ROSETTA for Single Trace Analysis

Recovery Of Secret Exponent by Triangular Trace Analysis

Christophe Clavier$^1$
Christophe Giraud$^3$
Benoit Feix$^{1,2}$
Georges Gagnerot$^{1,2}$
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
1. Introduction

2. Context of the attack

3. Description of Rosetta

4. Countermeasures

5. Conclusion
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
1 Introduction

2 Context of the attack

3 Description of Rosetta

4 Countermeasures

5 Conclusion
Blinding the exponentiation

Blinding countermeasure

Blinding \( s = m^d \mod n \) makes use of two random \( \lambda \)-bit integers \( r_d \) and \( r_m \):

\[
(\lambda \geq 32 \text{ bits})
\]

- **exponent** \( d^* \leftarrow d + r_d \cdot \varphi(n) \)
- **message** \( m^* \leftarrow m + r_m \cdot n \)
Blinding the exponentiation

Blinding countermeasure

Blinding \( s = m^d \mod n \) makes use of two random \( \lambda \)-bit integers \( r_d \) and \( r_m \):

\[
(\lambda \geq 32 \text{ bits})
\]

- **Exponent** \( d^* \leftarrow d + r_d \cdot \varphi(n) \)
- **Message** \( m^* \leftarrow m + r_m \cdot n \)

The exponentiation is computed modulo \( 2^\lambda n \):

\[
s = (m + r_m \cdot n)^{d + r_d \cdot \varphi(n)} \mod 2^\lambda n
\]

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**

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

- **Montgomery 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**

- Montgomery 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 $M$: 

<table>
<thead>
<tr>
<th>Regular exponentiation algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Montgomery ladder: $1M + 1S$</td>
</tr>
<tr>
<td>• Joye ladder: $1M + 1S$</td>
</tr>
<tr>
<td>• Square always: $2S$</td>
</tr>
</tbody>
</table>
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 $M$:

- **Montgomery 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: $x^d \mod n$

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: \hspace{1em} $R_0 \leftarrow R_0 \times R_k \mod n$
7: \hspace{1em} $k \leftarrow k \oplus d_i$
8: \hspace{1em} $i \leftarrow i - \neg k$
9: return $R_0$

[\oplus \text{ stands for bitwise X-or}]
[\neg \text{ stands for bitwise negation}]

Considered implementation

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

The fastest regular exponentiation is:

**Algorithm 1** Atomic multiply-always exponentiation

\[ x, n \in \mathbb{N}, d = (d_{v-1}d_{v-2} \ldots d_0)_2 \]

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

**Output:** \( x^d \mod n \)

1. \( R_0 \leftarrow 1 \)
2. \( R_1 \leftarrow x \)
3. \( i \leftarrow v - 1 \)
4. \( k \leftarrow 0 \)
5. \( \textbf{while } i \geq 0 \textbf{ do} \)
6. \( R_0 \leftarrow R_0 \times R_k \mod n \)
7. \( k \leftarrow k \oplus d_i \quad [\oplus \text{ stands for bitwise X-or}] \)
8. \( i \leftarrow i \oplus d_i \quad [\oplus \text{ stands for bitwise negation}] \)
9. \( \textbf{return } R_0 \)

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 \( \text{LIM}(x, y) = x \cdot y \) is computed with a small \( t \)-bit multiplier
- Schoolbook method on \( \ell \)-word integers expressed in base \( b = 2^t \)
  \[
  x = (x_{\ell-1}x_{\ell-2} \cdots x_1x_0)_b \quad y = (y_{\ell-1}y_{\ell-2} \cdots y_1y_0)_b
  \]
Inside the long integer multiplication (LIM)

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

- Each LIM$(x, y) = x \cdot y$ is computed with a small $t$-bit multiplier
- Schoolbook method on $\ell$-word integers expressed in base $b = 2^t$
  \[ x = (x_{\ell-1}x_{\ell-2} \ldots x_1x_0)_b \quad y = (y_{\ell-1}y_{\ell-2} \ldots y_1y_0)_b \]
- Each $k$-th LIM side-channel trace $T^k$ can be split into $\ell^2$ trace segments $T_{i,j}^k$ 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...
Two possible threats: *SAC 2008* attack

Is the multiply-always algorithm a *true* regular exponentiation?

Not really...
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** \( \text{HW}(x \cdot x) \) (uniformly distributed \( x \))
- **Multiplication** \( \text{HW}(x \cdot y) \) (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** \( \text{HW}(x \cdot x) \) (uniformly distributed \( x \))
- **multiplication** \( \text{HW}(x \cdot y) \) (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** \( \text{HW}(x \cdot x) \) (uniformly distributed \( x \))
- **Multiplication** \( \text{HW}(x \cdot y) \) (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 → 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:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Formula</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>square</strong></td>
<td>$\text{HW}(x \cdot x)$</td>
<td>(uniformly distributed $x$)</td>
</tr>
<tr>
<td><strong>multiplication</strong></td>
<td>$\text{HW}(x \cdot y)$</td>
<td>(independant uniformly distributed $x$ and $y$)</td>
</tr>
</tbody>
</table>

- 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:

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 \(m\) 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.
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:

\[
\begin{align*}
\text{first LIM:} & \quad 1 \times m \\
\text{second LIM:} & \quad m \times m
\end{align*}
\]
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

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

\[
x \times y = \ell - 1 \sum_{i=0}^{\ell-1} \sum_{j=0}^{\ell-1} x^i y^j b^i + j
\]

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

\[
M=
\begin{bmatrix}
 x_0 & y_0 & x_0 & y_1 \\
 x_0 & y_0 & x_0 & y_2 \\
 x_0 & y_0 & x_0 & y_3 \\
 x_1 & y_0 & x_1 & y_1 \\
 x_1 & y_0 & x_1 & y_2 \\
 x_1 & y_0 & x_1 & y_3 \\
 x_2 & y_0 & x_2 & y_1 \\
 x_2 & y_0 & x_2 & y_2 \\
 x_2 & y_0 & x_2 & y_3 \\
 x_3 & y_0 & x_3 & y_1 \\
 x_3 & y_0 & x_3 & y_2 \\
 x_3 & y_0 & x_3 & y_3 
\end{bmatrix}
\]

If the LIM is a squaring then

\[
S=
\begin{bmatrix}
 x_0 & x_0 & x_0 & x_1 \\
 x_0 & x_0 & x_0 & x_2 \\
 x_0 & x_0 & x_0 & x_3 \\
 x_1 & x_0 & x_1 & x_1 \\
 x_1 & x_0 & x_1 & x_2 \\
 x_1 & x_0 & x_1 & x_3 \\
 x_2 & x_0 & x_2 & x_1 \\
 x_2 & x_0 & x_2 & x_2 \\
 x_2 & x_0 & x_2 & x_3 \\
 x_3 & x_0 & x_3 & x_1 \\
 x_3 & x_0 & x_3 & x_2 \\
 x_3 & x_0 & x_3 & x_3 
\end{bmatrix}
\]
Principle of the attack

\[ \text{LIM}(x, y) \text{ in base } b = 2^t \text{ by classical schoolbook method:} \]

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

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

$$S = \begin{pmatrix} x_0 x_0 & x_0 x_1 & x_0 x_2 & x_0 x_3 \\ x_1 x_0 & x_1 x_1 & x_1 x_2 & x_1 x_3 \\ x_2 x_0 & x_2 x_1 & x_2 x_2 & x_2 x_3 \\ x_3 x_0 & x_3 x_1 & x_3 x_2 & x_3 x_3 \end{pmatrix}$$
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 \(\ell = 4\):

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

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

On the diagonal square $LIM(\ x, \ y)$ with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1$ \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 $\ell$ trace segments is not so much ($\ell = |n|t$; typical values: 32, 64)

On the two symmetric triangles ($\text{Rosetta}$)

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

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

We expect to detect the conditional triangular collision

Advantage: as much as $(\ell^2 - \ell) / 2$ trace segments
Principle of the attack

On the diagonal

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

Multiplication \[ \text{LIM}(x, y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \quad \forall i \neq j. \]

This gives the opportunity to apply the SAC 2008 attack to the set of diagonal operations \[ x_i \times y_i \text{ of a LIM on a single trace} \]

Drawback: only \[ \ell \text{ trace segments is not so much} \quad (\ell = |n|t; \text{typical values: 32, 64}) \]

On the two symmetric triangles (Rosetta)

Multiplication \[ \text{LIM}(x, y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \quad \forall i \neq j. \]

We expect to detect the conditional triangular collision

Advantage: as much as \[ (\ell^2 - \ell)/2 \text{ trace segments} \]
Principle of the attack

On the diagonal

\textbf{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
Principle of the attack

On the diagonal

- **square**: $LIM(x, y)$ with $x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \; \forall i$
- **multiplication**: $LIM(x, y)$ with $x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \; \forall i$
Principle of the attack

On the diagonal

- **Square**: \( \text{LIM}(x, y) \text{ with } x = y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \ \forall i \)
- **Multiplication**: \( \text{LIM}(x, y) \text{ with } x \neq y \Rightarrow \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \ \forall i \)

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

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 $\ell$ trace segments is not so much $\left( \ell = \frac{|n|}{t} ; \text{typical values: 32, 64} \right)$
Principle of the attack

On the diagonal

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

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

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 \( \ell \) trace segments is not so much  \( (\ell = |n|/\epsilon; \text{typical values: } 32, 64) \)

On the two symmetric triangles  \( \text{(Rosetta)} \)
Principle of the attack

On the diagonal

- **LIM(x, y) with x = y ⇒ Prob(x_i × y_i is a squaring) = 1 ∀i**
- **LIM(x, y) with x ≠ y ⇒ Prob(x_i × y_i is a squaring) ≈ 0 ∀i**

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

**Drawback:** only ℓ trace segments is not so much

(ℓ = \( \frac{|n|}{t} \); typical values: 32, 64)

On the two symmetric triangles (**Rosetta**)

- **LIM(x, y) with x = y ⇒ Prob(x_i × y_j = x_j × y_i) = 1 ∀i ≠ j.**
Principle of the attack

On the diagonal

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

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

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

On the two symmetric triangles (Rosetta)

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

**multiplication** \( \text{LIM}(x, y) \) with \( x \neq y \) \( \Rightarrow \) \( \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \) \( \forall i \neq j \).
Principle of the attack

On the diagonal

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 \\
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

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 \(\ell\) trace segments is not so much \((\ell = \frac{|n|}{\varepsilon}; \text{typical values: 32, 64})\)

On the two symmetric triangles \((\text{Rosetta})\)

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. \\
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.

We expect to detect the conditional \textit{triangular collision}
Principle of the attack

On the diagonal

- **Square** \( \text{LIM}(x, y) \) with \( x = y \) \( \Rightarrow \) \( \text{Prob}(x_i \times y_i \text{ is a squaring}) = 1 \) \( \forall i \)
- **Multiplication** \( \text{LIM}(x, y) \) with \( x \neq y \) \( \Rightarrow \) \( \text{Prob}(x_i \times y_i \text{ is a squaring}) \approx 0 \) \( \forall i \)

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

On the two symmetric triangles (**Rosetta**)

- **Square** \( \text{LIM}(x, y) \) with \( x = y \) \( \Rightarrow \) \( \text{Prob}(x_i \times y_j = x_j \times y_i) = 1 \) \( \forall i \neq j \).
- **Multiplication** \( \text{LIM}(x, y) \) with \( x \neq y \) \( \Rightarrow \) \( \text{Prob}(x_i \times y_j = x_j \times y_i) \approx 0 \) \( \forall i \neq j \).

We expect to detect the conditional **triangular collision**

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

Principle of the attack

Mean Euclidean distance between pairs of trace segments $T_i, j$ and $T_j, i$:

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

Collision-Correlation between two series of trace segments (with same $(i, j)$ ordering):

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

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

$$\hat{\rho}_{\Theta_0, \Theta_1}(t) = \text{Cov}(\Theta_0(t), \Theta_1(t))$$

$$\sigma_{\Theta_0}(t) \sigma_{\Theta_1}(t)$$
Two distinguishers

Mean Euclidean distance

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

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

Mean Euclidean distance

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

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

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

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

Two distinguishers

Mean Euclidean distance

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

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

Collision-Correlation

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

$$\Theta_0 = \{ T_{i,j} \text{ s.t. } 0 \leq i < j \leq \ell - 1 \} \quad (\text{upper right triangle})$$
$$\Theta_1 = \{ T_{j,i} \text{ s.t. } 0 \leq i < j \leq \ell - 1 \} \quad (\text{lower left triangle})$$

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

- $32 \times 32$-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:

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

- $32 \times 32$-bit multiplier ($b = 2^{32}$)
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\text{HW}(x_i)$, $\text{HW}(y_j)$, $\text{HW}((x_i \times y_j) \text{ div } b)$, $\text{HW}((x_i \times y_j) \text{ mod } b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1000 LIM experiments for each (attack, noise level)
Simulation results

Generation of simulated side-channel traces of LIM:

- $32 \times 32$-bit multiplier ($b = 2^{32}$)
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\text{HW}(x_i)$, $\text{HW}(y_j)$, $\text{HW}((x_i \times y_j) \text{ div } b)$, $\text{HW}((x_i \times y_j) \text{ mod } b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1000 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:

- $32 \times 32$-bit multiplier ($b = 2^{32}$)
- Hamming weight leakage model
- Four points of leakage per single-precision mult:
  - $\text{HW}(x_i), \text{HW}(y_j), \text{HW}((x_i \times y_j) \div b), \text{HW}((x_i \times y_j) \mod b)$
- Add a zero-mean Gaussian noise with 3 noise levels: $\sigma \in \{0, 2, 7\}$
- 1000 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 ($\sigma = 0$)

<table>
<thead>
<tr>
<th>Technique</th>
<th>512 bits</th>
<th>768 bits</th>
<th>1024 bits</th>
<th>1536 bits</th>
<th>2048 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Mac</td>
<td>0.986</td>
<td>0.990</td>
<td>0.993</td>
<td>0.994</td>
<td>0.995</td>
</tr>
<tr>
<td>SAC 2008</td>
<td>0.533</td>
<td>0.618</td>
<td>0.734</td>
<td>0.858</td>
<td>0.897</td>
</tr>
<tr>
<td>Big Mac CoCo</td>
<td>0.999</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta ED</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta CoCo</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Technique</th>
<th>512 bits</th>
<th>768 bits</th>
<th>1024 bits</th>
<th>1536 bits</th>
<th>2048 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Mac</td>
<td>0.986</td>
<td>0.990</td>
<td>0.993</td>
<td>0.994</td>
<td>0.995</td>
</tr>
<tr>
<td>SAC 2008</td>
<td>0.533</td>
<td>0.618</td>
<td>0.734</td>
<td>0.858</td>
<td>0.897</td>
</tr>
<tr>
<td>Big Mac CoCo</td>
<td>0.999</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta ED</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta CoCo</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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)
Simulation results

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

<table>
<thead>
<tr>
<th>Technique</th>
<th>512 bits</th>
<th>768 bits</th>
<th>1024 bits</th>
<th>1536 bits</th>
<th>2048 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Mac</td>
<td>0.767</td>
<td>0.775</td>
<td>0.807</td>
<td>0.816</td>
<td>0.818</td>
</tr>
<tr>
<td>SAC 2008</td>
<td>0.546</td>
<td>0.629</td>
<td>0.717</td>
<td>0.805</td>
<td>0.855</td>
</tr>
<tr>
<td>Big Mac CoCo</td>
<td>0.981</td>
<td>0.998</td>
<td>0.999</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta ED</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta CoCo</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

**Table**: Success rate with a moderate noise, $\sigma = 2$
## Success rate with a moderate noise ($\sigma = 2$)

<table>
<thead>
<tr>
<th>Technique</th>
<th>512 bits</th>
<th>768 bits</th>
<th>1024 bits</th>
<th>1536 bits</th>
<th>2048 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Mac</td>
<td>0.767</td>
<td>0.775</td>
<td>0.807</td>
<td>0.816</td>
<td>0.818</td>
</tr>
<tr>
<td>SAC 2008</td>
<td>0.546</td>
<td>0.629</td>
<td>0.717</td>
<td>0.805</td>
<td>0.855</td>
</tr>
<tr>
<td>Big Mac CoCo</td>
<td>0.981</td>
<td>0.998</td>
<td>0.999</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta ED</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Rosetta CoCo</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

*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 ($\sigma = 7$)

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

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

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

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
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  \((\text{bypass exponent blinding})\)
- do not require the knowledge of the message nor the modulus  \((\text{bypass message/modulus blinding})\)

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. $m^* \leftarrow m^* + n$)
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. $m^* \leftarrow m^* + n$)
- *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. $m^* \leftarrow m^* + n$)
- *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*:

1. Internally shuffling the order of the single-precision multiplications
2. Blinding the operands of each single-precision multiplication
And so what...?

LIM leakage mitigation

Two countermeasures proposed against Horizontal CPA also apply to Rosetta:

- internally shuffling the order of the single-precision multiplications

But 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:

- Montgomery ladder: 1
- Joye ladder: 1
- Square always: 1
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*:

- Montgomery ladder: \(1 \cdot 2^{5 \cdot \text{S per exponent bit}}\)
- Joye ladder: \(1 \cdot 2^{5 \cdot \text{S per exponent bit}}\)
- Square always: \(1 \cdot 2^{5 \cdot \text{S per exponent bit}}\)
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*:

- Montgomery 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*:
- internally shuffling the order of the single-precision multiplications
- 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*:
- **Montgomery 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*:

- **Montgomery 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
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
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
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\textsuperscript{1} \quad Benoit Feix\textsuperscript{1,2} \quad Georges Gagnerot\textsuperscript{1,2}
Christophe Giraud\textsuperscript{3} \quad Mylène Roussellet\textsuperscript{2} \quad Vincent Verneuil\textsuperscript{2}

\textsuperscript{1}Université de Limoges
\textsuperscript{2}Inside Secure
\textsuperscript{3}Oberthur Technologies

Kolkata – 10 December 2012