Power Attack on Small RSA Public Exponent from Partial Information

Pierre-alain Fouque $^{1}$ Sébastien Kunz-Jacques$^{1,2}$
Gwenaëlle Martinet$^{2}$ Frédéric Muller$^{3}$ Frédéric Valette$^{4}$

$^{1}$École normale supérieure,

$^{2}$DCSSI Crypto Lab,

$^{3}$HSBC,

$^{4}$CELAR.

October 13, 2006
Outline

1 Introduction
   - RSA with small public exponent
   - Side-channel attacks
   - Windowing algorithms
   - Exponent Randomization

2 Description of the Attack
   - Hypotheses and Mathematical Background
   - Overview of the Attack
   - Recovering the \( \lambda_i \)
   - Recovering \( \varphi(N) \) and \( d \)
   - Success Condition

3 Extensions
   - Public Exponent \( e = 2^{16} + 1 \)
   - Other Randomizations
   - Practical Results
Outline

1 Introduction
   - RSA with small public exponent
   - Side-channel attacks
   - Windowing algorithms
   - Exponent Randomization

2 Description of the Attack
   - Hypotheses and Mathematical Background
   - Overview of the Attack
   - Recovering the $\lambda_i$
   - Recovering $\phi(N)$ and $d$
   - Success Condition

3 Extensions
   - Public Exponent $e = 2^{16} + 1$
   - Other Randomizations
   - Practical Results
1 Introduction
- RSA with small public exponent
- Side-channel attacks
- Windowing algorithms
- Exponent Randomization

2 Description of the Attack
- Hypotheses and Mathematical Background
- Overview of the Attack
- Recovering the $\lambda_i$
- Recovering $\varphi(N)$ and $d$
- Success Condition

3 Extensions
- Public Exponent $e = 2^{16} + 1$
- Other Randomizations
- Practical Results
RSA with small public exponent

Notations:
- the modulus \( N = p \times q \) of size \( n \)
- the public exponent \( e \)
- the private exponent \( d \) is the inverse of \( e \) modulo \( \varphi(N) \)

Public exponent in RSA is usually small: \( e = 3 \) or \( 2^{16} + 1 \)

Advantage: speed up the signature verification or encryption

Known attacks on RSA with small public exponent:
- knowledge of consecutive bits of the private exponent leads to the entire key (needs one quarter)
- non-consecutive bits: no attack
leakage in classical implementations

- Partial information on the private exponent is often revealed by power consumption or electromagnetic radiations
- Mainly two cases:
  - Bias on the value of specific bits (i.e., $d_i = 1$ with probability $\frac{1}{2} + \epsilon$)
  - Known positions for specific bit patterns (e.g., 00)
- Mainly due to poor SPA countermeasures
Side-channel leakage in optimized windowing algorithms

- **Fixed-size window:**
  - $M^a$ is precomputed for $0 \leq a \leq 2^b - 1$
  - The exponent $d$ is processed $b$ bits at a time
  - If a $b$-bit window of $d$ is 0, no multiplication occurs $\Rightarrow$ SPA leakage
  - Usually, we cannot distinguish operand of the multiplications $\Rightarrow$ partial leakage

- **Variable-size window:**
  - As before, but maximal identically 0 windows are used to further speed up exponentiation
  - After a zero window, we know a window begins by 1
The Exponent Randomization Algorithm

- Common technique to protect against power attacks is to randomize
  - the message
  - the secret exponent
  - the modulus...

- The attacked algorithm is the following
  - Inputs: a message $M$, an exponent $d$, a modulus $N$ and $\varphi(N)$
  - Output: $M^d \mod N$

1. Pick at random $\lambda \in \{0, \ldots, 2^\ell - 1\}$
2. Compute $d' = d + \lambda \cdot \varphi(N)$
3. Return exponentiation $M^{d'} \mod N$

- for performance reasons, $\ell$ is small: typically 20 or 32
Hypotheses and Mathematical Background

Hypotheses:

- public exponent $e = 3$
- private exponent $d_i$ is randomized: $d_i = d + \lambda_i \cdot \varphi(N)$
- power analysis of a single curve reveals $1/r$ bits of $d_i$

Free information:

- about $n/2$ MSB of $\varphi(N)$ are known and equal to the $n/2$ MSB of the modulus $N$
- $d = (1 + k\varphi(N))/e$ with $k < e$
- for $e = 3$, $k = 2$: upper half of $d$ equals upper half of $\bar{d} = 2N/3$
Overview of the Attack

1. Perform SCA and store each partially known $d_i$.
2. Find the unknown value $\lambda_i$ associated to each $d_i$ using $d_i \approx \bar{d} + \lambda_iN$ and the most significant known bits of $d_i$.
3. Find recursively the least significant slices of $\varphi(N)$ and $d$ using the least significant known bits of $d_i$. 

P.-A. Fouque et al. Power Attack on Small RSA Public Exponent
Recovering the $\lambda_i$

**Inputs:** a partially known exponent $d_i$

**Outputs:** $\lambda_i$ s.t. $d_i = d + \lambda_i \times \varphi(N)$

```
for j = 0 to $2^\ell$ do
    if $[d_i]_{n/2+\ell,n+\ell} \equiv [\bar{d} + j \times N]_{n/2+\ell,n+\ell}$ then
        $\lambda_i \leftarrow j$; break
    end if
end for
```

return $\lambda_i$
Recovering $\varphi(N)$ and $d$

- work recursively with a 8-bit window (for example)
- Inputs:
  - $\{(d_i, \lambda_i)\}_{1 \leq i \leq \omega}$
  - a candidate $\phi$ for the 8s LSBs of $\varphi(N)$, assumed to be correct $\pmod{2^{8(s-1)}}$
- Output: a boolean value telling whether $\phi$ is correct

Idea (first 8 bits)
- From $\phi$, deduce the 8 LSBs of $d$
- for each $i$, using $\lambda_i$, compute the 8 LSBs of $d_i = d + \lambda_i\phi$, and check matching with corresponding curve
Recovering $\varphi(N)$ and $d$

Inputs: $\{(d_i, \lambda_i)\}, \phi$

Outputs: boolean $b$

\[
D \leftarrow \frac{1+2\phi}{3} \text{ mod } 2^{8s}
\]

ok $\leftarrow$ True

for $i = 1$ to $\omega$ do

if $\neg \{ [d_i]_{0,8s-1} \equiv D + \lambda_i \times \phi \text{ mod } 2^{8s} \}$ then

ok $\leftarrow$ False

end if

end for

return ok
Success Condition

- Ability to guess a unique value for $\lambda_i$
  - there are $\frac{n}{2r}$ known bits of $d_i$
  - if $\frac{n}{2r} > \ell$ one $\lambda_i$ will be associated to each $d_i$
- Ability to guess a unique value of $\varphi(N) \mod 2^{8k}$
  - there are $\frac{8}{r}$ known bits on a 8-bit window of some $d_i$
  - as long as $\omega \leq 2^8$: experiences for $\neq d_i \approx$ independent
  - if $\frac{8\omega}{r} \gg 8$ only one candidate is maintained with high probability
e = 2^{16} + 1

- We have $\bar{d} = \left\lfloor \frac{1+kN}{e} \right\rfloor + \lambda N$
- $0 < k < e$
- For $e = 3$, we knew that $k = 2$
- If $e = 2^{16} + 1$, first step: retrieve $\{\lambda_i\}$ and $k$
- Once $k$ is known, the previous attack applies
- Finding $k$:
  - simultaneous exhaustive search on $k$ and $\lambda_1$
  - can be optimized (see paper)
Other Randomizations

- The attack still works if
  - the message is randomized
  - the modulus is randomized during the computation
  - the bits of the private exponent are known only with some probability
## Practical Results

<table>
<thead>
<tr>
<th>Modulus size</th>
<th>value of $e$</th>
<th>size of random</th>
<th>ratio of partial information known</th>
<th>attack success</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>3</td>
<td>20</td>
<td>$1/16$</td>
<td>no</td>
</tr>
<tr>
<td>1024</td>
<td>3</td>
<td>20</td>
<td>$1/16$</td>
<td>yes</td>
</tr>
<tr>
<td>1024</td>
<td>$2^{16} + 1$</td>
<td>20</td>
<td>$1/16$</td>
<td>yes</td>
</tr>
<tr>
<td>2048</td>
<td>3</td>
<td>32</td>
<td>$1/32$</td>
<td>yes</td>
</tr>
<tr>
<td>2048</td>
<td>$2^{16} + 1$</td>
<td>32</td>
<td>$1/32$</td>
<td>yes</td>
</tr>
</tbody>
</table>
Conclusion

- Unfortunate interaction of DPA countermeasure and partial SPA leakage
- The anti-DPA randomization also randomizes leakages...
- ...allowing to retrieve the full private exponent.