ROSETTA for Single Trace Analysis

Recovery Of Secret Exponent by Triangular Trace Analysis

Christophe Clavier[1] Benoit Feix[1,2] Georges Gagnerot[1,2]
Christophe Giraud [3] Mylène Roussellet[2] Vincent Verneuil[2]

[1]Université de Limoges
[2]Inside Secure
[3]Oberthur Technologies

Kolkata – 10 December 2012

1. Introduction
2. Context of the attack
3. Description of Rosetta
4. Countermeasures
5. Conclusion

Introduction

Several Side-Channel Analysis (SCA) apply to RSA modular exponentiation on embedded devices:

- Simple Side-Channel Analysis : information about the private exponent is directly extracted from one side-channel trace
- Differential Side-Channel Analysis : exploits a statistical treatment of many traces

Introduction

Several Side-Channel Analysis (SCA) apply to RSA modular exponentiation on embedded devices:

- Simple Side-Channel Analysis : information about the private exponent is directly extracted from one side-channel trace
- Differential Side-Channel Analysis : exploits a statistical treatment of many traces

To protect against SCA:

- The exponentiation operands are blinded by randomization
- The choice of the exponentiation method is important

Blinding the exponentiation

Blinding countermeasure

Blinding $s = m^d \bmod n$ makes use of two random λ-bit integers r_d and r_m:
($\lambda \geq 32$ bits)

exponent $d^\star \leftarrow d + r_d \cdot \varphi(n)$

message $m^\star \leftarrow m + r_m \cdot n$

Blinding the exponentiation

Blinding countermeasure

Blinding $s = m^d \bmod n$ makes use of two random λ-bit integers r_d and r_m:
($\lambda \geq 32$ bits)

exponent $d^\star \leftarrow d + r_d \cdot \varphi(n)$

message $m^\star \leftarrow m + r_m \cdot n$

The exponentiation is computed modulo $2^\lambda n$:

$$s = (m + r_m \cdot n)^{d + r_d \cdot \varphi(n)} \bmod 2^\lambda n$$

followed by a final reduction modulo n.

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiation algorithm

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiation algorithm

- Montgomerry ladder: $1M + 1S$ (per exponent bit)

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiation algorithm

- Montgomerry ladder: $1M + 1S$ (per exponent bit)
- Joye ladder: $1M + 1S$

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiation algorithm

- Montgomerry ladder: $1M + 1S$ (per exponent bit)
- Joye ladder: $1M + 1S$
- Square always: $2S$

Regular exponentiations

Many algorithms like basic square-and-multiply present an irregular sequence of successive squarings and multiplications by the message.

Identifying this sequence reveals the private exponent bits.

One should use an exponentiation which presents a regular pattern of S and M:

Regular exponentiation algorithm

- Montgomerry ladder: $1M + 1S$ (per exponent bit)
- Joye ladder: $1M + 1S$
- Square always: $2S$
- Atomic multiply-always: $1.5M$

Atomic multiply-always

The fastest regular exponentiation is:

Algorithm 1 Atomic multiply-always exponentiation

Input: $x, n \in \mathbb{N}$, $d = (d_{v-1}d_{v-2}\ldots d_0)_2$
Output: $x^d \bmod n$

1: $R_0 \leftarrow 1$
2: $R_1 \leftarrow x$
3: $i \leftarrow v - 1$
4: $k \leftarrow 0$
5: **while** $i \geq 0$ **do**
6: $\quad R_0 \leftarrow R_0 \times R_k \bmod n$
7: $\quad k \leftarrow k \oplus d_i$ [$\oplus$ stands for bitwise X-or]
8: $\quad i \leftarrow i - \neg k$ [$\neg$ stands for bitwise negation]
9: **return** R_0

Atomic multiply-always

The fastest regular exponentiation is:

Algorithm 1 Atomic multiply-always exponentiation

Input: $x, n \in \mathbb{N}$, $d = (d_{v-1}d_{v-2}\ldots d_0)_2$
Output: $x^d \bmod n$
1: $R_0 \leftarrow 1$
2: $R_1 \leftarrow x$
3: $i \leftarrow v - 1$
4: $k \leftarrow 0$
5: **while** $i \geq 0$ **do**
6: $\quad R_0 \leftarrow R_0 \times R_k \bmod n$
7: $\quad k \leftarrow k \oplus d_i$ [$\oplus$ stands for bitwise X-or]
8: $\quad i \leftarrow i - \neg k$ [$\neg$ stands for bitwise negation]
9: **return** R_0

Considered secure exponentiation

We focus on the atomic multiply-always exponentiation protected by exponent and message blindings.

Inside the long integer multiplication (LIM)

Inside the long integer multiplication (LIM)

- Each $\text{LIM}(x, y) = x \cdot y$ is computed with a small t-bit multiplier

Inside the long integer multiplication (LIM)

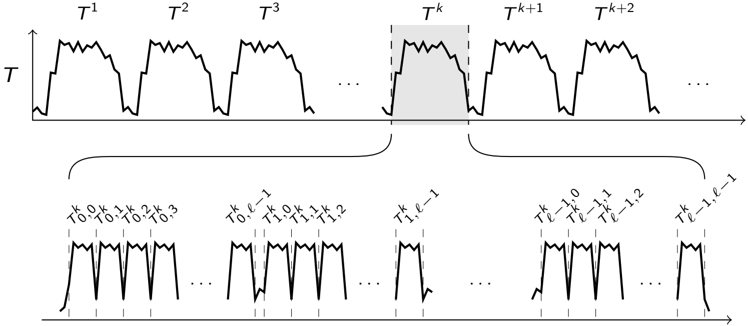
- Each $\mathrm{LIM}(x, y) = x \cdot y$ is computed with a small t-bit multiplier
- Schoolbook method on ℓ-word integers expressed in base $b = 2^t$
  $x = (x_{\ell-1}x_{\ell-2} \ldots x_1 x_0)_b \qquad y = (y_{\ell-1}y_{\ell-2} \ldots y_1 y_0)_b$

Inside the long integer multiplication (LIM)

- Each $\mathrm{LIM}(x, y) = x \cdot y$ is computed with a small t-bit multiplier
- Schoolbook method on ℓ-word integers expressed in base $b = 2^t$
  $x = (x_{\ell-1}x_{\ell-2}\ldots x_1x_0)_b \qquad y = (y_{\ell-1}y_{\ell-2}\ldots y_1y_0)_b$
- Each k-th LIM side-channel trace T^k can be split into ℓ^2 trace segments $T^k_{i,j}$ for each single-precision operation $x_i \cdot y_j$

Inside the long integer multiplication (LIM)

- Each $\mathrm{LIM}(x, y) = x \cdot y$ is computed with a small t-bit multiplier
- Schoolbook method on ℓ-word integers expressed in base $b = 2^t$
  $x = (x_{\ell-1}x_{\ell-2}\ldots x_1x_0)_b \qquad y = (y_{\ell-1}y_{\ell-2}\ldots y_1y_0)_b$
- Each k-th LIM side-channel trace T^k can be split into ℓ^2 trace segments $T^k_{i,j}$ for each single-precision operation $x_i \cdot y_j$



Two possible threats : *SAC 2008* attack

Two possible threats : *SAC 2008* attack

Is the multiply-always algorithm a true regular exponentiation?

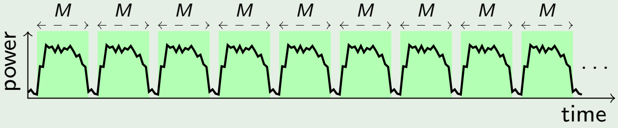
Two possible threats : *SAC 2008* attack

Is the multiply-always algorithm a true regular exponentiation?
Not really. . .

Two possible threats : *SAC 2008* attack

Is the multiply-always algorithm a true regular exponentiation?
Not really. . .

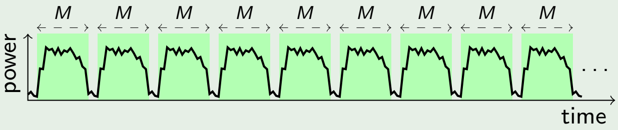
Regular at instruction level



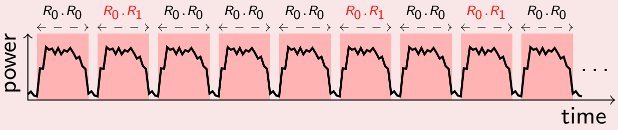
Two possible threats : *SAC 2008* attack

Is the multiply-always algorithm a true regular exponentiation?
Not really. . .

Regular at instruction level



Not regular at data level



Two possible threats : *SAC 2008* attack

Amiel et al. [AFT+08] noticed that the average Hamming weight of the two following distributions differ:

Two different distributions

square	$\mathrm{HW}(x \cdot x)$	(uniformly distributed x)
multiplication	$\mathrm{HW}(x \cdot y)$	(independant uniformly distributed x and y)

Two possible threats : *SAC 2008* attack

Amiel et al. [AFT+08] noticed that the average Hamming weight of the two following distributions differ:

Two different distributions

square	$HW(x \cdot x)$	(uniformly distributed x)
multiplication	$HW(x \cdot y)$	(independant uniformly distributed x and y)

- By averaging many exponentiation traces it is possible to distinguish between a $R_0 \times R_0$ LIM and a $R_0 \times R_1$ one

Two possible threats : *SAC 2008* attack

Amiel et al. [AFT+08] noticed that the average Hamming weight of the two following distributions differ:

Two different distributions

square	$\mathrm{HW}(x \cdot x)$	(uniformly distributed x)
multiplication	$\mathrm{HW}(x \cdot y)$	(independant uniformly distributed x and y)

- By averaging many exponentiation traces it is possible to distinguish between a $R_0 \times R_0$ LIM and a $R_0 \times R_1$ one
- Many traces $\rightarrow$ the attack is prevented by exponent blinding

Two possible threats : *SAC 2008* attack

Amiel et al. [AFT+08] noticed that the average Hamming weight of the two following distributions differ:

Two different distributions

square	$\mathrm{HW}(x \cdot x)$	(uniformly distributed x)
multiplication	$\mathrm{HW}(x \cdot y)$	(independant uniformly distributed x and y)

- By averaging many exponentiation traces it is possible to distinguish between a $R_0 \times R_0$ LIM and a $R_0 \times R_1$ one
- Many traces $\rightarrow$ the attack is prevented by exponent blinding
- Authors suggested to apply this distinguisher horizontally on the set of trace segments $\{T_{i,i}^{k}\}_{0 \leq i < \ell}$ (they did not experiment this idea)

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
  - first LIM : $1 \times m$

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
    - first LIM : $1 \times m$
    - second LIM : $m \times m$

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
  - first LIM : $1 \times m$
  - second LIM : $m \times m$
- Based on these two LIM, build a template for the set of average leakages of single-precision multiplications by m_j

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
  - first LIM : $1 \times m$
  - second LIM : $m \times m$
- Based on these two LIM, build a template for the set of average leakages of single-precision multiplications by m_j
- For each k-th LIM, compute the corresponding set of average leakages and decide:

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
  - first LIM : $1 \times m$
  - second LIM : $m \times m$
- Based on these two LIM, build a template for the set of average leakages of single-precision multiplications by m_j
- For each k-th LIM, compute the corresponding set of average leakages and decide:
  - the LIM is a multiplication by m if this set of leakages is close to the template (Euclidean distance)

Two possible threats : *Big Mac* attack

Walter [Wal01] proposed a single trace attack able to distinguish squarings from multiplications:

- First and second LIM of the exponentiation both imply the message as right operand :
  - first LIM : $1 \times m$
  - second LIM : $m \times m$
- Based on these two LIM, build a template for the set of average leakages of single-precision multiplications by m_j
- For each k-th LIM, compute the corresponding set of average leakages and decide:
  - the LIM is a multiplication by m if this set of leakages is close to the template (Euclidean distance)
  - the LIM is a square if it is not

Principle of the attack

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Example of all single-precision operations with $\ell = 4$:

$$M = \begin{pmatrix} x_0 y_0 & x_0 y_1 & x_0 y_2 & x_0 y_3 \\ x_1 y_0 & x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_0 & x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_0 & x_3 y_1 & x_3 y_2 & x_3 y_3 \end{pmatrix}$$

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Example of all single-precision operations with $\ell = 4$:

$$M = \begin{pmatrix} x_0y_0 & x_0y_1 & x_0y_2 & x_0y_3 \\ x_1y_0 & x_1y_1 & x_1y_2 & x_1y_3 \\ x_2y_0 & x_2y_1 & x_2y_2 & x_2y_3 \\ x_3y_0 & x_3y_1 & x_3y_2 & x_3y_3 \end{pmatrix}$$

If the LIM is a squaring then $x = y$:

$$S = \begin{pmatrix} x_0x_0 & x_0x_1 & x_0x_2 & x_0x_3 \\ x_1x_0 & x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_0 & x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_0 & x_3x_1 & x_3x_2 & x_3x_3 \end{pmatrix}$$

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Example of all single-precision operations with $\ell = 4$:

$$M = \begin{pmatrix} x_0 y_0 & x_0 y_1 & x_0 y_2 & x_0 y_3 \\ x_1 y_0 & x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_0 & x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_0 & x_3 y_1 & x_3 y_2 & x_3 y_3 \end{pmatrix}$$

If the LIM is a squaring then $x = y$:

$$S = \begin{pmatrix} x_0 x_0 & {\color{red} x_0 x_1} & x_0 x_2 & x_0 x_3 \\ {\color{red} x_1 x_0} & x_1 x_1 & x_1 x_2 & x_1 x_3 \\ x_2 x_0 & x_2 x_1 & x_2 x_2 & x_2 x_3 \\ x_3 x_0 & x_3 x_1 & x_3 x_2 & x_3 x_3 \end{pmatrix}$$

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Example of all single-precision operations with $\ell = 4$:

$$M = \begin{pmatrix} x_0 y_0 & x_0 y_1 & x_0 y_2 & x_0 y_3 \\ x_1 y_0 & x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_0 & x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_0 & x_3 y_1 & x_3 y_2 & x_3 y_3 \end{pmatrix}$$

If the LIM is a squaring then $x = y$:

$$S = \begin{pmatrix} x_0 x_0 & x_0 x_1 & x_0 x_2 & x_0 x_3 \\ x_1 x_0 & x_1 x_1 & x_1 x_2 & x_1 x_3 \\ x_2 x_0 & x_2 x_1 & x_2 x_2 & x_2 x_3 \\ x_3 x_0 & x_3 x_1 & x_3 x_2 & x_3 x_3 \end{pmatrix}$$

Principle of the attack

LIM(x, y) in base $b = 2^t$ by classical schoolbook method:

$$x \times y = \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x_i y_j b^{i+j}$$

Example of all single-precision operations with $\ell = 4$:

$$M = \begin{pmatrix} x_0 y_0 & x_0 y_1 & x_0 y_2 & x_0 y_3 \\ x_1 y_0 & x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_0 & x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_0 & x_3 y_1 & x_3 y_2 & x_3 y_3 \end{pmatrix}$$

If the LIM is a squaring then $x = y$:

$$S = \begin{pmatrix} \color{blue}{x_0 x_0} & \color{red}{x_0 x_1} & \color{red}{x_0 x_2} & \color{red}{x_0 x_3} \\ \color{red}{x_1 x_0} & \color{blue}{x_1 x_1} & \color{red}{x_1 x_2} & \color{red}{x_1 x_3} \\ \color{red}{x_2 x_0} & \color{red}{x_2 x_1} & \color{blue}{x_2 x_2} & \color{red}{x_2 x_3} \\ \color{red}{x_3 x_0} & \color{red}{x_3 x_1} & \color{red}{x_3 x_2} & \color{blue}{x_3 x_3} \end{pmatrix}$$

Principle of the attack

Principle of the attack

On the diagonal

Principle of the attack

On the diagonal

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i$

Principle of the attack

On the diagonal

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i$

multiplication LIM(x, y) with $x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i$

Principle of the attack

On the diagonal

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i$

multiplication LIM(x, y) with $x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Principle of the attack

On the diagonal

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_i$ is a squaring$) = 1 \quad \forall i$

multiplication LIM(x, y) with $x \neq y \Rightarrow \text{Prob}(x_i \times y_i$ is a squaring$) \approx 0 \quad \forall i$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much $(\ell = \frac{|n|}{t}$; typical values: 32, 64)

Principle of the attack

On the diagonal

square	$\mathrm{LIM}(x, y)$ with $x = y \Rightarrow \mathrm{Prob}(x_i \times y_i$ is a squaring$) = 1 \quad \forall i$
multiplication	$\mathrm{LIM}(x, y)$ with $x \neq y \Rightarrow \mathrm{Prob}(x_i \times y_i$ is a squaring$) \approx 0 \quad \forall i$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much $(\ell = \frac{|n|}{t}$; typical values: 32, 64)

On the two symmetric triangles (*Rosetta*

)

Principle of the attack

On the diagonal

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i$

multiplication LIM(x, y) with $x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much $\left(\ell = \frac{|n|}{t}\right.$; typical values: 32, 64)

On the two symmetric triangles (*Rosetta*)

square LIM(x, y) with $x = y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) = 1 \quad \forall i \neq j.$

Principle of the attack

On the diagonal

$$\text{square} \quad \text{LIM}(x,y) \text{ with } x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i$$
$$\text{multiplication} \quad \text{LIM}(x,y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i$$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much $(\ell = \frac{|n|}{t}$; typical values: 32, 64)

On the two symmetric triangles (*Rosetta*)

$$\text{square} \quad \text{LIM}(x,y) \text{ with } x = y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) = 1 \quad \forall i \neq j.$$
$$\text{multiplication} \quad \text{LIM}(x,y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \quad \forall i \neq j.$$

Principle of the attack

On the diagonal

$$\begin{aligned} \text{square} \quad & \mathrm{LIM}(x,y) \text{ with } x = y \Rightarrow \mathrm{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i \\ \text{multiplication} \quad & \mathrm{LIM}(x,y) \text{ with } x \neq y \Rightarrow \mathrm{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i \end{aligned}$$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much $\left(\ell = \frac{|n|}{t}\right.$; typical values: 32, 64)

On the two symmetric triangles (*Rosetta*

)

$$\begin{aligned} \text{square} \quad & \mathrm{LIM}(x,y) \text{ with } x = y \Rightarrow \mathrm{Prob}(x_i \times y_j = x_j \times y_i) = 1 \quad \forall i \neq j. \\ \text{multiplication} \quad & \mathrm{LIM}(x,y) \text{ with } x \neq y \Rightarrow \mathrm{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \quad \forall i \neq j. \end{aligned}$$

We expect to detect the conditional triangular collision

Principle of the attack

On the diagonal

$$\begin{array}{rl} \text{square} & \text{LIM}(x,y) \text{ with } x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \quad \forall i \\ \text{multiplication} & \text{LIM}(x,y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \quad \forall i \end{array}$$

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations $x_i \times y_i$ of a LIM on a single trace

Drawback: only ℓ trace segments is not so much ($\ell = \frac{|n|}{t}$; typical values: 32, 64)

On the two symmetric triangles (*Rosetta*
)

$$\begin{array}{rl} \text{square} & \text{LIM}(x,y) \text{ with } x = y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) = 1 \quad \forall i \neq j. \\ \text{multiplication} & \text{LIM}(x,y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \quad \forall i \neq j. \end{array}$$

We expect to detect the conditional triangular collision

Advantage: as much as $(\ell^2 - \ell)/2$ trace segments

Two distinguishers

Two distinguishers

Mean Euclidean distance

Between pairs of trace segments $T_{i,j}$ and $T_{j,i}$:

Two distinguishers

Mean Euclidean distance

Between pairs of trace segments $T_{i,j}$ and $T_{j,i}$:

$$d = \sqrt{\frac{2}{\ell^2 - \ell} \sum_{0 \leq i < j < \ell} (T_{i,j} - T_{j,i})^2}$$

Two distinguishers

Mean Euclidean distance

Between pairs of trace segments $T_{i,j}$ and $T_{j,i}$:

$$d = \sqrt{\frac{2}{\ell^2 - \ell} \sum_{0 \leq i < j < \ell} (T_{i,j} - T_{j,i})^2}$$

Collision-Correlation

Between two series of trace segments (with same (i,j) ordering):

$\Theta_0 = \{ T_{i,j} \text{ s.t. } 0 \leq i < j \leq \ell - 1 \}$ (upper right triangle)

$\Theta_1 = \{ T_{j,i} \text{ s.t. } 0 \leq i < j \leq \ell - 1 \}$ (lower left triangle)

Two distinguishers

Mean Euclidean distance

Between pairs of trace segments $T_{i,j}$ and $T_{j,i}$:

$$d = \sqrt{\frac{2}{\ell^2 - \ell} \sum_{0 \leq i < j < \ell} (T_{i,j} - T_{j,i})^2}$$

Collision-Correlation

Between two series of trace segments (with same (i, j) ordering):

$\Theta_0 = \{T_{i,j} \text{ s.t. } 0 \leq i < j \leq \ell - 1\}$ (upper right triangle)

$\Theta_1 = \{T_{j,i} \text{ s.t. } 0 \leq i < j \leq \ell - 1\}$ (lower left triangle)

$$\hat{\rho}_{\Theta_0,\Theta_1}(t) = \frac{\mathrm{Cov}(\Theta_0(t), \Theta_1(t))}{\sigma_{\Theta_0(t)}\sigma_{\Theta_1(t)}}$$

Simulation results

Simulation results

Generation of simulated side-channel traces of LIM:

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\mathrm{HW}(x_i)$, $\mathrm{HW}(y_j)$, $\mathrm{HW}((x_i \times y_j) \operatorname{div} b)$, $\mathrm{HW}((x_i \times y_j) \bmod b)$

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\mathrm{HW}(x_i)$, $\mathrm{HW}(y_j)$, $\mathrm{HW}((x_i \times y_j) \operatorname{div} b)$, $\mathrm{HW}((x_i \times y_j) \bmod b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\mathrm{HW}(x_i)$, $\mathrm{HW}(y_j)$, $\mathrm{HW}((x_i \times y_j) \operatorname{div} b)$, $\mathrm{HW}((x_i \times y_j) \bmod b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1 000 LIM experiments for each (attack,noise level)

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\mathrm{HW}(x_i)$, $\mathrm{HW}(y_j)$, $\mathrm{HW}((x_i \times y_j) \operatorname{div} b)$, $\mathrm{HW}((x_i \times y_j) \bmod b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1 000 LIM experiments for each (attack,noise level)

Comparison of five attacks:

- single trace variant of *SAC 2008* technique
- original *Big Mac* (Euclidean distance)

Simulation results

Generation of simulated side-channel traces of LIM:

- 32×32-bit multiplier $(b = 2^{32})$
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\mathrm{HW}(x_i)$, $\mathrm{HW}(y_j)$, $\mathrm{HW}((x_i \times y_j) \operatorname{div} b)$, $\mathrm{HW}((x_i \times y_j) \bmod b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1 000 LIM experiments for each (attack,noise level)

Comparison of five attacks:

- single trace variant of *SAC 2008* technique
- original *Big Mac* (Euclidean distance)
- *Big Mac CoCo* (variant with Collision-Correlation)
- *Rosetta ED* (Euclidean distance)
- *Rosetta CoCo* (Collision-Correlation)

Success rate with a null noise ($\sigma = 0$)

Success rate with a null noise ($\sigma = 0$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.986	0.990	0.993	0.994	0.995
SAC 2008	0.533	0.618	0.734	0.858	0.897
Big Mac CoCo	0.999	1.00	1.00	1.00	1.00
Rosetta ED	1.00	1.00	1.00	1.00	1.00
Rosetta CoCo	1.00	1.00	1.00	1.00	1.00

Table: Success rate with a null noise, $\sigma = 0$

Success rate with a null noise ($\sigma = 0$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.986	0.990	0.993	0.994	0.995
SAC 2008	0.533	0.618	0.734	0.858	0.897
Big Mac CoCo	0.999	1.00	1.00	1.00	1.00
Rosetta ED	1.00	1.00	1.00	1.00	1.00
Rosetta CoCo	1.00	1.00	1.00	1.00	1.00

Table: Success rate with a null noise, $\sigma = 0$

- All techniques give excellent results except *SAC 2008*
  (The number of trace segments is too small, except for large moduli)

Success rate with a moderate noise ($\sigma = 2$)

Success rate with a moderate noise ($\sigma = 2$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.767	0.775	0.807	0.816	0.818
SAC 2008	0.546	0.629	0.717	0.805	0.855
Big Mac CoCo	0.981	0.998	0.999	1.00	1.00
Rosetta ED	1.00	1.00	1.00	1.00	1.00
Rosetta CoCo	1.00	1.00	1.00	1.00	1.00

Table: Success rate with a moderate noise, $\sigma = 2$

Success rate with a moderate noise ($\sigma = 2$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.767	0.775	0.807	0.816	0.818
SAC 2008	0.546	0.629	0.717	0.805	0.855
Big Mac CoCo	0.981	0.998	0.999	1.00	1.00
Rosetta ED	1.00	1.00	1.00	1.00	1.00
Rosetta CoCo	1.00	1.00	1.00	1.00	1.00

Table: Success rate with a moderate noise, $\sigma = 2$

- As for *SAC 2008*, original *Big Mac* does not give good results
- The three new techniques are quite efficient

Success rate with a strong noise ($\sigma = 7$)

Success rate with a strong noise ($\sigma = 7$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.557	0.577	0.621	0.614	0.632
SAC 2008	0.551	0.577	0.623	0.662	0.702
Big Mac CoCo	0.737	0.855	0.909	0.963	0.981
Rosetta ED	0.711	0.821	0.878	0.953	0.992
Rosetta CoCo	0.685	0.816	0.906	0.992	0.997

Table: Success rate with a strong noise, $\sigma = 7$

Success rate with a strong noise ($\sigma = 7$)

Technique	512 bits	768 bits	1024 bits	1536 bits	2048 bits
Big Mac	0.557	0.577	0.621	0.614	0.632
SAC 2008	0.551	0.577	0.623	0.662	0.702
Big Mac CoCo	0.737	0.855	0.909	0.963	0.981
Rosetta ED	0.711	0.821	0.878	0.953	0.992
Rosetta CoCo	0.685	0.816	0.906	0.992	0.997

Table: Success rate with a strong noise, $\sigma = 7$

- All three new attacks still give good success rates at strong noise level
- *Big Mac CoCo* (up to 1024 bits) and *Rosetta Coco* (above 1024 bits) are the most efficient ones

1. Introduction
2. Context of the attack
3. Description of Rosetta
4. Countermeasures
5. Conclusion

Important difference between *Big Mac* and *Rosetta*

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

They all apply to a fully blinded atomic multiply-always exponentiation, but. . .

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

They all apply to a fully blinded atomic multiply-always exponentiation, but...

- when attacking a LIM *Big Mac* refers to the leakage of previous ones (templates)

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

They all apply to a fully blinded atomic multiply-always exponentiation, but...

- when attacking a LIM *Big Mac* refers to the leakage of previous ones (templates)
- refreshing the message blinding at each LIM would thwart *Big Mac* (e.g. $m^\star \leftarrow m^\star + n$)

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

They all apply to a fully blinded atomic multiply-always exponentiation, but...

- when attacking a LIM *Big Mac* refers to the leakage of previous ones (templates)
- refreshing the message blinding at each LIM would thwart *Big Mac* (e.g. $m^\star \leftarrow m^\star + n$)
- *Rosetta* is strictly local: only the leakage of the current LIM is exploited

Important difference between *Big Mac* and *Rosetta*

The 5 techniques we considered:

- make use of a single side-channel trace (bypass exponent blinding)
- do not require the knowledge of the message nor the modulus (bypass message/modulus blinding)

They all apply to a fully blinded atomic multiply-always exponentiation, but...

- when attacking a LIM *Big Mac* refers to the leakage of previous ones (templates)
- refreshing the message blinding at each LIM would thwart *Big Mac* (e.g. $m^\star \leftarrow m^\star + n$)
- *Rosetta* is strictly local: only the leakage of the current LIM is exploited

Rosetta applies even if the message blinding is refreshed at each LIM

(as well as single trace *SAC 2008*, but it is less efficient)

And so what. . . ?

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . .

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

Hopefully, CT-RSA 2013 paper authors propose a fix for the first one.

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

Hopefully, CT-RSA 2013 paper authors propose a fix for the first one.

Using true regular algorithms

While less efficient, the following exponentiation methods resist to *Rosetta*:

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

Hopefully, CT-RSA 2013 paper authors propose a fix for the first one.

Using true regular algorithms

While less efficient, the following exponentiation methods resist to *Rosetta*:

- Montgomerry ladder: $1.5M \rightarrow 1M + 1S$ per exponent bit

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

Hopefully, CT-RSA 2013 paper authors propose a fix for the first one.

Using true regular algorithms

While less efficient, the following exponentiation methods resist to *Rosetta*:

- Montgomerry ladder: $1.5M \rightarrow 1M + 1S$ per exponent bit
- Joye ladder: $1.5M \rightarrow 1M + 1S$ per exponent bit

And so what. . . ?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to *Rosetta*:

1. internally shuffling the order of the single-precision multiplications
2. blinding the operands of each single-precision multiplication

Buts. . . these countermeasures have been broken (to appear at CT-RSA 2013).

Hopefully, CT-RSA 2013 paper authors propose a fix for the first one.

Using true regular algorithms

While less efficient, the following exponentiation methods resist to *Rosetta*:

- Montgomerry ladder: $1.5M \rightarrow 1M + 1S$ per exponent bit
- Joye ladder: $1.5M \rightarrow 1M + 1S$ per exponent bit
- Square always: $1.5M \rightarrow 2S$ per exponent bit

1. Introduction
2. Context of the attack
3. Description of Rosetta
4. Countermeasures
5. Conclusion

Conclusion

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding
  - message and modulus blinding (even if refreshed at each LIM)

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding
  - message and modulus blinding (even if refreshed at each LIM)
- Simulation experiments show that *Rosetta* remains efficient even in the presence of a strong noise level

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding
  - message and modulus blinding (even if refreshed at each LIM)
- Simulation experiments show that *Rosetta* remains efficient even in the presence of a strong noise level

Possible future works

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding
  - message and modulus blinding (even if refreshed at each LIM)
- Simulation experiments show that *Rosetta* remains efficient even in the presence of a strong noise level

Possible future works

- Implement *Rosetta* on a real device

Conclusion

- By exploiting locally the leakage of a LIM, *Rosetta* recovers the sequence of squaring and multiplications using a single trace.
- It threatens both standard and CRT RSA implemented using the most state-of-the-art atomic exponentiation:
  - exponent blinding
  - message and modulus blinding (even if refreshed at each LIM)
- Simulation experiments show that *Rosetta* remains efficient even in the presence of a strong noise level

Possible future works

- Implement *Rosetta* on a real device
- Design other countermeasures which apply to the atomic exponentiation

The end

Thank you for your attention!

Questions?

ROSETTA for Single Trace Analysis

Recovery Of Secret Exponent by Triangular Trace Analysis

Christophe Clavier[1] Benoit Feix[1,2] Georges Gagnerot[1,2]
Christophe Giraud [3] Mylène Roussellet[2] Vincent Verneuil[2]

[1]Université de Limoges
[2]Inside Secure
[3]Oberthur Technologies

Kolkata – 10 December 2012