On the Use of RSA Public Exponent to Improve Implementation Efficiency and Side-Channel Resistance

Christophe Giraud

Speaker: Gilles Piret

Oberthur Technologies
Cryptography & Security Group

COSADE 2014

Agenda

1 Introduction

2 A New Approach

3 Conclusion

Agenda

1. Introduction
2. A New Approach
3. Conclusion

Notations:

- $N = p \cdot q$,
- $d \cdot e = 1 \bmod \varphi(N)$

Standard RSA

- Signature

$$S = m^d \bmod N$$

- Public Verification

$$S^e \bmod N \stackrel{?}{=} m$$

RSA on Embedded Systems

Notations:

- $i_q = q^{-1} \bmod p$
- $d_p = d \bmod p - 1$
- $d_q = d \bmod q - 1$

RSA CRT-Based

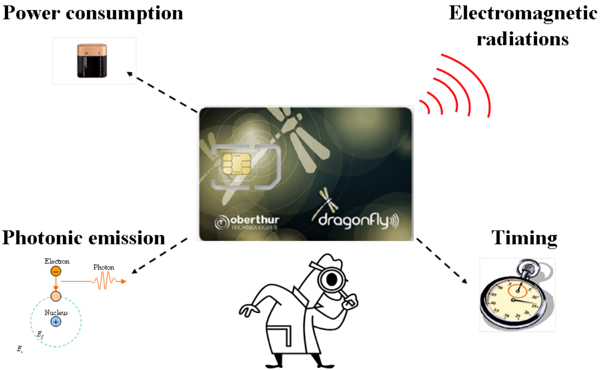
- Signature

$$\begin{aligned} S_p &= m^{d_p} \bmod p, \\ S_q &= m^{d_q} \bmod q, \\ S &= CRT(S_p, S_q) = S_q + q(i_q(S_p - S_q) \bmod p) \end{aligned}$$

- Public Verification

$$S^e \bmod N \stackrel{?}{=} m$$

Side Channel Analysis



Message and Exponents

Use masking to manipulate data independent from secret exponents

- Message masking: $S'_p = (m + k_0 \cdot p)^{d_p} \bmod 2^{64} \cdot p$

Message and Exponents

Use masking to manipulate data independent from secret exponents

- Message masking: $S'_p = (m + k_0 \cdot p)^{d_p} \bmod 2^{64} \cdot p$
- Exponent masking : $S'_p = (m + k_0 \cdot p)^{d_p + k_1 \cdot (p-1)} \bmod 2^{64} \cdot p$

Side-Channel Countermeasures

Message and Exponents

Use masking to manipulate data independent from secret exponents

- Message masking: $S'_p = (m + k_0 \cdot p)^{d_p} \bmod 2^{64} \cdot p$
- Exponent masking : $S'_p = (m + k_0 \cdot p)^{d_p + k_1 \cdot (p-1)} \bmod 2^{64} \cdot p$

Exponentiation

The sequence of modular operations must not depend on the exponent

- Ex: S&M Always, Atomicity, Montgomery Ladder, ...

Message and Exponents

Use masking to manipulate data independent from secret exponents

- Message masking: $S'_p = (m + k_0 \cdot p)^{d_p} \bmod 2^{64} \cdot p$
- Exponent masking : $S'_p = (m + k_0 \cdot p)^{d_p + k_1 \cdot (p-1)} \bmod 2^{64} \cdot p$

Exponentiation

The sequence of modular operations must not depend on the exponent

- Ex: S&M Always, Atomicity, Montgomery Ladder, ...

Two different modular operations must be independent

- Ex: Single-precision multiplication order randomization, systematic operands blinding, ...

If the computation of S_p is disturbed

The faulty signature $S^{\lightning}$ satisfies: $\begin{cases} S^{\lightning} \not\equiv S \bmod p \\ S^{\lightning} \equiv S \bmod q \end{cases}$

The secret prime q can be recovered by computing:

$$\gcd(S^{\lightning} - S, N) \text{ or } \gcd({S^{\lightning}}^{e} - m, N)$$

The most natural countermeasure

One can check the correctness of S :

$$S^e \bmod N \stackrel{?}{=} m$$

Very efficient since 99.95% of the public exponent $e \leq 2^{16} + 1$ in practice.

Agenda

1 Introduction

2 A New Approach

3 Conclusion

Public Exponent Exploitation

- Used for 15 years to counteract Fault Injection
- But no study to improve RSA performance and side-channel resistance !!

Public Exponent Exploitation

- Used for 15 years to counteract Fault Injection
- But no study to improve RSA performance and side-channel resistance !!
- This can be explained by the fact that e is not always provided as input (e.g. JavaCard)

Motivation

Public Exponent Exploitation

- Used for 15 years to counteract Fault Injection
- But no study to improve RSA performance and side-channel resistance !!
- This can be explained by the fact that *e* is not always provided as input (e.g. JavaCard)

However...

- 99.95% of the RSA public exponents belong to: $\{3, 5, 7, 11, 13, 17, 19, 21, 23, 35, 41, 47, 2^8 + 1, 2^{16} \pm 1\}$
- The public exponent can thus be efficiently recovered by using only 1 multiplications and 15 comparisons at most.

A Useful Relation

$$i_q \cdot S_p \equiv (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p$$

A Useful Relation

$$i_q \cdot S_p \equiv (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p$$

Proof:

$$\begin{aligned} (m \cdot q^e)^{d_p - 1} &\equiv m^{d_p - 1} \cdot q^{e \cdot (d_p - 1)} \bmod p \\ &\equiv m^{d_p - 1} \cdot q^{e \cdot d_p - e} \bmod p \\ &\equiv m^{d_p - 1} \cdot q^{1 - e} \bmod p \end{aligned}$$

A Useful Relation

$$i_q \cdot S_p \equiv (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p$$

Proof:

$$\begin{aligned}(m \cdot q^e)^{d_p-1} &\equiv m^{d_p-1} \cdot q^{e\cdot(d_p-1)} \bmod p\\ &\equiv m^{d_p-1} \cdot q^{e\cdot d_p - e} \bmod p\\ &\equiv m^{d_p-1} \cdot q^{1-e} \bmod p\end{aligned}$$

Therefore

$$\begin{aligned}(m \cdot q^e)^{d_p-1} \cdot m \cdot q^{e-2} &\equiv m^{d_p-1} \cdot q^{1-e} \cdot m \cdot q^{e-2} \bmod p\\ &\equiv m^{d_p} \cdot q^{1-e+e-2} \bmod p\\ &\equiv m^{d_p} \cdot q^{-1} \bmod p\end{aligned}$$

□

Our Relations

$$\begin{cases} i_q \cdot S_p \equiv (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p \\ i_p \cdot S_q \equiv (m \cdot p^e)^{d_q - 1} \cdot m \cdot p^{e-2} \bmod q \end{cases}$$

Our Relations

$$\begin{cases} i_q \cdot S_p \equiv (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p \\ i_p \cdot S_q \equiv (m \cdot p^e)^{d_q - 1} \cdot m \cdot p^{e-2} \bmod q \end{cases}$$

Gauss Recombination

$$S = p \cdot i_p \cdot S_q + q \cdot i_q \cdot S_p \bmod N$$

which is equivalent to:

$$S = p \cdot (i_p \cdot S_q \bmod q) + q \cdot (i_q \cdot S_p \bmod p) \bmod N$$

General Description

Gauss Recombination

$$S = p \cdot (i_p \cdot S_q \bmod q) + q \cdot (i_q \cdot S_p \bmod p) \bmod N$$

So the RSA signature can be computed as

$$S = p \cdot S1_q + q \cdot S1_p \bmod N$$

where

$$\begin{aligned} S1_p &= (m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \bmod p, \\ S1_q &= (m \cdot p^e)^{d_q - 1} \cdot m \cdot p^{e-2} \bmod q. \end{aligned}$$

Non Secure Implementations Comparison

Traditional implementation	Our Approach
$S_p \leftarrow m^{d_p} \bmod p$	$q_1 \leftarrow m \cdot q^{e-2} \bmod p$ $q_2 \leftarrow q_1 \cdot q^2 \bmod p$ $S1_p \leftarrow q_2^{d_p-1} \cdot q_1 \bmod p$

Non Secure Implementations Comparison

Traditional implementation	Our Approach
$S_p \leftarrow m^{d_p} \bmod p$	$q_1 \leftarrow m \cdot q^{e-2} \bmod p$ $q_2 \leftarrow q_1 \cdot q^2 \bmod p$ $S1_p \leftarrow q_2^{d_p-1} \cdot q_1 \bmod p$
$S_q \leftarrow m^{d_q} \bmod q$	$p_1 \leftarrow m \cdot p^{e-2} \bmod q$ $p_2 \leftarrow p_1 \cdot p^2 \bmod q$ $S1_q \leftarrow p_2^{d_q-1} \cdot p_1 \bmod q$

Non Secure Implementations Comparison

Traditional implementation	Our Approach
$S_p \leftarrow m^{d_p} \bmod p$	$q_1 \leftarrow m \cdot q^{e-2} \bmod p$ $q_2 \leftarrow q_1 \cdot q^2 \bmod p$ $S1_p \leftarrow q_2^{d_p-1} \cdot q_1 \bmod p$
$S_q \leftarrow m^{d_q} \bmod q$	$p_1 \leftarrow m \cdot p^{e-2} \bmod q$ $p_2 \leftarrow p_1 \cdot p^2 \bmod q$ $S1_q \leftarrow p_2^{d_q-1} \cdot p_1 \bmod q$
$S \leftarrow S_q + q \cdot (i_q \cdot (S_p - S_q) \bmod p)$	$S \leftarrow p \cdot S1_q + q \cdot S1_p \bmod N$

Non Secure Implementations Comparison

Performances on 2048-bit RSA

- If $e = 3$: $+0.3\%$ modular operations
- If $e = 2^{16} + 1$: $+2.2\%$ modular operations

Non Secure Implementations Comparison

Performances on 2048-bit RSA

- If $e = 3$: $+0.3\%$ modular operations
- If $e = 2^{16} + 1$: $+2.2\%$ modular operations

But i_q is not required anymore!

- Gain of $\log_2(N)/2 - \log_2(e)$ bits of memory to store the key
- This represents 125 bytes for a 2048-bit RSA with $e = 2^{16} + 1$

Secure Implementation Consideration

Against SCA

- Message blinding, e.g. $m \to m + k_0 pq$
- Exponents blinding, e.g. $d_p \to d_p + k_1(p-1)$
- Secure Exponentiations

Against FA

- Signature verification, i.e. $S^e \bmod N \stackrel{?}{=} m$

Secure Implementation Consideration

Against SCA

- **Message blinding, e.g. $\mathbf{m} \rightarrow \mathbf{m} + \mathbf{k_0 pq}$**
- Exponents blinding, e.g. $d_p \rightarrow d_p + k_1(p-1)$
- Secure Exponentiations

Against FA

- Signature verification, i.e. $S^e \bmod N \stackrel{?}{=} m$

A Free Blinding Message Method

Another remark

$$(m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \equiv i_q \cdot S_p \bmod p$$

Another remark

$$(m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \equiv i_q \cdot S_p \bmod p$$

By replacing q by $q' = q \cdot r \bmod p$ (with r random):

$$(m \cdot q'^e)^{d_p - 1} \cdot m \cdot q'^{e-2} \equiv i_q \cdot S_p \cdot r^{-1} \bmod p$$

A Free Blinding Message Method

Another remark

$$(m \cdot q^e)^{d_p - 1} \cdot m \cdot q^{e-2} \equiv i_q \cdot S_p \bmod p$$

By replacing q by $q' = q \cdot r \bmod p$ (with r random):

$$(m \cdot q'^e)^{d_p - 1} \cdot m \cdot q'^{e-2} \equiv i_q \cdot S_p \cdot r^{-1} \bmod p$$

So a randomized RSA signature can be computed from

$$\begin{aligned} S' &= p \cdot S1'_q + q \cdot S1'_p \bmod N \\ &= r^{-1} \cdot S \bmod N \end{aligned}$$

where

$$\begin{aligned} S1'_p &= (m \cdot q'^e)^{d_p - 1} \cdot m \cdot q'^{e-2} \bmod p, \\ S1'_q &= (m \cdot p'^e)^{d_q - 1} \cdot m \cdot p'^{e-2} \bmod q. \end{aligned}$$

oberthur TECHNOLOGIES THE M COMPANY

Secure Implementations Comparison*

Traditional implementation	Our Approach
$m' \leftarrow m + k_0 pq$	$q' \leftarrow q \cdot r \bmod p$
	$q_1 \leftarrow m \cdot q'^{e-2} \bmod p$
$S_p \leftarrow m'^{d_p} \bmod 2^{64} p$	$q_2 \leftarrow q_1 \cdot q'^2 \bmod p$
	$S1_p \leftarrow q_2^{d_p - 1} \cdot q_1 \bmod p$

oberthur TECHNOLOGIES THE M COMPANY

Secure Implementations Comparison*

Traditional implementation	Our Approach
$m' \leftarrow m + k_0 pq$	$q' \leftarrow q \cdot r \bmod p$
	$q_1 \leftarrow m \cdot q'^{e-2} \bmod p$
$S_p \leftarrow m'^{d_p} \bmod 2^{64} p$	$q_2 \leftarrow q_1 \cdot q'^2 \bmod p$
	$S1_p \leftarrow q_2^{d_p - 1} \cdot q_1 \bmod p$
	$p' \leftarrow p \cdot r \bmod q$
	$p_1 \leftarrow m \cdot p'^{e-2} \bmod q$
$S_q \leftarrow m'^{d_q} \bmod 2^{64} q$	$p_2 \leftarrow p_1 \cdot p'^2 \bmod q$
	$S1_q \leftarrow p_2^{d_q - 1} \cdot p_1 \bmod q$

Secure Implementations Comparison*

Traditional implementation	Our Approach
$m' \leftarrow m + k_0 pq$	$q' \leftarrow q \cdot r \bmod p$
	$q_1 \leftarrow m \cdot {q'}^{e-2} \bmod p$
$S_p \leftarrow {m'}^{d_p} \bmod 2^{64} p$	$q_2 \leftarrow q_1 \cdot {q'}^2 \bmod p$
	$S1_p \leftarrow q_2^{d_p - 1} \cdot q_1 \bmod p$
	$p' \leftarrow p \cdot r \bmod q$
	$p_1 \leftarrow m \cdot {p'}^{e-2} \bmod q$
$S_q \leftarrow {m'}^{d_q} \bmod 2^{64} q$	$p_2 \leftarrow p_1 \cdot {p'}^2 \bmod q$
	$S1_q \leftarrow p_2^{d_q - 1} \cdot p_1 \bmod q$
$S \leftarrow S_q + q \cdot (i_q \cdot (S_p - S_q) \bmod p)$	$S \leftarrow p \cdot S1_q + q \cdot S1_p \bmod N$
$S^e \stackrel{?}{\equiv} m \bmod N$	$(r \cdot S)^e \stackrel{?}{\equiv} m \bmod N$

*: For clarity reasons, exponent and exponentiation countermeasures are not represented.

Comparison with State-of-the-Art Approach

- Performances of the coprocessors $\sim$ Operands size.
  - Additive masking extends the length of the operands.
  - Multiplicative masking doesn't!

Comparison with State-of-the-Art Approach

- **Performances of the coprocessors $\sim$ Operands size.**
  - **Additive masking extends the length of the operands.**
  - **Multiplicative masking doesn't!**
  - Analysis on a smart card providing a 32-bit modular multiplication co-processor:

CRT-RSA key size	Performance improvement
1024	14.2%
2048	8.2%

oberthur TECHNOLOGIES THE M COMPANY

Comparison with State-of-the-Art Approach

- **Performances of the coprocessors $\sim$ Operands size.**
  - **Additive masking extends the length of the operands.**
  - **Multiplicative masking doesn't!**
  - Analysis on a smart card providing a 32-bit modular multiplication co-processor:

CRT-RSA key size	Performance improvement
1024	14.2%
2048	8.2%

- The key parameter i_q is not required
  $\Rightarrow$ gain of $\log_2(N)/2$ bits of memory to store the key

Agenda

1. Introduction
2. A New Approach
3. Conclusion

Conclusion

- Original approach to implement CRT-RSA taking advantage of the public exponent
  - Efficient message blinding alternative
  - Improved performances

Conclusion

- Original approach to implement CRT-RSA taking advantage of the public exponent
  - Efficient message blinding alternative
  - Improved performances
- Performances:
  - 1024 bits: 14% faster with a 19% shorter key
  - 2048 bits: 8% faster with a 20% shorter key

Conclusion

- Original approach to implement CRT-RSA taking advantage of the public exponent
  - Efficient message blinding alternative
  - Improved performances
- Performances:
  - 1024 bits: 14% faster with a 19% shorter key
  - 2048 bits: 8% faster with a 20% shorter key
- Side-Channel : most efficient message blinding published so far

Conclusion

- Original approach to implement CRT-RSA taking advantage of the public exponent
  - Efficient message blinding alternative
  - Improved performances
- Performances:
  - 1024 bits: 14% faster with a 19% shorter key
  - 2048 bits: 8% faster with a 20% shorter key
- Side-Channel : most efficient message blinding published so far
- We hope more research will be done to take advantage of the public exponent!

Thank you for
your attention!!!