micro et nanoélectronique
microsystèmes
intelligence ambiante
chaîne de l'image
biologie et santé

2008

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

Alexandre Berzati, Cécile Canovas and Louis Goubin

E-mail : alexandre.berzati@cea.fr

UNIVERSITE DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

leti

MINATEC

INSTITUT CARNOT CEA LETI

cea

Outline

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

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

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

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

CRT-RSA Countermeasure

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

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

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

Outline

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:

$$\hat{S_{p*}} = S_{p*} \oplus \varepsilon$$

where $\varepsilon = R_8 \cdot 2^{8i}$, 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 $i \in [\![0; \frac{(n/2)+\kappa}{8} - 1]\!]$

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

Outline

Faulty Execution

Ciet & Joye Algorithm

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

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

Faulty Execution

Ciet & Joye Algorithm

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

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

Faulty Execution

Ciet & Joye Algorithm

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

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

Faulty Execution

Ciet & Joye Algorithm

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

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

Faulty Execution

Ciet & Joye Algorithm

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

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

Faulty Execution

Ciet & Joye Algorithm

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

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

Outline

Fault Analysis

Consequences of the fault

The faulty result $\hat{S_{p*}}$ has been modeled as:

$$\hat{S_{p*}} = S_{p*} \oplus R_8 \cdot 2^{8i}$$

Fault Analysis

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:

$$\begin{aligned}
\hat{c_1} &\equiv (\hat{S^*} - s_1 + 1) \bmod r_1 \\
&\equiv 1 + R_8 \cdot 2^{8i} \bmod r_1 \\
&\approx 1 + R_8 \cdot 2^{8i} \\
c_2 &\equiv (\hat{S^*} - s_2 + 1) \bmod r_2 \\
&\equiv 1 \bmod r_2
\end{aligned}$$

Fault Analysis

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:

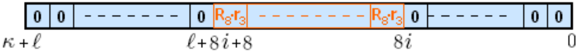
$$\begin{aligned}
\hat{c_1} &\equiv (\hat{S*} - s_1 + 1) \bmod r_1 \\
&\equiv 1 + R_8 \cdot 2^{8i} \bmod r_1 \\
&\approx 1 + R_8 \cdot 2^{8i} \\
c_2 &\equiv (\hat{S*} - s_2 + 1) \bmod r_2 \\
&\equiv 1 \bmod r_2
\end{aligned}$$

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

$$\begin{aligned}
\hat{\gamma} &= \left\lfloor \frac{(r_3 \cdot \hat{c_1} + (2^l - r_3) \cdot c_2)}{2^l} \right\rfloor \\
&= \left\lfloor \frac{R_8 \cdot r_3 \cdot 2^{8i}}{2^l} \right\rfloor + 1
\end{aligned}$$

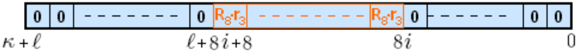
Fault Analysis

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

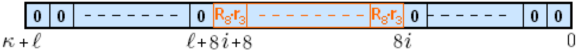


Fault Analysis

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

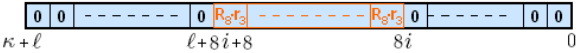


- Result of the right shift by l bits if $l > 8i$:

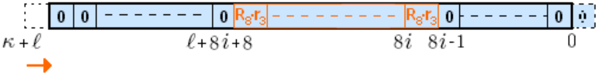


Fault Analysis

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

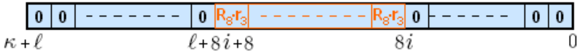


- Result of the right shift by l bits if $l > 8i$:

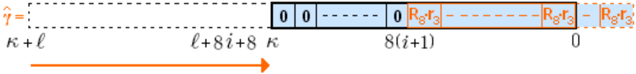


Fault Analysis

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

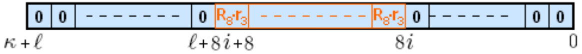


- Result of the right shift by l bits if $l > 8i$:

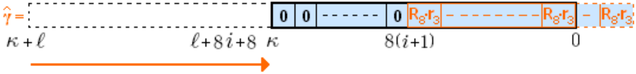


Fault Analysis

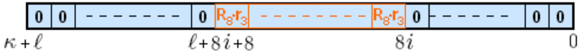
- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$



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

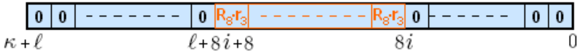


- Result of the right shift by l bits if $l < 8i$ and $l < \kappa$:

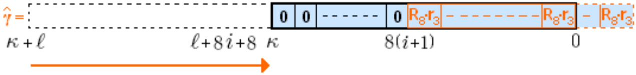


Fault Analysis

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



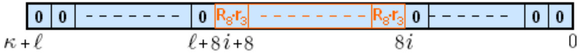
- Result of the right shift by l bits if $l > 8i$:



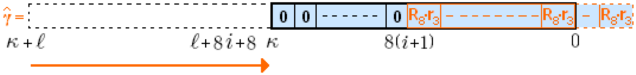
- Result of the right shift by l bits if $l < 8i$ and $l < \kappa$:

Fault Analysis

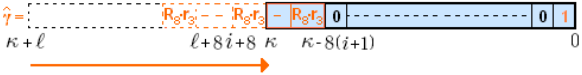
- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$



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

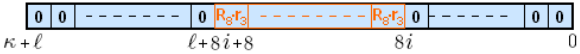


- Result of the right shift by l bits if $l < 8i$ and $l < \kappa$:

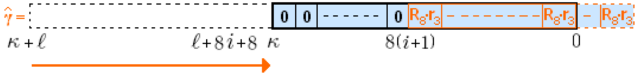


Fault Analysis

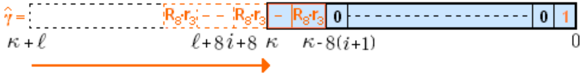
- Bit distribution of $R_8 \cdot r_3 \cdot 2^{8i}$



- 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:

$$\begin{aligned} \hat{S}^e \bmod N &= \dot{m}^{d \cdot e \cdot \hat{\gamma}} \bmod N \\ &= \dot{m}^{\hat{\gamma}} \bmod N \end{aligned}$$

Fault Analysis

- First, one can advantageously notice that:

$$\begin{aligned} \hat{S}^e \bmod N &= \dot{m}^{d \cdot e \cdot \hat{\gamma}} \bmod N \\ &= \dot{m}^{\hat{\gamma}} \bmod N \end{aligned}$$

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

Fault Analysis

- First, one can advantageously notice that:

$$\hat{S}^e \bmod N = \dot{m}^{d \cdot e \cdot \hat{\gamma}} \bmod N = \dot{m}^{\hat{\gamma}} \bmod N$$

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

Attack algorithm

1. The attacker chooses a candidate value for $\hat{\gamma}$
2. The attacker computes:

$$q' = gcd\left((\hat{S}^e - \dot{m}^{\hat{\gamma}}) \bmod N \, , N\right)$$

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

$$
\begin{aligned}
\mathbf{Pr}(\text{success}) &= \mathbf{Pr}\left[\hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \ \mathbf{\&}\ \hat{\gamma} \text{ is recoverable by brute force}\right] \\
&= \mathbf{Pr}\left[1 + R_8 \cdot 2^{8i} < r_1 \ \mathbf{\&}\ length(\hat{\gamma}) < B_f\right]
\end{aligned}
$$

Performance

- Success probability for a fault that suits the model

$$\begin{aligned}\mathbf{Pr}(\text{success}) &= \mathbf{Pr}\left[\hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \text{ \& } \hat{\gamma} \text{ is recoverable by brute force}\right] \\ &= \mathbf{Pr}\left[1 + R_8 \cdot 2^{8i} < r_1 \text{ \& } length(\hat{\gamma}) < B_f\right]\end{aligned}$$

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

Performance

- Success probability for a fault that suits the model

$$\begin{aligned}\mathbf{Pr}(\text{success}) &= \mathbf{Pr}\left[\hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \;\&\; \hat{\gamma} \text{ is recoverable by brute force}\right] \\ &= \mathbf{Pr}\left[1 + R_8 \cdot 2^{8i} < r_1 \;\&\; \mathit{length}(\hat{\gamma}) < B_f\right]\end{aligned}$$

- For $n = 1024$ bits, $\kappa = l = 80$ bits and $B_f = 40$ bits
  - **Pr**(success) $\approx$ **5,4%** for a suitable fault

Performance

- Success probability for a fault that suits the model

$$\begin{aligned}\mathbf{Pr}(\text{success}) &= \mathbf{Pr}\left[\hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \;\&\; \hat{\gamma} \text{ is recoverable by brute force}\right] \\ &= \mathbf{Pr}\left[1 + R_8 \cdot 2^{8i} < r_1 \;\&\; length(\hat{\gamma}) < B_f\right]\end{aligned}$$

- For $n = 1024$ bits, $\kappa = l = 80$ bits and $B_f = 40$ bits
  - **Pr**(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

$$\begin{aligned}\mathbf{Pr}(\text{success}) &= \mathbf{Pr}\left[\hat{c}_1 \approx 1 + R_8 \cdot 2^{8i} \;\&\; \hat{\gamma} \text{ is recoverable by brute force}\right] \\ &= \mathbf{Pr}\left[1 + R_8 \cdot 2^{8i} < r_1 \;\&\; \mathit{length}(\hat{\gamma}) < B_f\right]\end{aligned}$$

- For $n = 1024$ bits, $\kappa = l = 80$ bits and $B_f = 40$ bits
  - $\mathbf{Pr}(\text{success}) \approx$ **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 !