_\mathcal{A} + q_\mathcal{A}.N_\mathcal{A})/B$	$\rightarrow$	$w_\mathcal{B}$

Return x.y.B^{-1} mod N

RNS Montgomery Multiplication

$$s_{\mathcal{B}} = x_{\mathcal{B}} . y_{\mathcal{B}}$$

$$s_{\mathcal{A}} = x_{\mathcal{A}} . y_{\mathcal{A}}$$

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

$$\boxed{q_{\mathcal{A}} \leftarrow q_{\mathcal{B}}}$$

$$w_{\mathcal{A}} = (s_{\mathcal{A}} + q_{\mathcal{A}} . N_{\mathcal{A}}) . B^{-1}$$

$$\boxed{w_{\mathcal{B}} \leftarrow w_{\mathcal{A}}}$$

Fast Approximation Base Extension (CRT):

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

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

BE1

$$\boxed{\begin{array}{l} f = \left\lfloor \left(\sum_{i=1}^{k} |q.B_i^{-1}|_{b_i} \right) / 2^m \right\rfloor \\ q_{\mathcal{A}} = | \sum_{i=1}^{k} |q|_{b_i} . B_i |_{\mathcal{A}} - |f.B|_{\mathcal{A}} \end{array}}$$

BE2

$$\boxed{\begin{array}{l} f = \left\lfloor \left(2^{m-1} + \sum_{i=1}^{k} |w.A_i^{-1}|_{a_i} \right) / 2^m \right\rfloor \\ w_{\mathcal{B}} = | \sum_{i=1}^{k} |w|_{a_i} . A_i |_{\mathcal{B}} - |f.A|_{\mathcal{B}} \end{array}}$$

RNS Montgomery Multiplication Improved Version [*]

$$s_{\mathcal{B}} = x_{\mathcal{B}}.y_{\mathcal{B}}$$

$$s_{\mathcal{A}} = x_{\mathcal{A}}.y_{\mathcal{A}}$$

$$q_{\mathcal{B}} = |s_{\mathcal{B}}.B_i^{-1}. - N^{-1}|_{\mathcal{B}}$$

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

$$w_{\mathcal{A}} = s_{\mathcal{A}}.B^{-1} + \sum_{i=1}^{k} |q|_{b_i}.B_i.N.B^{-1}|_{\mathcal{A}} - |f.B.N.B^{-1}|_{\mathcal{A}}$$

$$q_{\mathcal{A}} = |w.A_i^{-1}|_{\mathcal{A}}$$

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

$$w_{\mathcal{B}} = |\sum_{i=1}^{k} |w|_{a_i}.A_i|_{\mathcal{B}} - |f.A|_{\mathcal{B}}$$

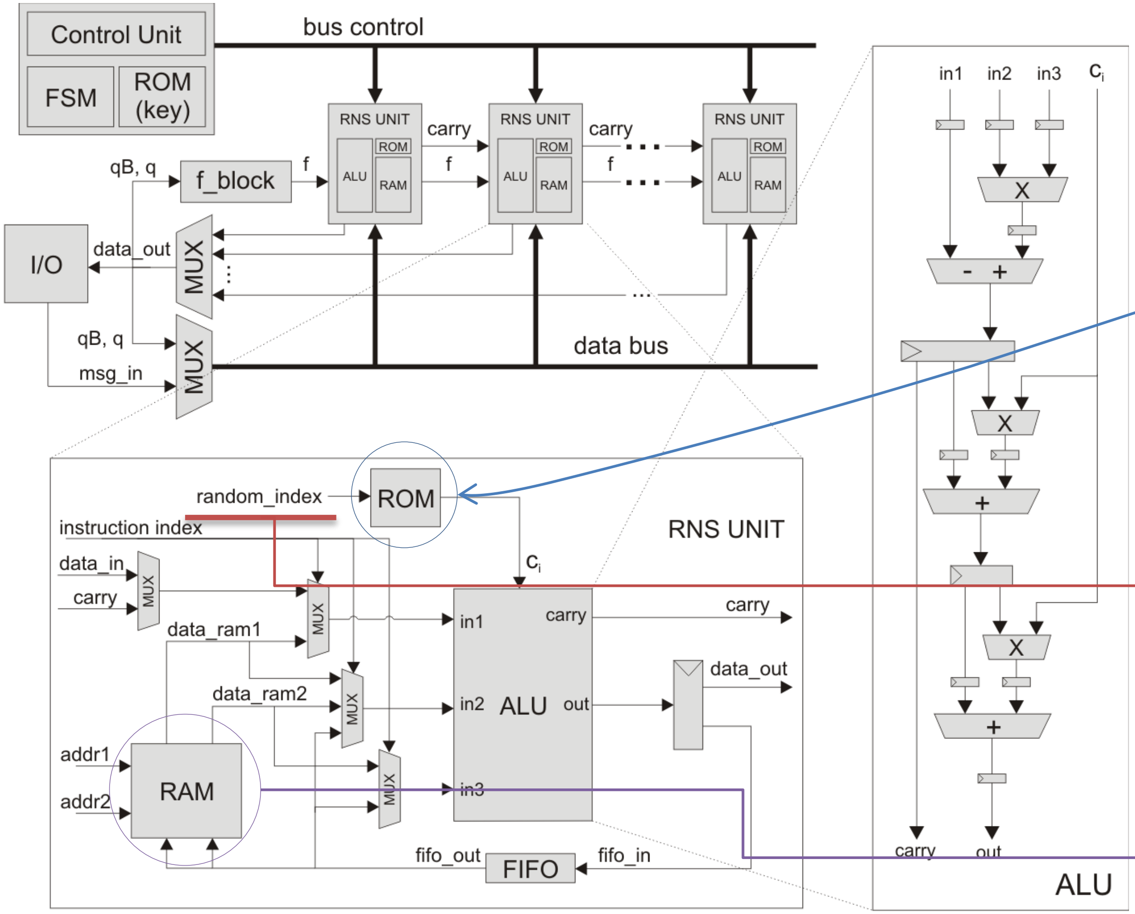
	RNS MM	RNS MM Improved
Pre-computations	$2k^2 + 7k$	$2k^2 + 5k$
RNS multiplications	$2k^2 + 7k$	$2k^2 + 5k$

* 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-Marsenne Number:

$$b_i = 2^w - c_i$$

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

(two times)

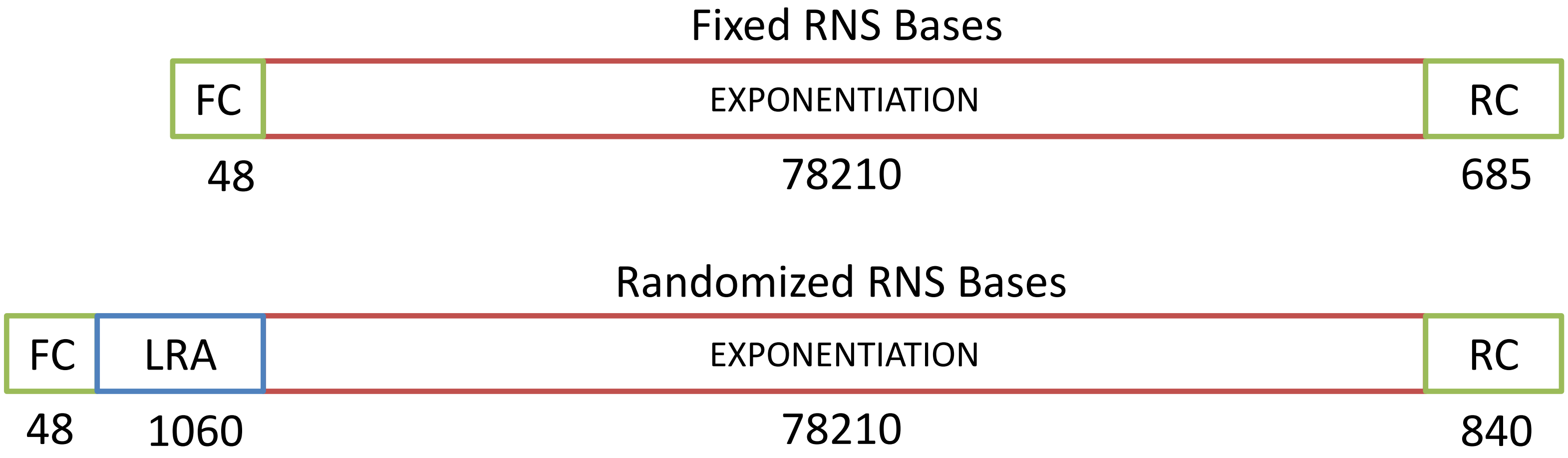
All RNS Units operate in all RNS moduli

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):



Fixed RNS Bases

FC | EXPONENTIATION | RC

48 | 78210 | 685

Randomized RNS Bases

FC | LRA | EXPONENTIATION | RC

48 | 1060 | 78210 | 840

FC = Radix to RNS
RC = RNS to Radix

Clock Cycles Overhead: 1%

Memory Overhead: 92%

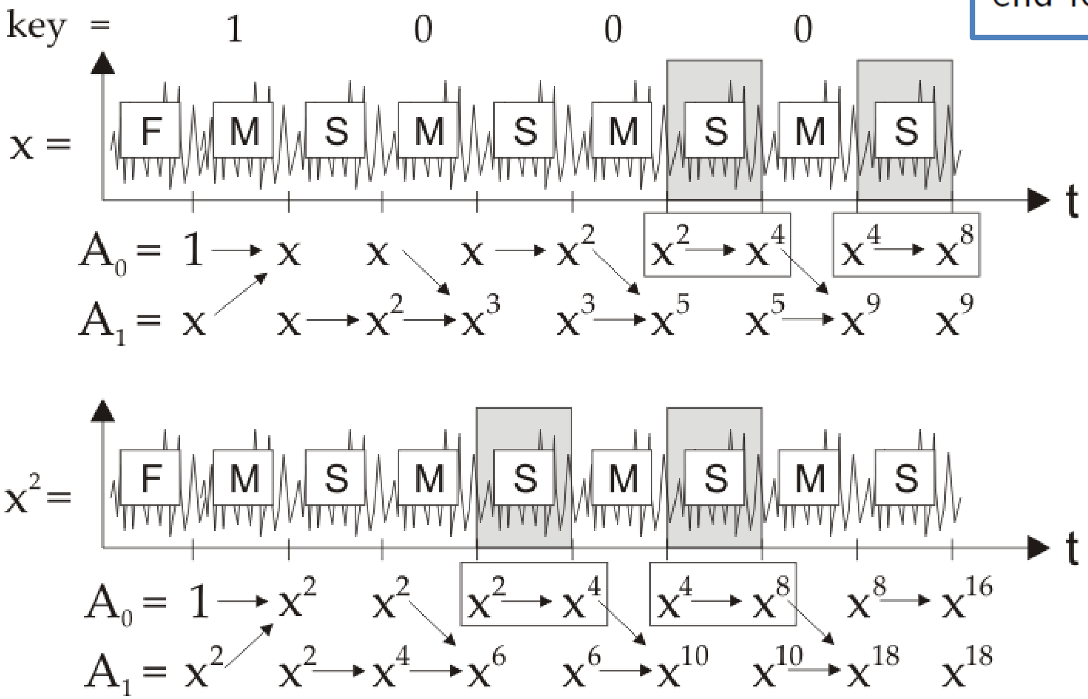
Agenda

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

Collision Attacks

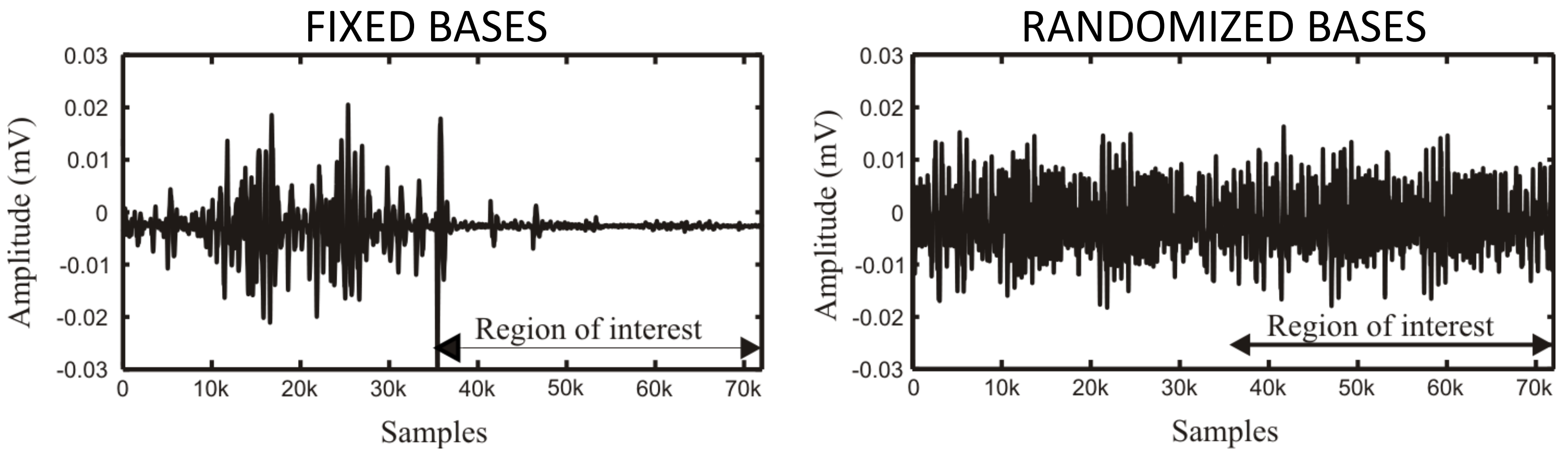
- Identify redundant operations by collecting two (averaged or not) traces for different chosen-message pairs:
  - (x,x^2): Doubling Attack
  - $(x,-x)$: Yen's et al Attack
  - (x^α,y^β): Homma's et al Attack

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



key = 1 0 0 0
x =
F M S M S M S M S
t
$A_0 = 1 \rightarrow x \quad x \quad x \rightarrow x^2 \quad x^2 \rightarrow x^4 \quad x^4 \rightarrow x^8$
$A_1 = x \quad x \rightarrow x^2 \rightarrow x^3 \quad x^3 \rightarrow x^5 \quad x^5 \rightarrow x^9 \quad x^9$
$x^2 =$
F M S M S M S M S
t
$A_0 = 1 \rightarrow x^2 \quad x^2 \quad x^2 \rightarrow x^4 \quad x^4 \rightarrow x^8 \quad x^8 \rightarrow x^{16}$
$A_1 = x^2 \quad x^2 \rightarrow x^4 \rightarrow x^6 \quad x^6 \rightarrow x^{10} \quad x^{10} \rightarrow x^{18} \quad x^{18}$

LRA vs Collision Attacks

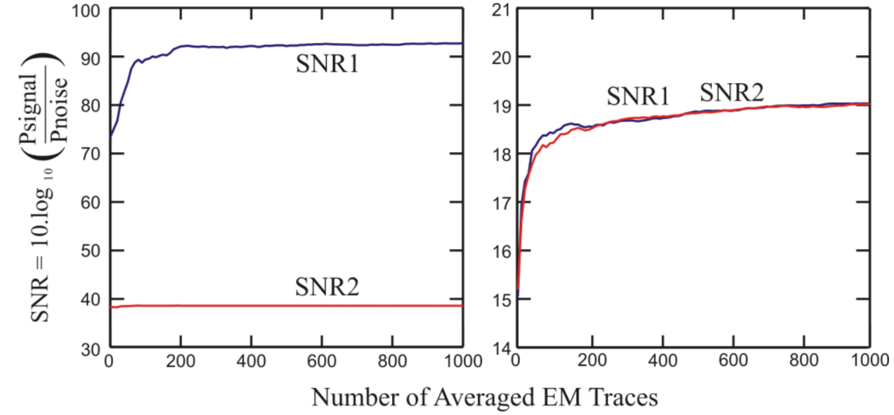


EM(T_S, x, e_i) = squaring EM trace at e_i

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

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

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



Correlation Attacks

HW_m = Hamming Weight of a Data *m*
$t_i(j)$ = sample *j* of a trace *i*

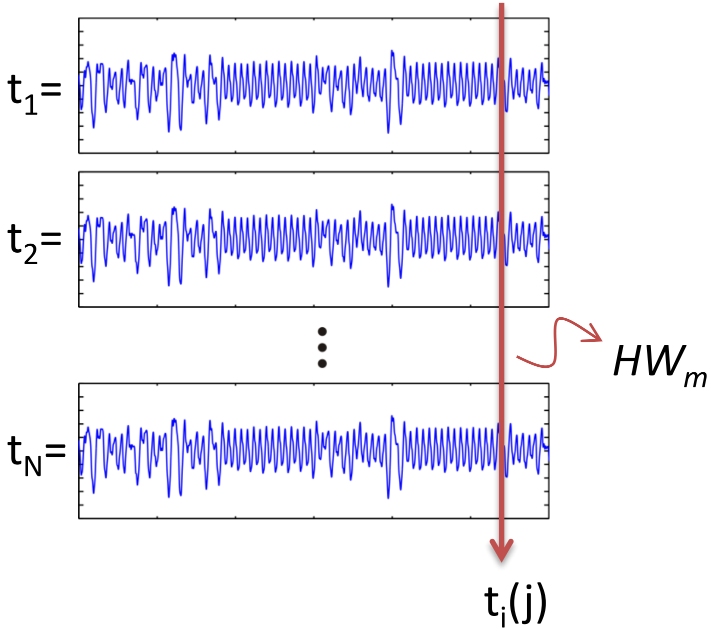
$$\rho(HW_m, t_i(j)) = \frac{cov(HW_m, t_i(j))}{\sqrt{var(HW_m)var(t_i(j))}}$$

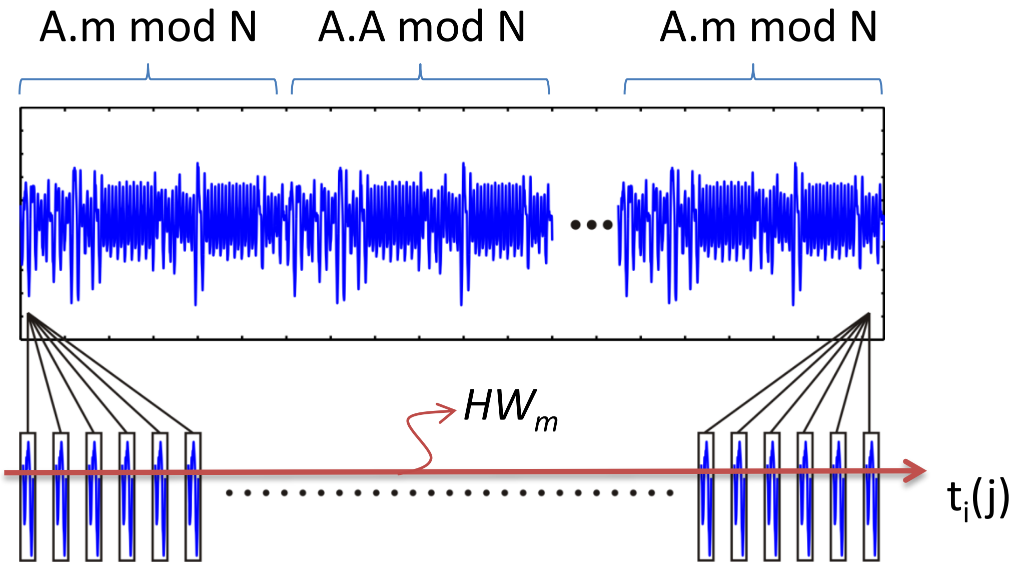
Vertical:

- Correlate HW x Trace

Horizontal (Immune to Exponent Blinding):

- Correlate HW x Trace
- Correlate Trace x Trace*



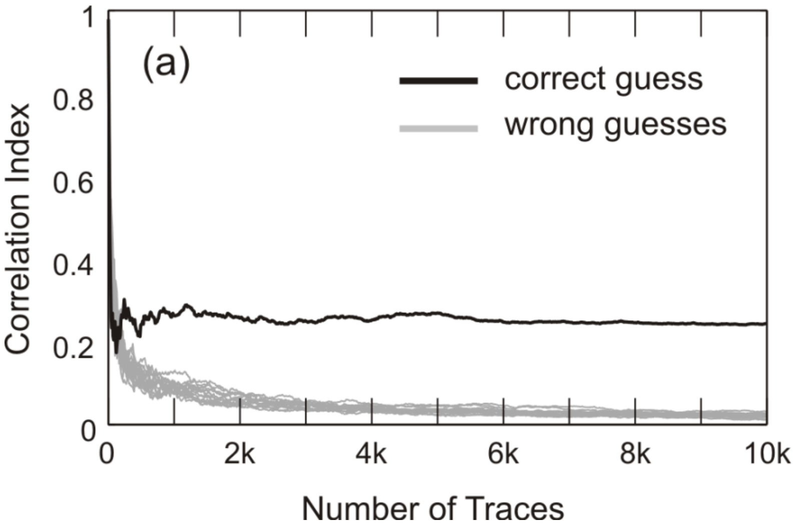


*C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Rousselet and V. Verneuil, “ROSETTA for Single Trace Analysis,” in *INDOCRYPT 2012;*

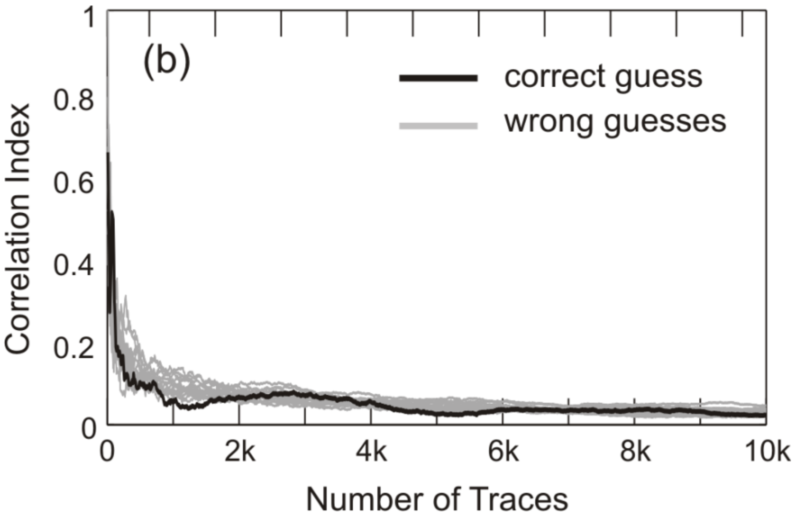
LRA vs Correlation Attacks

Vertical:

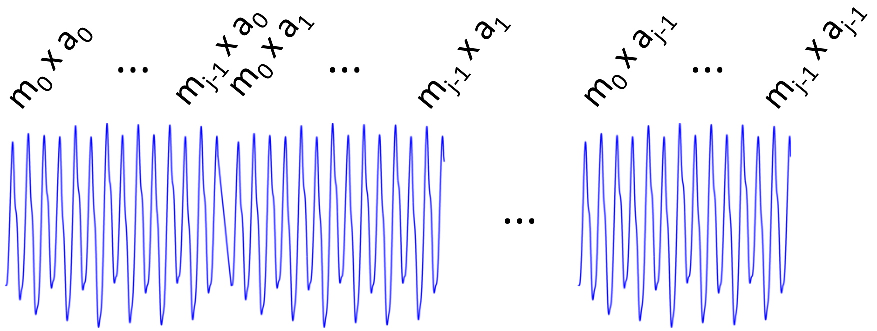
FIXED BASES



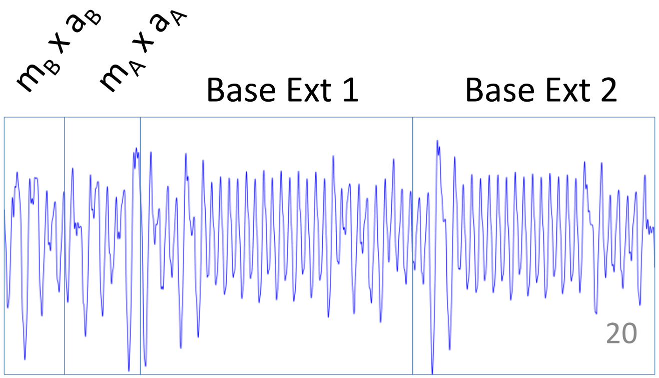
RANDOMIZED BASES



Horizontal: Proposed for Long-Integer Multiplications



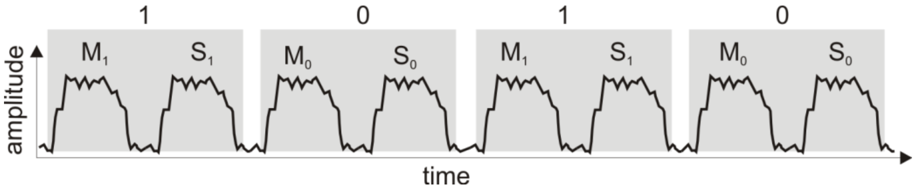
Horizontal: RNS Multiplications



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:



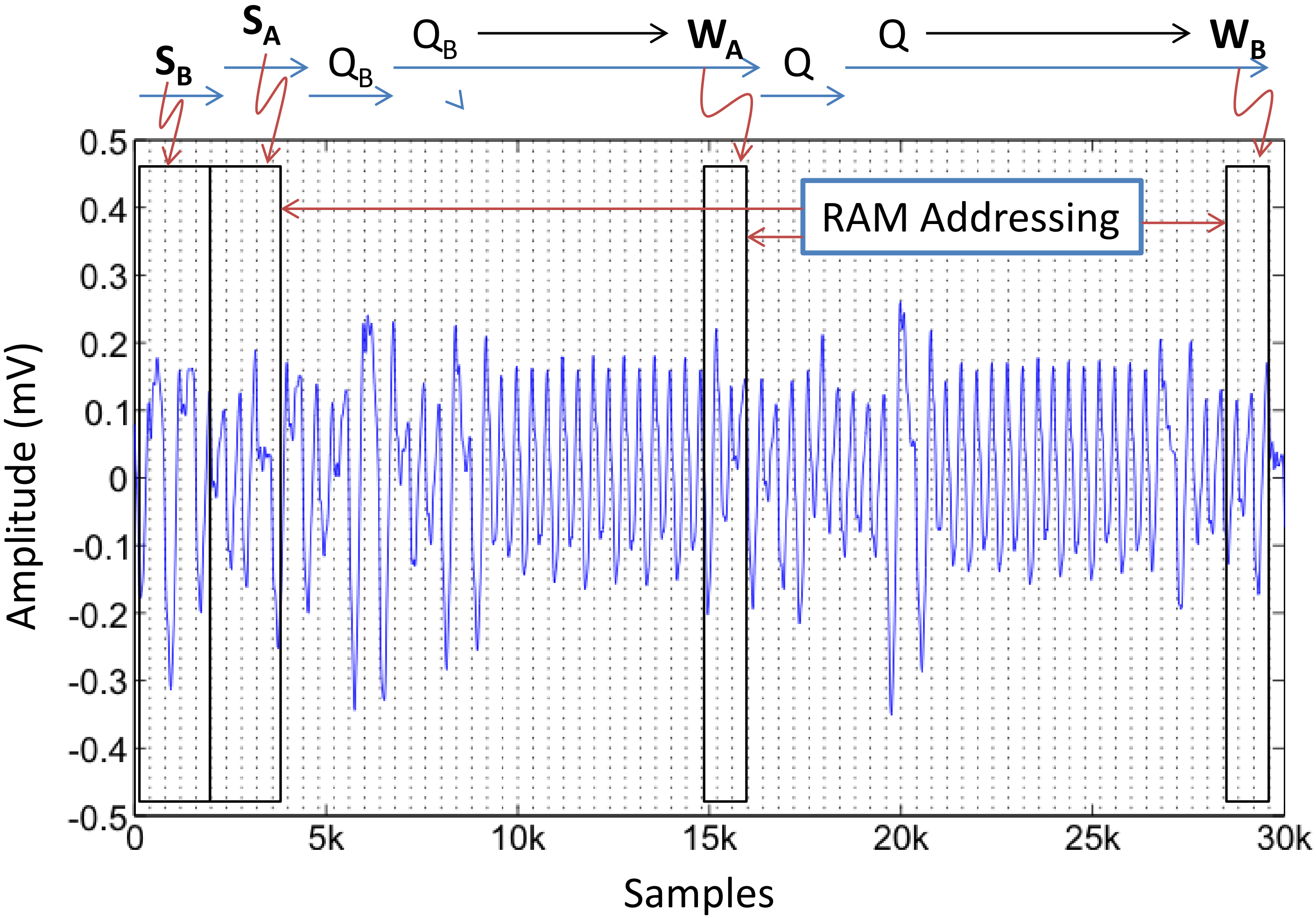
❑ Montgomery Ladder -> Find the means (μ) and std dev (σ) of two classes:

- $N(\mu_{(m0)},\sigma_{(m0)})$: 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**

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

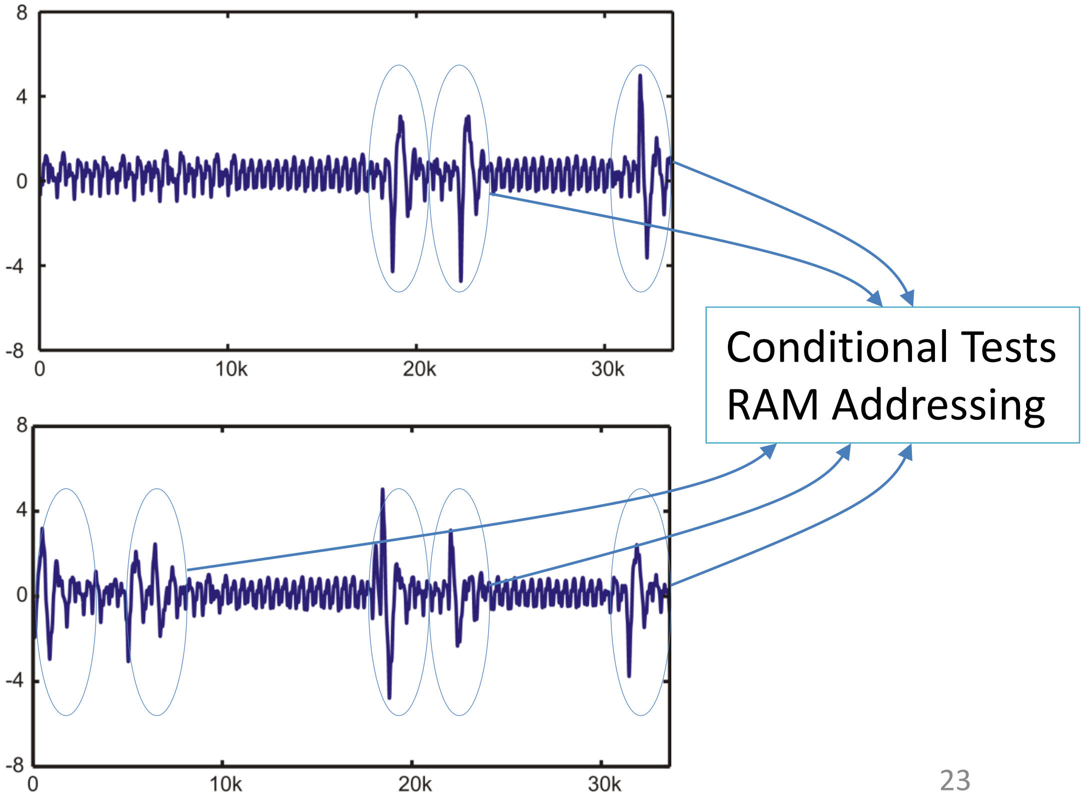
$$\overline{m_1} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} m_i(1)$$

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

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

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

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



Conditional Tests
RAM Addressing

RAM Addressing Randomization

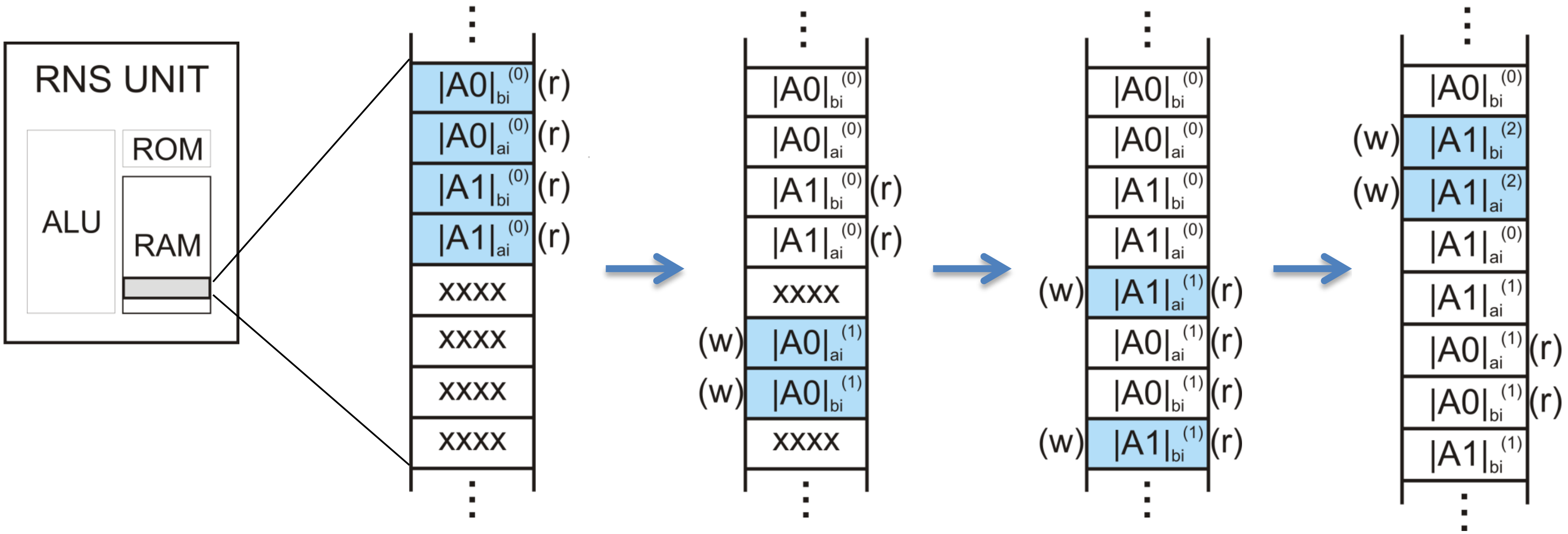
Intermediate results are never stored in same positions:

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

Multiplication(1):
A0 = A0.A1 mod N
read(A0, A1)
write(A0)

Squaring(1):
A1 = A1.A1 mod N
read(A1)
write(A1)

Multiplication(0):
A1 = A0.A1 mod N
read(A1)
write(A1)

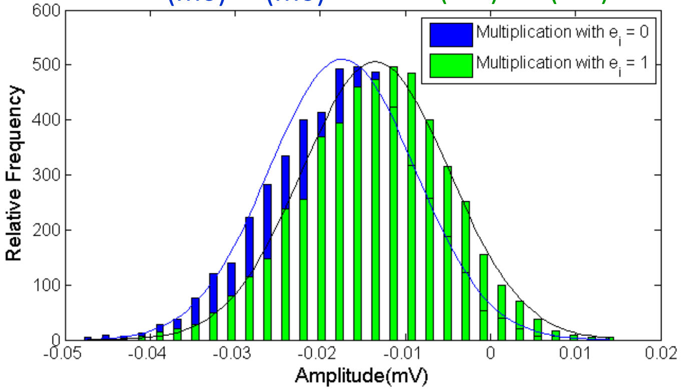


RAM Addressing Randomization

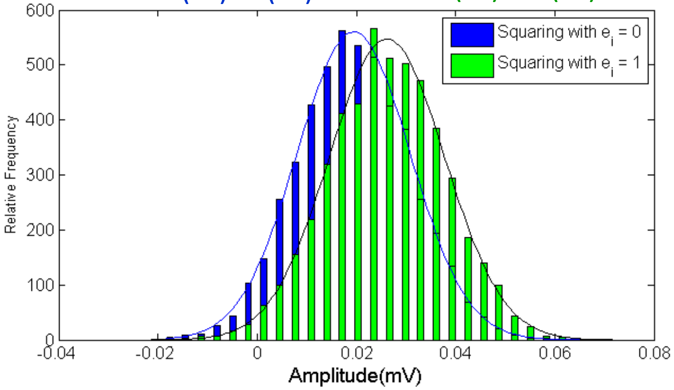
We took a fixed sample point t_i representing the RAM addressing (writing):

Unprotected:

$N(\mu_{(m0)},\sigma_{(m0)})$ $N(\mu_{(m1)},\sigma_{(m1)})$

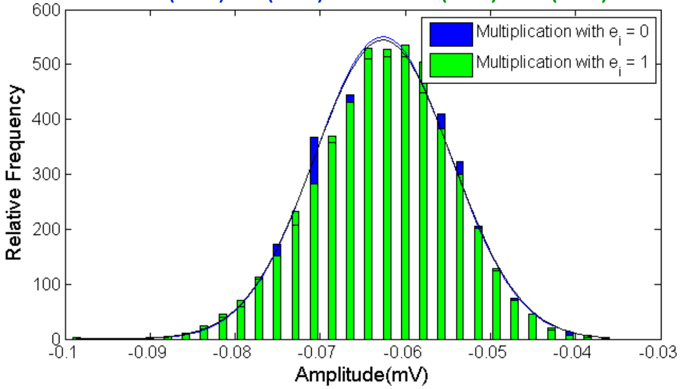


$N(\mu_{(s0)},\sigma_{(s0)})$ $N(\mu_{(s1)},\sigma_{(s1)})$

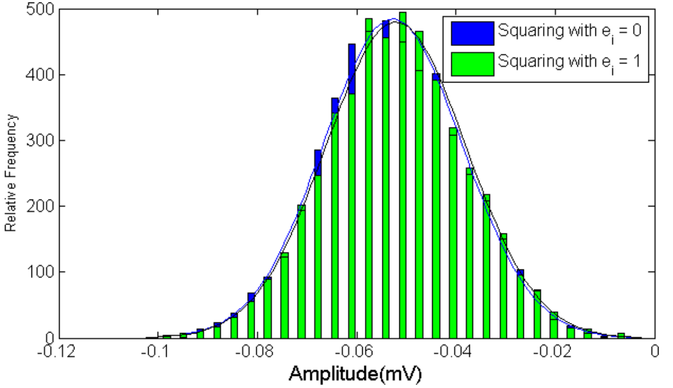


Protected:

$N(\mu_{(m0)},\sigma_{(m0)})$ $N(\mu_{(m1)},\sigma_{(m1)})$



$N(\mu_{(s0)},\sigma_{(s0)})$ $N(\mu_{(s1)},\sigma_{(s1)})$



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