An Introduction to Coppersmith’s method and Applications in Cryptology

Rina Zeitoun

r.zeitoun@oberthur.com

Oberthur Technologies

Master SFPN - UPMC

November 19th, 2015
Agenda

1. Solving Polynomials and Cryptology
2. Coppersmith’s Method
3. Some Applications
1. Solving Polynomials and Cryptology
2. Coppersmith’s Method
3. Some Applications
Algebraic Attacks

\[
\begin{align*}
    x_0 + x_1 x_2 + x_1 x_4 &= 0 \\
    x_0 x_1 x_4 + x_3 x_5 + x_2 &= 0 \\
    x_0 x_1 x_3 + x_0 x_2 + x_5 x_2 &= 0
\end{align*}
\]

key = ????????
Algebraic Attacks

\[ \text{key} = 0x8C1337DA \]

\[
\begin{align*}
x_0 + x_1 x_2 + x_1 x_4 &= 0 \\
x_0 x_1 x_4 + x_3 x_5 + x_2 &= 0 \\
x_0 x_1 x_3 + x_0 x_2 + x_5 x_2 &= 0
\end{align*}
\]

\[
\begin{align*}
x_1 &= 73218 \\
x_2 &= 984 \\
x_3 &= 0 \\
x_4 &= 37984155 \\
x_5 &= 4869
\end{align*}
\]
Algebraic Attacks

key = ????????

\[ x_0 + x_1 x_2 + x_1 x_4 = 0 \\
 x_0 x_1 x_4 + x_3 x_5 + x_2 = 0 \\
 x_0 x_1 x_3 + x_0 x_2 + x_5 x_2 = 0 \]

Solver

RSA

Factorization of \( N = p \times q \)

\[ N - x \cdot y = 0 \quad \text{Difficult} \ [1] \]

Algebraic Attacks with Additional Information

\[ \begin{align*}
    x_0 + x_1 x_2 + x_1 x_4 &= 0 \\
    x_0 x_1 x_4 + x_3 x_5 + x_2 &= 0 \\
    x_0 x_1 x_3 + x_0 x_2 + x_5 x_2 &= 0 \\
    x_2 + x_3 + 1 &= 0 \\
    x_4 x_1 + x_5 &= 0
\end{align*} \]
Algebraic Attacks with Additional Information

key = ????????

\[ x_0 + x_1 x_2 + x_1 x_4 = 0 \]
\[ x_0 x_1 x_4 + x_3 x_5 + x_2 = 0 \]
\[ x_0 x_1 x_3 + x_0 x_2 + x_5 x_2 = 0 \]

\[ x_2 + x_3 + 1 = 0 \]
\[ x_4 x_1 + x_5 = 0 \]

Additional Information

Solver

\[ N - x \cdot y = 0 \] Difficult

What if more information? [2]

RSA

Factorization of \( N = p \times q \)

Knowing Some Information

Knowing Some Information

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]

\[ m = k \]
Knowing Some Information

\[ m = k \]

\[ f(m, \ldots) = 0 \]

\[ f(aB + x, \ldots) = 0 \]

Coppersmith's method in polynomial time (1996) [1]

- Find all small integer roots of some polynomial equations.

⇒ Numerous applications in cryptology.

Knowing Some Information

\[ m \quad = \quad k \quad + \quad x \]

**Known** \quad **Unknown**

\[ f(m, \ldots) = 0 \]

\[ f(aB + x, \ldots) = 0 \]

Coppersmith’s method in polynomial time (1996) [1]

- Find all **small** integer roots of some polynomial equations.
  \[ \Rightarrow \] Numerous applications in cryptology.

How to get additional information?

Non-invasive Attacks

- Can allow to recover some bits of information
How to get additional information?

A Diode Laser Station
(picture from Riscure Inspector Data Sheet)

Invasive Attacks

- Can modify some secret data
Algebraic Attacks with Side-Channel Information

key = 0x8C1337DA

\[
x_1 = 73218 \\
x_2 = 984 \\
x_3 = 0 \\
x_4 = 37984155 \\
x_5 = 4869
\]
Outline

1. Solving Polynomials and Cryptology
2. Coppersmith’s Method
3. Some Applications
Don Coppersmith was born in 1950 in America.
Holds a doctorate in mathematics from Harvard University.
Played a major part of the design of DES within the IBM team.
One of the designers of MARS (finalist for the AES).
Famous for his works on discrete logarithms, cryptanalysis of RSA, rapid matrix multiplication.
**Coppersmith’s Theorem**

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial \( f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1x + a_0 \).
  - \( N \) an integer of unknown factorization.

- **Find:**
  - All integers \( x_0 \) such that \( f(x_0) \equiv 0 \mod N \).
Coppersmith’s Theorem

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial \( f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1x + a_0 \).
  - \( N \) an integer of unknown factorization.

- **Find:**
  - All integers \( x_0 \) such that \( f(x_0) \equiv 0 \mod N \).

Coppersmith’s Theorem for the Univariate Modular case

- The solutions \( x_0 \) can be found in polynomial time in \( \log(N) \) if
  \[
  |x_0| < N^{1/\delta}.
  \]
Coppersmith’s Method  
(Howgrave-Graham)

**The problem:** find all small integers $x_0$ s.t. $f(x_0) \equiv 0 \mod N$.

**The idea:** find a small polynomial $g$ s.t. $g(x_0) = 0$ over $\mathbb{Z}$.

### Euclidean Lattices

New **small** polynomial $g$ \[\text{LLL Reduction}\] [1].

---

Coppersmith’s Method (Howgrave-Graham)

The problem: find all small integers $x_0$ s.t. $f(x_0) \equiv 0 \mod N$.

The idea: find a small polynomial $g$ s.t. $g(x_0) = 0$ over $\mathbb{Z}$.

How to find the polynomial $g$:

- $g(x_0) \equiv 0 \mod N^m$
- $g(x_0) < N^m$

$\Rightarrow g(x_0) = 0$ over $\mathbb{Z}$. 
Step 1: Family of Polynomials

Consider the Family of Polynomials:

\[ g_{i,j}(x) = x^j N^{m-i} f^i(x) \]

- For \( 0 \leq i < m \) and \( 0 \leq j < \delta \)
- For \( i = m \) and \( j = 0 \)

Crucial Property of the Polynomials

\[ g_{i,j}(x_0) \equiv x_0^j N^{m-i} N^i \equiv x_0^j N^m \equiv 0 \mod N^m \]
Step 2: Coppersmith’s Matrix

Construction of the Matrix with $|x_0| < X$

- For $i$ from 0 to $m - 1$
  - For $j$ from 0 to $\delta - 1$
    $$M[\delta i + j] = (xX)^j N^{m-i} f^i(xX)$$
- $M[\delta m] = f^m(xX)$
Step 2: Coppersmith’s Matrix

Coppersmith’s matrix with $m = 1$

$f(x) = a_0 + a_1 x + \cdots + x^\delta$

$$B = \begin{pmatrix}
N & XN \\
\vdots & \ddots \\
x^\delta \cdot N & \cdots & N \\
\end{pmatrix}$$
Step 2: Coppersmith’s Matrix

Coppersmith’s matrix

with $m = 2$

$$f(x) = a_0 + a_1 x + \cdots + x^\delta$$

$$B = \begin{pmatrix}
N^2 \\
X N^2 \\
\vdots \\
X^\delta - 1 N^2 \\
\end{pmatrix}$$
Step 2: Coppersmith’s Matrix

$$B = \begin{pmatrix}
N^m & XN^m & & & \\
& \ddots & \ddots & & \\
& & X^{\delta-1}N^m & & \\
a_0N^{m-1} & \cdots & X^\delta N^{m-1} & \cdots & X^{\delta+1}N^{m-1} \\
a_0XN^{m-1} & \cdots & X^{\delta+1}N^{m-1} & \cdots & \cdots \\
& \ddots & \ddots & \ddots & \\
a_0X^{\delta-1}N^{m-1} & \cdots & X^{2\delta-1}N^{m-1} & \cdots & \cdots \\
a_0^{m-1}N & \cdots & \cdots & \cdots & X^{\delta(m-1)}N \\
& \ddots & \ddots & \ddots & \ddots \\
& & a_0^{m-1}X^{\delta-1}N & \cdots & X^{\delta(m-1)+1}N \\
a_0^m & \cdots & \cdots & \cdots & X^{\delta m-1}N \\
a_0^m & \cdots & \cdots & \cdots & X^{\delta m} \\
\end{pmatrix}$$
Step 3: Lattice Reduction

A lattice is a grid which is:

- regular
- infinite
Step 3: Lattice Reduction

It is defined by a basis:
Step 3: Lattice Reduction

There is not uniqueness:
Step 3: Lattice Reduction

Lattice Reduction Problem

“Given an ordinary basis, find a good basis”

A good basis is composed of vectors which are:

- relatively short
- nearly orthogonal

The Lattice Reduction problem is NP-hard.
Step 3: Lattice Reduction

A lattice is a discrete additive subgroup of $\mathbb{R}^n$

- Let vectors $b_1, b_2, \ldots, b_d \in \mathbb{R}^n$ form a basis $B$.
- A lattice $\mathcal{L}$ is: $\mathcal{L} = \{ \sum_{i=1}^{d} \alpha_i b_i \text{ with } \alpha_i \in \mathbb{Z} \}$

Some Important Properties

- **Dimension** $d$ of a lattice: number of vectors in the basis.
- **Invariant** of all bases of the same lattice: volume.
- If $\mathcal{L}$ is full rank ($d = n$), then $\text{vol}(\mathcal{L}) = |\text{det}(B)|$. 
Step 3: Lattice Reduction

**HKZ Algorithm (Hermite, Korkine, Zolotarev)**

- The HKZ algorithm gives the **best basis**
- It is **exponential** in the dimension of the lattice

**LLL Algorithm (Lenstra, Lenstra, Lovász)**

- **LLL** is a **polynomial time** reduction algorithm (1982)
- Quality of **LLL**-reduced basis: exponentially poor in $d$
- Shortest vector of **LLL**-reduced basis:
  \[
  \|v\| < 2^{(d-1)/4}(\det B)^{1/d}
  \]
- **LLL** works surprisingly well in practice
### Step 4: Finding Small Solutions

**Obtain Polynomial** \( g \) such that \( g(x_0) \equiv 0 \mod N^m \)

- First vector of the \( LLL \)-reduced basis: \( v = (v_0, v_1, \ldots, v_{d-1}) \)
- Get new polynomial \( g(x) = v_0 + \frac{v_1}{X} x + \cdots + \frac{v_{d-1}}{X^{d-1}} x^{d-1} \)

**Theorem: (Howgrave-Graham)**

- Let \( g(x_0) \equiv 0 \mod N^m \) with degree \( d - 1 \) and \( |x_0| < X \)

\[
\|g(xX)\| < \frac{N^m}{\sqrt{d}} \quad \Rightarrow \quad g(x_0) = 0 \text{ over } \mathbb{Z}
\]

**Condition for** \( g(x_0) = 0 \text{ over } \mathbb{Z} \)

\[
2^{(d-1)/4} (\det B)^{1/d} < \frac{N^m}{\sqrt{d}}
\]
In practice, for $\lceil \log_2(N) \rceil = 1024$ and $\delta = 2$

<table>
<thead>
<tr>
<th>Upper bound for $x_0$</th>
<th>$2^{492}$</th>
<th>$2^{496}$</th>
<th>$2^{500}$</th>
<th>$2^{503}$</th>
<th>$2^{504}$</th>
<th>$2^{505}$</th>
<th>...</th>
<th>$2^{512}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice Dimension $n = \delta m + 1$</td>
<td>29</td>
<td>35</td>
<td>51</td>
<td>71</td>
<td>77</td>
<td>87</td>
<td>...</td>
<td>NA</td>
</tr>
<tr>
<td>Size of elements in $B$ (bits)</td>
<td>15360</td>
<td>18432</td>
<td>26624</td>
<td>36864</td>
<td>39936</td>
<td>45056</td>
<td>...</td>
<td>NA</td>
</tr>
<tr>
<td>Time for LLL (seconds)</td>
<td>10.6</td>
<td>35.2</td>
<td>355</td>
<td>2338</td>
<td>4432</td>
<td>11426</td>
<td>...</td>
<td>NA</td>
</tr>
</tbody>
</table>

Tests performed using Magma V2.19-5.
Performing exhaustive search

- Split the variable $x$ into $\alpha$ and $x'$.
- The new variable is $x'$.
- Perform an exhaustive search on $\alpha$.

**Complexity:** $2^\alpha \times \text{(LLL-reduction complexity)}$
Exhaustive Search

\[ \alpha = 0 \quad \alpha = 1 \quad \alpha = 2 \quad \cdots \quad \alpha = 255 \]

Coppersmith Matrices

\[ \begin{pmatrix} B_0 \end{pmatrix} \quad \begin{pmatrix} B_1 \end{pmatrix} \quad \begin{pmatrix} B_2 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} B_{255} \end{pmatrix} \]

LLL-Reduced Matrices

\[ \begin{pmatrix} B_0^R \end{pmatrix} \quad \begin{pmatrix} B_1^R \end{pmatrix} \quad \begin{pmatrix} B_2^R \end{pmatrix} \quad \cdots \quad \begin{pmatrix} B_{255}^R \end{pmatrix} \]

\[ \emptyset \quad \emptyset \quad x_0' \quad \cdots \quad \emptyset \]
Timing / Solution size \(\lceil \log_2(N) \rceil = 1024\) and \(\delta = 2\)
### The Problem (Multivariate Modular Case):

- **Input:**
  - A polynomial \( f(x_1, x_2, \ldots, x_n) \)
  - \( N \) an integer of unknown factorization

- **Find:**
  - All integers \( x_1, x_2, \ldots, x_n \) such that \( f(x_1, x_2, \ldots, x_n) \equiv 0 \mod N \)

### The Idea to Find Small Solutions (Heuristic)

- Take \( n \) vectors from the \( LLL \)-reduced matrix
  - \( n \) unknowns \( \Rightarrow \) \( n \) equations over \( \mathbb{Z} \)
Other Related Results of Coppersmith

The Problem (Bivariate Integer Case):

- **Input:**
  - A polynomial $f(x, y)$
  - $N$ an integer of unknown factorization

- **Find:**
  - All integers $x_0, y_0$ such that $f(x_0, y_0) = 0$ over $\mathbb{Z}$

The Idea to Find Small Solutions

- Take 1 vector from the $LLL$-reduced matrix
  - 2 unknowns $\iff$ 2 equations over $\mathbb{Z}$
1. Solving Polynomials and Cryptology
2. Coppersmith’s Method
3. Some Applications
RSA Cryptosystem [1]

RSA Key Generation

- Generate two large primes $p$ and $q$
- Compute $N = p \times q$ and $\phi(N) = (p - 1)(q - 1)$
- Select $(e, d)$ such that $ed \equiv 1 \mod \phi(N)$

Encryption/Decryption Process

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \equiv m^e \mod N$</td>
<td>$m \equiv C^d \mod N$</td>
</tr>
</tbody>
</table>

Factorize $N = p \times q \Rightarrow$ Break RSA

RSA Attack with Small Exponent $e$

$\mathbf{m} = \begin{cases} \mathbf{k} & \text{Known} \\ \mathbf{x} & \text{Unknown} \end{cases}$

- Example 1: Stereotyped messages

$\mathbf{m} = \begin{cases} \text{Today, your password is} & \text{H!a2ch#e;m} \end{cases}$

- Example 2: Fixed pattern padding

$\mathbf{m} = \begin{cases} \text{11111111111111111111111} & \text{T:od!aRa#ba} \end{cases}$
RSA Attack with Small Exponent $e$

RSA Encryption of the message $m$

- Ciphertext $C = m^e \mod N = \left(2^t k + x\right)^e \mod N$

Application of Coppersmith’s Theorem

- Consider $P(x) = \left(2^t k + x\right)^e - C \equiv 0 \mod N$
- Coppersmith’s condition: $|x_0| < N^{1/e}$

 réserve: If $e = 3$, recover $m$ provided that 2/3 of the bits are known.
RSA Attack with Short Random Padding

• Alice sends to Bob the ciphertext $C_1 = m^e = (k + r_1)^e \mod N$

\[ m = k + r_1 \]

This is a confidential message addressed to Bob

• Eve intercepts $C_1$ so that Bob does not receive it

• Alice sends to Bob the ciphertext $C_2 = m'^e = (k + r_2)^e \mod N$

\[ m' = k + r_2 \]

This is a confidential message addressed to Bob

Random $r_1$

Random $r_2$
Eve knows the two polynomials:

\[
\begin{align*}
    p_1(k, r_1) &= (k + r_1)^e - C_1 \equiv 0 \mod N \\
    p_2(k, r_2) &= (k + r_2)^e - C_2 \equiv 0 \mod N
\end{align*}
\]
RSA Attack with Short Random Padding

Eve knows the two polynomials:

\[
\begin{align*}
    p_1(k, r_1) &= (k + r_1)^e - C_1 \equiv 0 \mod N \\
    p_2(k, r_2) &= (k + r_2)^e - C_2 \equiv 0 \mod N
\end{align*}
\]

Perform a change of variables:

\[
\begin{align*}
    x &= k + r_1 \\
    y &= r_2 - r_1
\end{align*} \implies
\begin{align*}
    p_1(x) &= (x)^e - c_1 \equiv 0 \mod N \\
    p_2(x, y) &= (x + y)^e - c_2 \equiv 0 \mod N
\end{align*}
\]
RSA Attack with Short Random Padding

Eve knows the two polynomials:

\[
\begin{align*}
 p_1(k, r_1) &= (k + r_1)^e - C_1 \equiv 0 \mod N \\
 p_2(k, r_2) &= (k + r_2)^e - C_2 \equiv 0 \mod N
\end{align*}
\]

Perform a change of variables:

\[
\begin{align*}
 x &= k + r_1 \\
 y &= r_2 - r_1
\end{align*}
\]

\[
\Rightarrow \begin{align*}
 p_1(x) &= x^e - c_1 \equiv 0 \mod N \\
 p_2(x, y) &= (x + y)^e - c_2 \equiv 0 \mod N
\end{align*}
\]

Use resultants and Coppersmith’s method

- Get a new polynomial \( p(y) \equiv 0 \mod N \) of degree \( e^2 \)
- Solution \( y_0 = r_2 - r_1 \) found if \( |y_0| < N^{1/e^2} \)
- Compute \( \gcd(p_1(x), p_2(x)) = x - m \)

\( \Rightarrow \) If \( e = 3 \), recover \( m \) provided that \( r_1 \) and \( r_2 \) are \( < N^{1/9} \).
Combined Attack on CRT-RSA
Why Public Verification Must Not Be Public?

G. Barbu, A. Battistello, G. Dabosville, C. Giraud,
G. Renault, S. Renner, R. Zeitoun

(PKC 2013)
Signature RSA-CRT / State of the art

\[ m \]

\[ m^d \mod p \]
\[ m^d \mod q \]

\[ S = m^d \mod N \]
Signature RSA-CRT / State of the art

Compute \( \gcd(\tilde{S}^e - m, N) = q \)

Signature RSA-CRT / State of the art

\[ S = m^d \mod N \]

\[ S^e \equiv m \mod N \]

- True: Return \( S \)
- False: Security Action
Our Combined Attack on CRT-RSA

\[ m \]

\[ m^d \mod p \]

\[ m^d \mod q \]

\[ \tilde{S} \]

\[ \tilde{S}^e \overset{?}{=} m \mod N \]

False

Security Action
Our Combined Attack on CRT-RSA

\[ (m + \varepsilon)^d \mod p \]

\[ m^d \mod q \]

\[ \tilde{S}^{e} = m + \varepsilon q i_q \mod N \]

where

\[ q \cdot i_q = 1 \mod p \]

False

Security Action
Our Combined Attack on CRT-RSA

\[
(m + \varepsilon)^d \mod p \quad \text{and} \quad m^d \mod q
\]

\[
\tilde{S} = m + \varepsilon q i q \mod N
\]

\[
\tilde{S}^e = m + \varepsilon q i q \mod N
\]

Side-Channel Analysis

Recover \( \varepsilon q i q \)

Factorise \( N \)

\[ q = \gcd (N, \varepsilon q i q) \]

Security Action

False
**CPA attacks: The idea**

- Observe a **large number of executions**
- Apply a **statistic treatment** to recover secret values
### CPA attacks: The idea

- Observe a **large number of executions**
- Apply a **statistic treatment** to recover secret values

### Target of a CPA

A **sensitive value** manipulated during execution of the algorithm
Side Channel Analysis: CPA attacks

CPA attacks: The idea

- Observe a **large number of executions**
- Apply a **statistic treatment** to recover secret values

Target of a CPA

A **sensitive value** manipulated during execution of the algorithm

What is a sensitive value?

Data depending on a

- **known** value (that **changes** from one execution to another)
- **secret** value (which remains **constant**)
Several Executions of RSA Signature Algorithm with Different Input Messages $m_i$ and a constant injected fault $\varepsilon$

Power measurements during manipulation of $\tilde{S}_i^e = m_i + \varepsilon q_i q$

Guesses on power measurements $\mathcal{L} \circ H(m_i, k) = \mathcal{L}(m_i + k)$

Statistical treatment
*Compute the Pearson’s correlation coefficient*

The right guess $k$ is retrieved
Several Executions of RSA Signature Algorithm with Different Input Messages $m_i$ and a constant injected fault $\varepsilon$

Power measurements during manipulation of $\tilde{S}_i^e = m_i + \varepsilon q_i q$

Stats on power measurements $\mathcal{L} \circ H(m_i, k) = \mathcal{L}(m_i + k)$

Statistical treatment

Compute the Pearson’s correlation coefficient

The right guess $k$ is retrieved
Convergence of the correlation for the $2^8$ possible values $k_i$ for the secret (the correct one being depicted in black) depending on the number of side-channel measurements ($\times 500$).
Improvement using Coppersmith’s method

1. Apply Combined Attack
2. Recover part of the bits of $\varepsilon q_i q$
3. Other part: small solution of bivariate equation $P(x, \varepsilon) \equiv 0 \mod N$
4. Apply Coppersmith’s method
5. **Heuristic:** Retrieve other part of $\varepsilon q_i q$ if $\log(x_0) + \log(\varepsilon) < \log(N)/2$
6. Factorise $N$
   \[ q = \gcd(N, \varepsilon q_i q) \]
### Conclusion

#### Coppersmith’s Method

- Find small solutions to polynomial equations which are:
  - univariate modular
  - multivariate modular (heuristic)
  - bivariate over the integers
  - multivariate over the integers (heuristic)
- Based on lattice techniques
Coppersmith’s Method

- Find small solutions to polynomial equations which are:
  - univariate modular
  - multivariate modular (heuristic)
  - bivariate over the integers
  - multivariate over the integers (heuristic)
- Based on lattice techniques

Applications of Coppersmith’s Method

- Many applications in cryptology
- Useful to recover a secret when:
  - the secret is small
  - part of the secret is known
  - part of the secret is redundant