Frédéric AMIEL AMESYS
Benoit FEIX INSIDE CONTACTLESS

On the BRIP Algorithms Security on RSA

WISTP' 08

Outline

- Introduction
  - Previous works
  - Our Attacks
- Collision Attacks on RSA BRIP Implementation
  - Remember Collision Attacks
  - Improved Collision Attacks Analysis on BRIP
- BRIP and Improvements for Exponentiations (RSA)
  - Use Montgomery Arithmetic to improve efficiency
  - A combined Power Analysis Attack
- Conclusion

Introduction

- Power Analysis (PA) is a **concrete threat** against embedded cryptosystems.

- Any **naive implementation** succumbs to such attacks:

  Public key primitives :

  Modular exponentiation : RSA, DH, etc.

  Scalar Product in Elliptic Curves Schemes: ECDSA, El Gamal, etc.

  Secret key algorithms :

  DES, AES, HMAC, etc…

Introduction

Some previous attacks

- P. C. Kocher..Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. CRYPTO 1996.
- P. C. Kocher, J. Jaffe, and B. Jun. *Differential power analysis*. CRYPTO '99,
- E. Brier, C. Clavier, and F. Olivier. *Correlation power analysis with a leakage model*. CHES 2004.
- P-A. Fouque and F. Valette.*The Doubling Attack - why upwards is better than downwards.* CHES 2003.
- S-M. Yen, W-C. Lien, S. Moon, and J. *Ha. Power Analysis by Exploiting Chosen Message and Internal Collisions - Vulnerability of Checking Mechanism for RSA-Decryption*. Mycrypt 2005.
- F. Amiel, B. Feix, and K. Villegas. *Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms*. SAC 2007.

Introduction

Our Work

- Improvement of the BRIP Power Collision Analysis

  Collision analysis on **blinded** messages.

- Proposal for an efficient implementation of BRIP

  Use **Montgomery arithmetic** to improve efficiency,

  Sensitive to a **combined** Power Analysis :

  SPA + D(C)PA can defeat this implementation.

Introduction

Reminder on RSA

- RSA based on modular exponentiation :

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

s : Signature

m : Message

n : Modulus (Public)

d : Exponent (Private)

Typically : $|n| = |s| = |m| = |d| = k$ bits.

n usually equals to 768, 1024, 1536 or 2048

RSA

A Naïve Implementation

- Classical « Square and Multiply » algorithm :

Algorithm 3.1 Exponentiation from left to right

INPUT: integers m and n such that $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\dots d_1 d_0)_2$
OUTPUT: ModExp$(m,d,n) = m^d \mod n$

Step 1. $a = 1$

Step 2. for i from $k-1$ to 0 **do**

$a = a \times a \mod n$

If $d_i = 1$ Then $a = a \times m \mod n$

Step 3. Return(a)

Power Analysis on Naive RSA

- **Timing Attacks** (TA) :
  Conditional branching : $\boxed{\text{If } d_i = 1}$
  Not constant time : $\boxed{a = a \times a \mod n}$ vs $\boxed{a = a \times m \mod n}$
- **Simple Power Analysis** (SPA) :
  Squaring power pattern is different from multiplication
  Chosen message attack : $\boxed{a = a \times m \mod n}$
- **Statistical Attacks** (CPA, DPA) :
  Making hypothesis on $\boxed{d_i}$ gives ability to predict output value of $\boxed{a = a \times a \mod n}$ or $\boxed{a = a \times m \mod n}$.
- Not exhaustive (**Collision/Template Attacks**/etc.) !

BRIP as countermeasure

- Introduced on RSA by Yen/Lien/Moon/Ha in '04,
- SPA/D(C)PA/TA resistant :

Algorithm 3.2 BRIP Exponentiation from left to right

INPUT: integers m and n such that $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\ldots d_1 d_0)_2$
OUTPUT: BRIP_Exp$(m,d,n) = m^d \mod n$

Step 1. If $m = 1$ **Return**(1)
Step 2. If $m = n - 1$ **Return**$((-1)^{d_0} \mod n)$
Step 3. Choose a random value v and compute $v^{-1} \mod n$
Step 4. $a = v$, $m_0 = v^{-1} \mod n$, $m_1 = v^{-1}.m \mod n$

→ Data Randomization

Step 5. **for** i from $k-1$ to 0 **do**
  $a = a \times a \mod n$
  $a = a \times m_{d_i} \mod n$

→ Constant time

Step 6. $a = a \times m_0$
Step 7. **Return**(a)

BRIP Drawbacks

- **Complexity : 2**

Step 5. for i from $k-1$ to 0 **do**

$a = a \times a \mod n$

$a = a \times m_{d_i} \mod n$

Improvement : Implement BRIP exponentiation using k-ary method.

- Need computation of an **inverse modulo**

Step 3. Choose a random value v and compute $v^{-1} \mod n$

Improvement : Use Montgomery modular arithmetic to ease computation (Ciet/Feix'05) .

Collision Attack on BRIP (1/3)

- Based on **assumption**:

> "Each computation in an embedded Public Key Implementation has a **power signature characteristic from data manipulated**."

- Ability to detect identical computations through power execution traces.

Collision Attack on BRIP (2/3)

- First Publications on ECC and RSA by Fouque and Valette.
- Yen & al. : Ability to find out private exponent value through collision detection by using particular chosen message value $\pm 1 \bmod n$,

Collision Attack on BRIP (3/3)

- Taken in **consideration** in BRIP :

Algorithm 3.2 BRIP Exponentiation from left to right

INPUT: integers m and n such that $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\dots d_1 d_0)_2$
OUTPUT: BRIP_Exp$(m,d,n) = m^d \mod n$

Step 1. If $m = 1$ **Return**(1)
Step 2. If $m = n - 1$ **Return**$((-1)^{d_0} \mod n)$
Step 3. Choose a random value v and compute $v^{-1} \mod n$
Step 4. $a = v$, $m_0 = v^{-1} \mod n$, $m_1 = v^{-1}.m \mod n$
Step 5. **for** i from $k-1$ to 0 **do**
    $a = a \times a \mod n$
    $a = a \times m_{d_i} \mod n$
Step 6. $a = a \times m_0$
Step 7. **Return**(a)

How message blinding protect against such attack ?

Collision Attack on BRIP (1/8)

- We analyze how collision attacks can endanger BRIP implementation beyond message blinding.

- Let's consider that ***v***, an ***h***-bit random value is used for BRIP blinding scheme :

> **Step 3.** Choose a random value v and compute $v^{-1} \mod n$

Collision Attack on BRIP (2/8)

- Condition of the attack :
  Find two colliding traces with $\{m, v_1\}$ and $\{-m, v_2\}$ and $\boldsymbol{v_1 = v_2}$
  i.e. sharing **same random value**.

	Message	Square	Message Multiplication ($d_i = 1$)	Square
d_i=1	m	$\left[m^{d''}.v_1\right]^2$	$\left[(m^{2.d''}).v_1^2\right] \times \left[m.v_1^{-1}\right]$	$\left[(m^{2.d''+1}).v_1\right]^2$
	$-m$	$\left[(-m)^{d''}.v_2\right]^2$	$\left[(m^{2.d''}).v_2^2\right] \times \left[-m.v_2^{-1}\right]$	$\left[((-m)^{2.d''+1}).v_2\right]^2$
	Collision if $v_1 = v_2$	-	No	No
	Message	**Square**	**Fake Multiplication ($d_i = 0$)**	**Square**
d_i=0	m	$\left[m^{d''}.v_1\right]^2$	$\left[(m^{2.d''}).v_1^2\right] \times \left[v_1^{-1}\right]$	$\left[(m^{2.d''}).v_1\right]^2$
	$-m$	$\left[(-m)^{d''}.v_2\right]^2$	$\left[(m^{2.d''}).v_2^2\right] \times \left[v_2^{-1}\right]$	$\left[(m^{2.d''}).v_2\right]^2$
	Collision if $v_1 = v_2$	-	Yes	Yes

In case of such event, **collisions appear for $d_i = 0$** .

Collision Attack on BRIP (3/8)

- **Attack process :**

Algorithm 4.5 BRIP Collision Attack

INPUT: $s = \text{RSA-BRIP}(m, d)$, $s' = \text{RSA-BRIP}(-m, d)$
OUTPUT: Secret exponent d

Step 1. Choose a random value m in $[2, n-2]$.
Step 2. Collect k traces $(C_0, ..., C_{k-1})$ of BRIP execution with m as input message.
Step 3. Collect k traces $(C'_0, ..., C'_{k-1})$ of BRIP execution with $-m$ as input message.
Step 4. Find traces C_i and C'_j such as both traces are colliding on each BRIP Fake Multiply.
Step 5. Compute $S = |C_i - C'_j|$.
Step 6. Each non zero difference on S identify a true multiplication, i.e. $d_i = 1$

- **Step 4** is realistic as on average **|*d*| operations collide** ! RSA-2048 gives 2048 collision occurrences, this make attack **concrete** in term of **signal processing** even with the presence of **hardware countermeasures**.

Collision Attack on BRIP (4/8)

- **Probability** of detecting at least **two colliding couples in two sets of *k* curves** can be expressed as :

$$p_{collision} \simeq 1 - e^{-((k^2)/|h|)}$$

- **Evaluating formula** with a typical random length :

h	k	collision
32	78000	0.507
32	$2^{17} \approx 131072$	0.864
32	161000	0.951
32	200000	0.990
32	$2^{18} \approx 262144$	0.999

- **Dramatically low number of curves for 32-bit random !**

Collision Attack on BRIP (5/8)

- For different random bit size :

h	k	collision probability
16	$2^9 = 512$	0.864
16	$2^{10} = 1024$	0.999
64	5.1×10^9	0.505
64	2^{33}	0.864
64	2^{34}	0.999
96	3.3×10^{14}	0.497
96	2^{48}	0.864
96	2^{49}	0.999

- As a **security factor**, we recommend use of **96-bit random values**.

Collision Attack on BRIP (6/8)

- With ***k*-ary exponentiation** method, k = 2 here :

Square & Multiply

Algorithm 3.1 Exponentiation from left to right

INPUT: integers m and n such that $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\dots d_1d_0)_2$
OUTPUT: ModExp$(m,d,n)= m^d \mod n$

Step 1. $a = 1$
Step 2. for i from $k-1$ to 0 **do**
  $a = a \times a \mod n$
  If $d_i = 1$ Then $a = a \times m \mod n$
Step 3. Return(a)

k-ary

Algorithm 3.3 2-ary Exponentiation from left to right

INPUT: integers m,n such that $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\dots d_1d_0)_2$
OUTPUT: $m^d \mod n$

Step 1. $a = 1$
Step 2. Compute $m_j = m^j \mod n, \quad j = 0,\dots,3$
Step 3. for i from $k-1$ to 0 by 2 **do**
  $a = a \times a \mod n$
  $a = a \times a \mod n$
  $a = a \times m_{(2.d_i+d_{i-1})} \mod n$
Step 4. Return(a)

- How behave collision attack with such implementation ?

Collision Attack on BRIP (7/8)

- Collisions happens too :

d_i=00

Message	Square	Square	$M_{(2d_i+d_{i-1}=0)}$ MontMul
m	$MM(m^{d''}.r^{v_1}, m^{d''}.r^{v_1})$	$MM(m^{2.d''}.r^{v_1}, m^{2.d''}.r^{v_1})$	$MM(m^{4.d''}.r^{4v_1}, r^{-3v_1})$
$-m$	$MM((-m)^{d''}.r^{v_2}, (-m)^{d''}.r^{v_2})$	$MM(m^{2.d''}.r^{v_2}, m^{2.d''}.r^{v_2})$	$MM(m^{4.d''}.r^{4v_2}, r^{-3v_2})$
If $v_1 = v_2$	-	Yes	Yes

d_i=01

Message	Square	Square	$M_{(2d_i+d_{i-1}=1)}$ MontMul
m	$MM(m^{d''}.r^{v_1}, m^{d''}.r^{v_1})$	$MM(m^{2.d''}.r^{v_1}, m^{2.d''}.r^{v_1})$	$MM(m^{4.d''}.r^{4v_1}, m.r^{-3v_1})$
$-m$	$MM((-m)^{d''}.r^{v_2}, (-m)^{d''}.r^{v_2})$	$MM(m^{2.d''}.r^{v_2}, m^{2.d''}.r^{v_2})$	$MM(m^{4.d''}.r^{4v_2}, (-m).r^{-3v_2})$
If $v_1 = v_2$	-	Yes	No

d_i=10

Message	Square	Square	$M_{(2d_i+d_{i-1}=2)}$ MontMul
m	$MM(m^{d''}.r^{v_1}, m^{d''}.r^{v_1})$	$MM(m^{2.d''}.r^{v_1}, m^{2.d''}.r^{v_1})$	$MM(m^{4.d''}.r^{4v_1}, m^2.r^{-3v_1})$
$-m$	$MM((-m)^{d''}.r^{v_2}, (-m)^{d''}.r^{v_2})$	$MM(m^{2.d''}.r^{v_2}, m^{2.d''}.r^{v_2})$	$MM(m^{4.d''}.r^{4v_2}, m^2.r^{-3v_2})$
If $v_1 = v_2$	-	Yes	Yes

d_i=11

Message	Square	Square	$M_{(2d_i+d_{i-1}=3)}$ MontMul
m	$MM(m^{d''}.r^{v_1}, m^{d''}.r^{v_1})$	$MM(m^{2.d''}.r^{v_1}, m^{2.d''}.r^{v_1})$	$MM(m^{4.d''}.r^{4v_1}, m^3.r^{-3v_1})$
$-m$	$MM((-m)^{d''}.r^{v_2}, (-m)^{d''}.r^{v_2})$	$MM(m^{2.d''}.r^{v_2}, m^{2.d''}.r^{v_2})$	$MM(m^{4.d''}.r^{4v_2}, (-m)^3.r^{-3v_2})$
If $v_1 = v_2$	-	Yes	No

- Through collision analysis {00,10} can be identified from {01,11} , **half** of ***d*** can be therefore recovered.

Collision Attack on BRIP (8/8)

- With **RSA CRT** (Chinese Remainder Theorem), $n=p.q$:

  Given two input messages $\pm m \bmod n$

  Relation is **maintained** as:

$$\begin{array}{|l|} \hline m_2 = -m_1 \bmod n \\ m_2 = -m_1 \bmod n \\ \hline \end{array} \rightarrow \begin{array}{|l|} \hline m_2 \bmod p = -m_1 \bmod p \\ m_2 \bmod q = -m_1 \bmod q \\ \hline \end{array}$$

  Colliding power trace both exponentiations would leak secret exponents **if** message is **randomized once at the beginning.**

- Collision Analysis applies identically !

Implementing BRIP RSA (1/6)

- BRIP RSA needs an **inverse modulo computation :**
  Costly operation in term of execution time,
  Major drawback of the countermeasure **must be avoided**.

- Trick from Ciet & Feix '04, use **Montgomery Modular Arithmetic** :

$$\text{MontMult}(a, b, n) = a \cdot b \cdot R^{-1} \bmod n$$
$$\text{MontMult}(1, 1, n) = R^{-1} \bmod n$$

to compute $v = R^s$ and $v^{-1} = R^{-s}$ with s a random value.

- **Pertinent** : Montgomery Modular Arithmetic is **well spread** in **Public Key hardware accelerators**.

Implementing BRIP RSA (2/6)

- Consider **exponentiation using Montgomery arithmetic**:

$$\text{MontExp}(x,e,n) = x^e \cdot R \bmod n$$

Idea is to set: $e = s$ (BRIP RSA random value)
$x = R^{-1}$, obtained from MontMult(1,1,n)

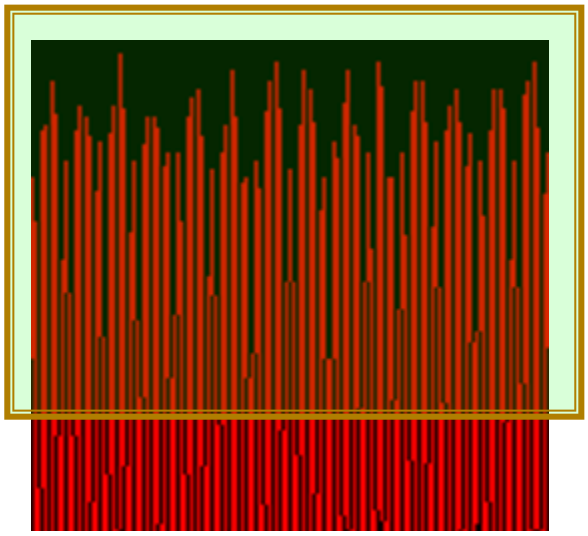
and then compute: $\text{MontExp}(R^{-1}, s, n) = R^{-s} \cdot R$

- **Inverse modulo** *n* is efficiently replaced by an **exponentiation with small exponent** !

Implementing BRIP RSA (3/6)

- **Seducing** computation method **BUT** introduces an **SPA leakage** !

Step 3. Choose a random value v and compute $v^{-1} \mod n$



R^s computation
MontExp(R, s, N)

R^-s computation
MontExp(R^-1, s, N)

Implementing BRIP RSA (4/6)

- What happen ? In **exponentiation loop** and each time $s_i = 1$, following computation is executed :

$$\text{MontMult}(Acc, R^{-1} \cdot R, n) = \text{MontMult}(Acc, 1, n)$$

- Due to Hamming weight of 2nd operand ('1'), *Multiply* becomes characteristic. Each *Multiply* operations can be identified by its low consumption.
Random value is therefore **revealed** !

Implementing BRIP RSA (5/6)

- Once the **random value** is **recovered** trough SPA, **internal values** in BRIP RSA can be **guessed**.

  Therefore any power analysis attacks based on **statistical analysis** (DPA, CPA) can be envisaged.

- **Experimentally** less than **one thousand of curves** are needed to recover **whole exponent value** (by using **CPA**).

Implementing BRIP RSA (6/6)

- **Simple tweak** to counteract SPA is to compute :

$$
\begin{aligned}
R^{-s} \bmod n &= -(-(R^{-1}))^{s} \bmod n \\
&= n - \mathrm{ModExp}(n\text{-}R^{-1}, s, n) \quad \text{if } s \text{ even} \\
&= \mathrm{ModExp}(n\text{-}R^{-1}, s, n) \quad \text{if } s \text{ odd}
\end{aligned}
$$

This concretely replace **1** replaced by ***n*-1** as operand of MontMult(*x*, *y*, *n*) operator.

***n*-1** has no more a **low hamming weight**: **SPA avoided**.

- **Sufficient ?**

Most advanced attacks should be considered too (**Template Attacks**, etc.) which could be used to recover **part of the random value**.

Adding countermeasures in R^s and R^{-s} computations will reduce at same time **performances** and therefore **interest** for BRIP RSA.

Conclusion

- Analysis presented :

  How collision detection can endanger BRIP RSA :

  - Avoid random values < 64 bits,
  - Using padding schemes prevent against such attack,
  - Analysis can be extended to others exponentiation methods.

  Montgomery inversion trick :

  - Must be implemented carefully.

Questions & Answers