Fault Analysis of Infective AES Computations

Alberto Battistello and Christophe Giraud
Overview

- Introduction
- Attacks
  - FDTC 2012 Countermeasure
  - LatinCrypt 2012 Countermeasure
- Conclusion
Overview

• Introduction
• Attacks
  • FDTC 2012 Countermeasure
  • LatinCrypt 2012 Countermeasure
• Conclusion
Fault Attacks

\[ M \xrightarrow{\text{Algorithm}} C \]
Fault Attacks

$M \xrightarrow{\text{Disturbed Algorithm}} C'$
An example of Fault Attack

- Instead of computing \( S = M^d \mod N \)

\[
\begin{align*}
S_p & \leftarrow M^{d_p} \mod p \\
S_q & \leftarrow M^{d_q} \mod q
\end{align*}
\]

CRT-recombination

\[
\begin{align*}
S & \equiv S_p \mod p \\
S & \equiv S_q \mod q
\end{align*}
\]
An example of Fault Attack

- Instead of computing $S = M^d \mod N$

\[
\left\{
\begin{array}{l}
S_p \equiv S \mod p \\
S_q \equiv S \mod q
\end{array}
\right.
\]

\[
\left\{
\begin{array}{l}
S_p \equiv S_p \mod p \\
S_q \equiv S_q \mod q
\end{array}
\right.
\]

$S$ is computed as

\[
S = S_p \mod p = S_q \mod q
\]

$S$ is inferred as

\[
S' = S_p \mod p = S_q \mod q
\]

CRT-recombination

\[
\gcd(S' - S, N) = q
\]
An example of Fault Attack

- What a challenge for the countermeasure!!!
• Detection:
• Detection:

$M \xrightarrow{\text{Algorithm}} \neq \xrightarrow{\text{Algorithm}} \text{securityAction}$

• Drawbacks:
  • Attacks during comparison
  • Different paths to manage
• Infective:
• Infective:

\[ M \rightarrow \text{Algorithm} \rightarrow C \]

• Comparison with Detection:
  + No comparison
  + Single path
  − Could be much slower
Infective Countermeasures History

• Asymmetric:
  • [Yen, Kim, Lim, Moon] 2001 ➔ [Yen, Kim, Moon] 2004
  • [Schmidt et al.] 2010 ➔ [Feix, Venelli] 2013

• Symmetric:
  • [Lomné, Roche, Thillard] 2012
  • [Gierlichhs, Schmidt, Tunstall] 2012
Overview

- Introduction
- Attacks
  - FDTC 2012 Countermeasure
  - LatinCrypt 2012 Countermeasure
- Conclusion
- For efficiency, multiplication is performed byte per byte
- Restriction on the multiplicative mask:
  - $R_i$ must be different from 0 and 1
For efficiency, multiplication is performed byte per byte

Restriction on the multiplicative mask:

- $R_i$ must be different from 0 and 1
AfricaCrypt 2009: Mukhopadhyay shows that:

\[(C, C^{\frac{1}{2}})\] gives the AES-128 key

if a byte-fault has disturbed the 8\textsuperscript{th} round.

⇒ Goal for the attacker: Recover \(C^{\frac{1}{2}}\) from \(C^{\text{\textcopyright}}\):

\[C^{\text{\textcopyright}}_i = C^{\frac{1}{2}}_i \oplus \Delta_i \cdot R_i\]

where \(\Delta_i = C_i \oplus C^{\frac{1}{2}}_i\) and \(R_i\) a random value \(\neq \{0, 1\}\).

Let us assume a constant fault model (i.e. \(\Delta\) cst):

\[R_i = 2 \quad C^{\text{\textcopyright}}_i = C^{\frac{1}{2}}_i \oplus 2 \cdot \Delta_i\]

\[R_i = 3 \quad C^{\text{\textcopyright}}_i = C^{\frac{1}{2}}_i \oplus 3 \cdot \Delta_i\]

\[\ldots\]

\[R_i = 255 \quad C^{\text{\textcopyright}}_i = C^{\frac{1}{2}}_i \oplus 255 \cdot \Delta_i\]

⇒ 2 values never appear: \(C^{\frac{1}{2}}_i\) and \(C^{\frac{1}{2}}_i \oplus \Delta_i = C_i\)
- **Attack procedure:**

  1. Inject a constant byte error during round 8 to obtain $C^*$
  2. For each byte $i$, remove $C^*_i$ from the list of possible values for $C'_i$
  3. If one $C'_i$ has more than 2 possible values, then go back to Step 1
  4. Identify each $C'_i$ since $C_i$’s are known
  5. Apply Mukhopadhyay’s attack to $(C, C')$ to recover the secret key
Simulations

- With 3000 $C^{\star\star}$'s, the AES key is recovered with 99% success rate.
Overview

• Introduction
• Attacks
  • FDTC 2012 Countermeasure
  • LatinCrypt 2012 Countermeasure
• Conclusion
LatinCrypt 2012 Countermeasure

\[ M \rightarrow \text{Effective Round} \rightarrow \text{Effective Round} \rightarrow \cdots \rightarrow \text{Effective Round} \rightarrow C \]

\[ M \rightarrow \text{Redundant Round} \rightarrow \text{Redundant Round} \rightarrow \cdots \rightarrow \text{Redundant Round} \rightarrow C \]

\[ \beta \rightarrow \text{Dummy Round} \rightarrow \beta \]
Fault Analysis of Infective AES Computations – FDTC 2013

LatinCrypt 2012 Countermeasure

\[ M \rightarrow \text{Redundant} \rightarrow \text{Effective} \rightarrow \text{Redundant} \rightarrow \text{Effective} \rightarrow \ldots \rightarrow \text{Redundant} \rightarrow \text{Effective} \rightarrow C \]

\[ \beta \rightarrow \text{Dummy Round} \rightarrow \beta \]
First Infective Mechanism

Redundant Round

Effective Round

© Oberthur Technologies 2013
First Infective Mechanism

- Redundant Round
- Effective Round
- Dummy Round

- Inv

Flow diagram illustrating the first infective mechanism with redundant, effective, and dummy rounds.
On the Use of Dummy Rounds

\[ M \rightarrow \text{Redundant} \rightarrow \text{Effective} \rightarrow \text{Dummy} \rightarrow \text{Redundant} \rightarrow \text{Dummy} \rightarrow \ldots \rightarrow \text{Dummy} \rightarrow \text{Effective} \rightarrow \text{Redundant} \rightarrow \text{Dummy} \rightarrow C \]

\[ M \rightarrow \text{Dummy} \rightarrow \text{Redundant} \rightarrow \text{Dummy} \rightarrow \text{Effective} \rightarrow \text{Redundant} \rightarrow \ldots \rightarrow \text{Redundant} \rightarrow \text{Effective} \rightarrow \text{Dummy} \rightarrow C \]
Second Infective Mechanism

Dummy Round
Second Infective Mechanism

- Redundant Round
- Effective Round
- Dummy Round

- Redundant Round
- Effective Round
- Dummy Round

β
A Useful Remark
A Useful Remark

Effective Round

Dummy Round
A Useful Remark

Redundant Round

Effective Round

Inv

Dummy Round

C

C

0

β

β

C
A Useful Remark

Redundant Round

Effective Round

1\textsuperscript{st} infection

\(C\)

\(C'\)

\(\beta\)

2\textsuperscript{nd} infection

\(\beta\)

\(\text{Inv}\)

\((1)\)

\((2)\)

Dummy Round
First Infection Analysis

- If disturbance of a byte of the input, the differential is:

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
e & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\xrightarrow{\text{SubBytes}}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
\alpha & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\xrightarrow{\text{ShiftRows}}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}

- So the first infection is equal to:

\[
(1) = \text{Inv}(C \oplus C'\ddagger) =
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha^{-1} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]
Second Infection Analysis

\[
\text{Round} \left( \beta \oplus \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{-1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right) \oplus \beta = \begin{pmatrix} 0 & 0 & \delta_0 & 0 \\ 0 & 0 & \delta_1 & 0 \\ 0 & 0 & \delta_2 & 0 \\ 0 & 0 & \delta_3 & 0 \end{pmatrix}
\]
The infected output is defined by:

$$C^{\text{inf}} = C^{\text{inf}}_1 \oplus C^{\text{inf}}_2 \oplus C^{\text{inf}}_3$$

Therefore, we have:

$$C^{\text{inf}} = C^{\text{inf}}_1 \oplus \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{-1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & \delta_0 & 0 \\ 0 & 0 & \delta_1 & 0 \\ 0 & 0 & \delta_2 & 0 \\ 0 & 0 & \delta_3 & 0 \end{pmatrix}$$

which is equivalent to:

$$C^{\text{inf}} = C^{\text{inf}}_1 \oplus \begin{pmatrix} 0 & 0 & \delta_0 & 0 \\ 0 & 0 & \delta_1 & \alpha^{-1} \\ 0 & 0 & \delta_2 & 0 \\ 0 & 0 & \delta_3 & 0 \end{pmatrix}$$
By using:

\[
C^{\dagger} = C^{\dagger} \oplus \begin{pmatrix}
0 & 0 & \delta_0 & 0 \\
0 & 0 & \delta_1 & \alpha^{-1} \\
0 & 0 & \delta_2 & 0 \\
0 & 0 & \delta_3 & 0
\end{pmatrix}
\]

and

\[
C \oplus C^{\dagger} = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

we obtain:

\[
C \oplus C^{\dagger} = \begin{pmatrix}
0 & 0 & \delta_0 & 0 \\
0 & 0 & \delta_1 & \alpha \oplus \alpha^{-1} \\
0 & 0 & \delta_2 & 0 \\
0 & 0 & \delta_3 & 0
\end{pmatrix}
\]

- The byte \(\alpha\) contains information on the key but:
  - \((1)\) does not efficiently blind this value
  - \((2)\) has no effect due to ShiftRows transformation
To sum up, we have:
\[ C_{13} \oplus C_{13}^{\text{ib}} = \alpha \oplus \alpha^{-1} \]

with
\[ \alpha = \text{SB}(s \oplus e) \oplus \text{SB}(s) \]

where \( s \) is the second input byte of the last effective round.

The byte \( s \) can thus be expressed as:
\[ s = \text{SB}^{-1}(C_{13} \oplus k_{13}) \]

The attack process is thus the following:

1. Guess the corresponding key byte \( k_h \in \{0, \cdots, 255\} \)
2. Compute \( s_h = \text{SB}^{-1}(C_{13} \oplus k_h) \)
3. Guess the error value \( e_h \in \{1, \cdots, 255\} \)
4. Compute \( \alpha_h = \text{SB}(s_h \oplus e_h) \oplus \text{SB}(s_h) \)
5. If \( C_{13} \oplus C_{13}^{\text{ib}} \neq \alpha_h \oplus \alpha_h^{-1} \) then discard \((k_h, e_h)\)
Simulations

- With 37 $C^*$'s, the last three rows of the AES key are recovered with 99% success rate.
Overview

- Introduction
- Attacks
  - FDTC 2012 Countermeasure
  - LatinCrypt 2012 Countermeasure
- Conclusion
The two existing symmetric infective countermeasures are flawed
Easy to patch but a framework is missing to formally prove countermeasures’ security
After 10 years of research in infective countermeasures, no original proposal has survived...
Do infective countermeasures have a future?
Any Questions?