(In)security Against Fault Injection Attacks on CRT-RSA Implementations

Alexandre Berzati, Cécile Canovas and Louis Goubin

E-mail: alexandre.berzati@cea.fr
Outline

1 Introduction
   ■ Previous work
   ■ Overview of our attack

2 Attack principle
   ■ Ciet & Joye Countermeasure
   ■ Fault Model
   ■ Faulty Execution
   ■ Fault Analysis

3 Conclusion
Introduction

Description

Fault analysis on a protected CRT-RSA implementation
Introduction

Description
Fault analysis on a protected CRT-RSA implementation

Motivation
Highlighting that protecting CRT-RSA against DFA is a challenging problem
Outline

1 Introduction
   - Previous work
     - Overview of our attack

2 Attack principle
   - Ciet & Joye Countermeasure
   - Fault Model
   - Faulty Execution
   - Fault Analysis

3 Conclusion

(In)security Against Fault Injection Attacks on CRT-RSA Implementations - Alexandre Berzati
Previous work

- DFA on CRT-RSA
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97

- *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
  - *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
  - *Cryptanalysis of a Provably Secure CRT-RSA Algorithm* (Wag04), ACM-CCS 2004
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
  - *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
  - *Cryptanalysis of a Provably Secure CRT-RSA Algorithm* (Wag04), ACM-CCS 2004

- **Methods for protecting CRT-RSA**
  - Shamir’s trick: *Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks* (Sha97), Rump Session of Eurocrypt’97
  - Infective Computation: *RSA Speedup with Residue Number System Immune Against Hardware Fault Cryptanalysis* (YKLM01), ISISC 2001
  - BOS Scheme: *A New CRT-RSA Algorithm Secure Against Bellcore Attack* (BOS03), ACM-CCS 2003
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
  - *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
  - *Cryptanalysis of a Provably Secure CRT-RSA Algorithm* (Wag04), ACM-CCS 2004

- **Methods for protecting CRT-RSA**
  - **Shamir’s trick**: *Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks* (Sha97), Rump Session of Eurocrypt’97
Previous work

■ DFA on CRT-RSA
  • *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
  
  • *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
  
  • *Cryptanalysis of a Provably Secure CRT-RSA Algorithm* (Wag04), ACM-CCS 2004

■ Methods for protecting CRT-RSA
  • **Shamir’s trick**: *Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks* (Sha97), Rump Session of Eurocrypt’97
  
  • **Infective Computation**: *RSA Speedup with Residue Number System Immune Against Hardware Fault Cryptanalysis* (YKLM01), ISISC 2001
Previous work

- **DFA on CRT-RSA**
  - *On the Importance of Checking Cryptographic Protocols for Faults* (BDL97), EUROCRYPT’97
  - *Fault Attack on RSA with CRT: Concrete Results and Practical Countermeasures* (ABF+02), CHES 2002
  - *Cryptanalysis of a Provably Secure CRT-RSA Algorithm* (Wag04), ACM-CCS 2004

- **Methods for protecting CRT-RSA**
  - **Shamir’s trick**: *Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks* (Sha97), Rump Session of Eurocrypt’97
  - **Infective Computation**: *RSA Speedup with Residue Number System Immune Against Hardware Fault Cryptanalysis* (YKLM01), ISISC 2001
  - **BOS Scheme**: *A New CRT-RSA Algorithm Secure Against Bellcore Attack* (BOS03), ACM-CCS 2003
Outline

1 Introduction
   ■ Previous work
   ■ Overview of our attack

2 Attack principle
   ■ Ciet & Joye Countermeasure
   ■ Fault Model
   ■ Faulty Execution
   ■ Fault Analysis

3 Conclusion
Overview of our attack

- Our attack applies on a protected CRT-RSA implementation.
Overview of our attack

- Our attack applies on a protected CRT-RSA implementation

- Provides a full secret key recovery by factorizing the public modulus $N$
Overview of our attack

- Our attack applies on a protected CRT-RSA implementation

- Provides a full secret key recovery by factorizing the public modulus $N$

- Can be applied on CRT-RSA functions that handles the secret key $d$:
  - Signature (with deterministic padding)
  - Decryption
Overview of our attack

- Our attack applies on a protected CRT-RSA implementation

- Provides a full secret key recovery by factorizing the public modulus $N$

- Can be applied on CRT-RSA functions that handles the secret key $d$:
  - Signature (with deterministic padding)
  - Decryption

- Based on a simple and practicable fault model
Outline

1 Introduction
   - Previous work
   - Overview of our attack

2 Attack principle
   - Ciet & Joye Countermeasure
   - Fault Model
   - Faulty Execution
   - Fault Analysis

3 Conclusion
**CRT-RSA Countermeasure**

Ciet & Joye Algorithm — *Practical Fault Countermeasures for Chinese Remaindering Based RSA (JC05), FDTC 2005*

**Input:** \( \hat{m}, \{p, q, d_p, d_q\} \)

**Output:** \( S = \hat{m}^d \mod N \)

**Parameters:** \( \kappa, l \)

1. For two \( \kappa \)-bit random integers \( r_1 \) and \( r_2 \)
   - (a) \( p^* = r_1 \cdot p \)
   - (b) \( q^* = r_2 \cdot q \)
   - (c) \( i_{q^*} = (q^*)^{-1} \mod p^* \)
   - (d) \( N = p \cdot q \).

2. Compute
   - (a) \( S_{p^*} \equiv \hat{m}^d p \mod p^* \) and \( s_2 \equiv \hat{m}^d q \mod \varphi(r_2) \mod r_2 \),
   - (b) \( S_{q^*} \equiv \hat{m}^d q \mod q^* \) and \( s_1 \equiv \hat{m}^d p \mod \varphi(r_1) \mod r_1 \).

3. Compute \( S^* \equiv S_{q^*} + q^* \cdot i_{q^*} \cdot (S_{p^*} - S_{q^*}) \mod p^* \)

4. Compute
   - (a) \( c_1 \equiv (S^* - s_1 + 1) \mod r_1 \)
   - (b) \( c_2 \equiv (S^* - s_2 + 1) \mod r_2 \)

5. For a \( l \)-bit integer \( r_3 \), set \( \gamma = \left\lfloor \frac{r_3 \cdot c_1 + (2^l - r_3) \cdot c_2}{2^l} \right\rfloor \)

6. Return \( S \equiv (S^*)^\gamma \mod N \)
Outline

1. Introduction
   - Previous work
   - Overview of our attack

2. Attack principle
   - Ciet & Joye Countermeasure
   - Fault Model
   - Faulty Execution
   - Fault Analysis

3. Conclusion
Fault model

- Perturbation of the CRT-RSA signature
  - Transient byte fault on $S_p^*$
Fault model

- Perturbation of the CRT-RSA signature
  - Transient byte fault on $S_p^*$

- The faulty result $\hat{S}_p^*$ can be model as:

$$\hat{S}_p^* = S_p^* \oplus \epsilon$$

where $\epsilon = R_8 \cdot 2^{8i}$, $R_8$ is a random byte value and $i \in [0; \frac{(n/2)+\kappa}{8} - 1]$
Fault model

- Perturbation of the CRT-RSA signature
  - Transient byte fault on $S_p^*$

- The faulty result $\hat{S}_p^*$ can be model as:

$$\hat{S}_p^* = S_p^* \oplus \varepsilon$$

where $\varepsilon = R_8 \cdot 2^{8i}$, $R_8$ is a random byte value and $i \in [0; \left(\frac{n}{2}\right)+\kappa - 1]$

- Then, the fault spreads over the computation:
  - During the CRT Recombination
  - Computation of the check values and gamma
  - Final signature
Outline

1 Introduction
   - Previous work
   - Overview of our attack

2 Attack principle
   - Ciet & Joye Countermeasure
   - Fault Model
   - Faulty Execution
   - Fault Analysis

3 Conclusion
Faulty Execution

Ciet & Joye Algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>(\dot{m}, {p, q, d_p, d_q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>(S = \dot{m}^d \mod N)</td>
</tr>
<tr>
<td>Parameters:</td>
<td>(\kappa, l)</td>
</tr>
</tbody>
</table>

1. For two \(\kappa\)-bit random integers \(r_1\) and \(r_2\)
   (a) \(p^* = r_1 \cdot p\),
   (b) \(q^* = r_2 \cdot q\),
   (c) \(i_{q^*} = (q^*)^{-1} \mod p^*\),
   (d) \(N = p \cdot q\).
2. Compute
   (a) \(S_{p^*} \equiv \dot{m}^{d_p} \mod p^*\) and \(s_2 \equiv \dot{m}^{d_q} \mod \varphi(r_2) \mod r_2\),
   (b) \(S_{q^*} \equiv \dot{m}^{d_q} \mod q^*\) and \(s_1 \equiv \dot{m}^{d_p} \mod \varphi(r_1) \mod r_1\).
3. Compute \(S^* \equiv S_{q^*} + q^* \cdot i_{q^*} \cdot (S_{p^*} - S_{q^*}) \mod p^*\)
4. Compute
   (a) \(c_1 \equiv (S^* - s_1 + 1) \mod r_1\)
   (b) \(c_2 \equiv (S^* - s_2 + 1) \mod r_2\)
5. For a \(l\)-bit integer \(r_3\), set \(\gamma = \left\lfloor \frac{(r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)}{2^l} \right\rfloor\)
6. Return \(S \equiv (S^*)^\gamma \mod N\)
Faulty Execution

Ciet & Joye Algorithm

Input: $\dot{m}, \{p, q, d_p, d_q\}$
Output: $S = \dot{m}^d \mod N$
Parameters: $\kappa, l$

1. For two $\kappa$-bit random integers $r_1$ and $r_2$
   (a) $p^* = r_1 \cdot p$,
   (b) $q^* = r_2 \cdot q$,
   (c) $i_{q^*} = (q^*)^{-1} \mod p^*$,
   (d) $N = p \cdot q$.

2. Compute
   (a) $S_{p^*} \equiv \dot{m}^{dp} \mod p^*$ and $s_2 \equiv \dot{m}^{d_q} \mod \varphi(r_2) \mod r_2$,
   (b) $S_{q^*} \equiv \dot{m}^{dq} \mod q^*$ and $s_1 \equiv \dot{m}^{dp} \mod \varphi(r_1) \mod r_1$.

3. Compute $S^* \equiv S_{q^*} + q^* \cdot i_{q^*} \cdot (S_{p^*} - S_{q^*}) \mod p^*$

4. Compute
   (a) $c_1 \equiv (S^* - s_1 + 1) \mod r_1$
   (b) $c_2 \equiv (S^* - s_2 + 1) \mod r_2$

5. For a $l$-bit integer $r_3$, set $\gamma = \left\lfloor \frac{(r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)}{2^l} \right\rfloor$

6. Return $S \equiv (S^*)^\gamma \mod N$
Faulty Execution

Ciet & Joye Algorithm

Input: \( \dot{m}, \{p, q, d_p, d_q\} \)
Output: \( S = \dot{m}^d \mod N \)
Parameters: \( \kappa, l \)

1. For two \( \kappa \)-bit random integers \( r_1 \) and \( r_2 \)
   - (a) \( p^* = r_1 \cdot p \)
   - (b) \( q^* = r_2 \cdot q \)
   - (c) \( i_{q^*} = (q^*)^{-1} \mod p^* \)
   - (d) \( N = p \cdot q \)

2. Compute
   - (a) \( \hat{S}^p_{\kappa} \equiv \dot{m}^{d_p} \mod p^* \) and \( s_2 \equiv \dot{m}^{d_q} \mod \varphi(r_2) \mod r_2 \)
   - (b) \( \hat{S}^q_{\kappa} \equiv \dot{m}^{d_q} \mod q^* \) and \( s_1 \equiv \dot{m}^{d_p} \mod \varphi(r_1) \mod r_1 \)

3. Compute \( \hat{S}^*_{\kappa} \equiv S_{\kappa}^* + q^* \cdot i_{q^*} \cdot (\hat{S}^p_{\kappa} - S_{\kappa}^q) \mod p^* \)

4. Compute
   - (a) \( c_1 \equiv (S^* - s_1 + 1) \mod r_1 \)
   - (b) \( c_2 \equiv (S^* - s_2 + 1) \mod r_2 \)

5. For a \( l \)-bit integer \( r_3 \), set \( \gamma = \left\lfloor \frac{r_3 \cdot c_1 + (2^l - r_3) \cdot c_2}{2^l} \right\rfloor \)

6. Return \( S \equiv (S^*)^\gamma \mod N \)
Ciet & Joye Algorithm

Input: \( \dot{m}, \{p, q, d_p, d_q\} \)
Output: \( S = \dot{m}^d \mod N \)
Parameters: \( \kappa, l \)

1. For two \( \kappa \)-bit random integers \( r_1 \) and \( r_2 \)
   (a) \( p^* = r_1 \cdot p \),
   (b) \( q^* = r_2 \cdot q \),
   (c) \( i_{q^*} = (q^*)^{-1} \mod p^* \),
   (d) \( N = p \cdot q \).
2. Compute
   (a) \( \hat{S}_{p^*} \equiv \dot{m}^{d_p} \mod p^* \) and \( s_2 \equiv \dot{m}^{d_q} \mod \varphi(r_2) \mod r_2 \),
   (b) \( S_{q^*} \equiv \dot{m}^{d_q} \mod q^* \) and \( s_1 \equiv \dot{m}^{d_p} \mod \varphi(r_1) \mod r_1 \).
3. Compute \( \hat{S}^* \equiv S_{q^*} + q^* \cdot i_{q^*} \cdot (\hat{S}_{p^*} - S_{q^*}) \mod p^* \).
4. Compute
   (a) \( \hat{c}_1 \equiv (\hat{S}^* - s_1 + 1) \mod r_1 \)
   (b) \( \hat{c}_2 \equiv (\hat{S}^* - s_2 + 1) \mod r_2 \).
5. For a \( l \)-bit integer \( r_3 \), set \( \gamma = \left\lfloor \frac{(r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)}{2^l} \right\rfloor \).
6. Return \( S \equiv (S^*)^\gamma \mod N \).
## Faulty Execution

### Ciet & Joye Algorithm

**Input:** \( \hat{m}, \{p, q, d_p, d_q\} \)

**Output:** \( S = \hat{m}^d \mod N \)

**Parameters:** \( \kappa, l \)

1. For two \( \kappa \)-bit random integers \( r_1 \) and \( r_2 \)
   - (a) \( p^* = r_1 \cdot p \)
   - (b) \( q^* = r_2 \cdot q \)
   - (c) \( i_{q^*} = (q^*)^{-1} \mod p^* \)
   - (d) \( N = p \cdot q \)

2. Compute
   - (a) \( \hat{S}_p^* \equiv \hat{m}^{dp} \mod p^* \) and \( s_2 \equiv \hat{m}^{dq} \mod \varphi(r_2) \mod r_2 \)
   - (b) \( \hat{S}_q^* \equiv \hat{m}^{dq} \mod q^* \) and \( s_1 \equiv \hat{m}^{dp} \mod \varphi(r_1) \mod r_1 \)

3. Compute \( \hat{S}^* \equiv \hat{S}_q^* + q^* \cdot i_{q^*} \cdot (\hat{S}_p^* - \hat{S}_q^*) \mod p^* \)

4. Compute
   - (a) \( \hat{c}_1 \equiv (\hat{S}^* - s_1 + 1) \mod r_1 \)
   - (b) \( \hat{c}_2 \equiv (\hat{S}^* - s_2 + 1) \mod r_2 \)

5. For a \( l \)-bit integer \( r_3 \), set \( \hat{\gamma} = \left\lfloor \frac{(r_3 \cdot \hat{c}_1 + (2^l - r_3) \cdot \hat{c}_2)}{2^l} \right\rfloor \)

6. Return \( S \equiv (S^*)^\gamma \mod N \)
Faulty Execution

Ciet & Joye Algorithm

| Input:  | \( \dot{m}, \{p, q, d_p, d_q\} \) |
| Output: | \( S = \dot{m}^d \mod N \) |
| Parameters: | \( \kappa, l \) |

1. For two \( \kappa \)-bit random integers \( r_1 \) and \( r_2 \)
   (a) \( p^* = r_1 \cdot p \),
   (b) \( q^* = r_2 \cdot q \),
   (c) \( i_{q^*} = (q^*)^{-1} \mod p^* \),
   (d) \( N = p \cdot q \).
2. Compute
   (a) \( S_{p^*} = \dot{m}^{d_p} \mod p^* \) and \( s_2 = \dot{m}^{d_q} \mod \varphi(r_2) \mod r_2 \),
   (b) \( S_{q^*} = \dot{m}^{d_q} \mod q^* \) and \( s_1 = \dot{m}^{d_p} \mod \varphi(r_1) \mod r_1 \).
3. Compute \( \hat{S}^* = S_{q^*} + q^* \cdot i_{q^*} \cdot (S_{p^*} - S_{q^*}) \mod p^* \)
4. Compute
   (a) \( \hat{c}_1 = (\hat{S}^* - s_1 + 1) \mod r_1 \)
   (b) \( \hat{c}_2 = (\hat{S}^* - s_2 + 1) \mod r_2 \)
5. For a \( l \)-bit integer \( r_3 \), set \( \hat{\gamma} = \left[ \frac{(r_3 \cdot \hat{c}_1 + (2^l - r_3) \cdot \hat{c}_2)}{2^l} \right] \)
6. Return \( \hat{S} = (\hat{S}^*)^{\hat{\gamma}} \mod N \)
Outline

1. Introduction
   - Previous work
   - Overview of our attack

2. Attack principle
   - Ciet & Joye Countermeasure
   - Fault Model
   - Faulty Execution
   - Fault Analysis

3. Conclusion
Fault Analysis

Consequences of the fault

The faulty result $S_p^*$ has been modeled as:

$$S_p^* = S_p^* \oplus R_B \cdot 2^{8i}$$
Consequences of the fault

The faulty result $\hat{S}_p^*$ has been modeled as:

$$\hat{S}_p^* = S_p^* \oplus R_8 \cdot 2^{8i}$$

Then, the fault infects the check values:

$$\hat{c}_1 \equiv (\hat{S}_p^* - s_1 + 1) \mod r_1$$
$$\equiv 1 + R_8 \cdot 2^{8i} \mod r_1$$
$$\approx 1 + R_8 \cdot 2^{8i}$$

$$\hat{c}_2 \equiv (\hat{S}_p^* - s_2 + 1) \mod r_2$$
$$\equiv 1 \mod r_2$$
Fault Analysis

Consequences of the fault

The faulty result $S_p^*$ has been modeled as:

$$S_p^* = S_p^* \oplus R_8 \cdot 2^{8i}$$

Then, the fault infects the check values:

$$\hat{c}_1 \equiv (\hat{S}^* - s_1 + 1) \mod r_1$$
$$\equiv 1 + R_8 \cdot 2^{8i} \mod r_1$$
$$\approx 1 + R_8 \cdot 2^{8i}$$

$$c_2 \equiv (\hat{S}_p^* - s_2 + 1) \mod r_2$$
$$\equiv 1 \mod r_2$$

So, the erroneous exponent $\hat{\gamma}$ can be written as:

$$\hat{\gamma} = \left\lfloor \frac{(r_3 \cdot \hat{c}_1 + (2^l - r_3) \cdot c_2)}{2^l} \right\rfloor$$
$$= \left\lfloor \frac{R_8 \cdot r_3 \cdot 2^{8i}}{2^l} \right\rfloor + 1$$
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

\[
\begin{array}{ccccccc}
0 & 0 & \cdots & 0 & R_8 \cdot r_3 & \cdots & R_8 \cdot r_3 & 0 & \cdots & 0 & 0 \\
\kappa + \ell & \ell + 8i + 8 & 8i & 0
\end{array}
\]
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:

\[
\begin{array}{c|c|c|c|c}
\kappa + l & l + 8i + 8 & 8i & 0 \\
\hline
0 & 0 & \ldots & 0 & R_8 r_3 & \ldots & R_8 r_3 & 0 & \ldots & 0 & 0 \\
\end{array}
\]
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:

- Result of the right shift by $l$ bits if $l < 8i$ and $l < \kappa$:
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:

- Result of the right shift by $l$ bits if $l < 8i$ and $l < \kappa$:
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:

- Result of the right shift by $l$ bits if $l < 8i$ and $l < \kappa$:
Fault Analysis

- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$

- Result of the right shift by $l$ bits if $l > 8i$:

- Result of the right shift by $l$ bits if $l < 8i$ and $l < \kappa$:

$\Rightarrow \hat{\gamma}$ is a random value located on LSB or MSB.
Fault Analysis

First, one can advantageously notice that:

\[ \hat{S}^e \mod N = \hat{m}d \cdot e \cdot \hat{\gamma} \mod N \]

\[ = \hat{m} \cdot \hat{\gamma} \mod N \]
Fault Analysis

- First, one can advantageously notice that:

\[ \hat{S}^e \mod N = \hat{m}^{d \cdot e \cdot \hat{\gamma}} \mod N = \hat{m}^{\hat{\gamma}} \mod N \]

- Then, the attacker tries to find \( \hat{\gamma} \)'s value to factorize the public modulus \( N \)
Fault Analysis

First, one can advantageously notice that:

\[ \hat{S}^e \mod N = \hat{m}^{d \cdot e \cdot \gamma} \mod N = \hat{m}^{\gamma} \mod N \]

Then, the attacker tries to find \( \gamma \)'s value to factorize the public modulus \( N \)

**Attack algorithm**

1. The attacker chooses a candidate value for \( \gamma \)
2. The attacker computes:

\[ q' = \gcd( (\hat{S}^e - \hat{m}^{\gamma}) \mod N, N ) \]

3. Hence,
   
   (a) if \( q' = 1 \), then the attacker tries again for another candidate,
   
   (b) \( q' \neq 1 \), then \( q' \) is a prime factor of \( N \).
Performance

- Success probability for a fault that suits the model

\[
\Pr(\text{success}) = \Pr \left[ C_1 \approx 1 + R_8 \cdot 2^{8i} \ & \ \hat{\gamma} \text{ is recoverable by brute force} \right]
\]

\[
= \Pr \left[ 1 + R_8 \cdot 2^{8i} < r_1 \ & \ \text{length}(\hat{\gamma}) < B_f \right]
\]
Performance

- Success probability for a fault that suits the model

\[ \text{Pr}(\text{success}) = \text{Pr}\left[ \hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \& \hat{\gamma} \text{ is recoverable by brute force} \right] \]

\[ = \text{Pr}\left[ 1 + R_8 \cdot 2^{8i} < r_1 \& \text{length}(\hat{\gamma}) < B_f \right] \]

- For \( n = 1024 \) bits, \( \kappa = l = 80 \) bits and \( B_f = 40 \) bits

\[
\Pr(\text{success}) \approx 5.4\% \text{ for a suitable fault}
\]

The success probability increases by lengthening the brute force search.

(For 83 suitable faults, the success rate is bigger than 99%)
Success probability for a fault that suits the model

\[
\Pr(\text{success}) = \Pr \left[ \hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \ \& \ \hat{\gamma} \text{ is recoverable by brute force} \right]
\]

\[
= \Pr \left[ 1 + R_8 \cdot 2^{8i} < r_1 \ \& \ \text{length}(\hat{\gamma}) < B_f \right]
\]

For \( n = 1024 \) bits, \( \kappa = l = 80 \) bits and \( B_f = 40 \) bits

- \( \Pr(\text{success}) \approx 5.4\% \) for a suitable fault
Performance

- Success probability for a fault that suits the model

\[
\Pr(\text{success}) = \Pr [\hat{C}_1 \approx 1 + R_8 \cdot 2^{8i} \& \hat{\gamma} \text{ is recoverable by brute force}]
\]

\[= \Pr [1 + R_8 \cdot 2^{8i} < r_1 \& \text{length}(\hat{\gamma}) < B_f] \]

- For \(n = 1024\) bits, \(\kappa = l = 80\) bits and \(B_f = 40\) bits
  - \(\Pr(\text{success}) \approx 5.4\%\) for a suitable fault
  - The success probability increases by lengthening the brute force search
Performance

- Success probability for a fault that suits the model

\[
\Pr(\text{success}) = \Pr \left[ \hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \land \hat{\gamma} \text{ is recoverable by brute force} \right]
\]

\[
= \Pr \left[ 1 + R_8 \cdot 2^{8i} < r_1 \land \text{length}(\hat{\gamma}) < B_f \right]
\]

- For \( n = 1024 \) bits, \( \kappa = l = 80 \) bits and \( B_f = 40 \) bits
  - \( \Pr(\text{success}) \approx 5.4\% \) for a suitable fault
  - The success probability increases by lengthening the brute force search

- For 83 suitable faults, the success rate is bigger than 99%
Conclusion

- The proposed fault model can be extended to a less restrictive one.
Conclusion

- The proposed fault model can be extended to a less restrictive one

- The attack has been extensively simulated using GMP Library
Conclusion

- The proposed fault model can be extended to a less restrictive one

- The attack has been extensively simulated using GMP Library

- This attack works against a protected CRT-RSA implementation...
Conclusion

- The proposed fault model can be extended to a less restrictive one

- The attack has been extensively simulated using GMP Library

- This attack works against a protected CRT-RSA implementation...

- ...but it can be avoided by
  - Forcing the modular reduction during $\hat{c}_1$ computation
Conclusion

- The proposed fault model can be extended to a less restrictive one.

- The attack has been extensively simulated using GMP Library.

- This attack works against a protected CRT-RSA implementation . . .

- . . . but it can be avoided by
  - Forcing the modular reduction during $\hat{c}_1$ computation
  - Replacing the final step by the proposed variant and returning

$$S = (\gamma \cdot S^* \oplus (\gamma - 1) \cdot r)$$

*Practical Fault Countermeasures for Chinese Remaindering Based RSA (JC05)*

FDTC 2005
Conclusion

- Thank you for your attention!