To exploit fault injection on non-injective Sboxes

Guillaume BETHOUART
Nicolas DEBANDE
Agenda

1. Introduction
   - Overview of fault attacks
   - Principle of our attack

2. Application to the Data Encryption Standard
   - Data Encryption Standard
   - Attack Simulation
   - Countermeasures

3. Conclusion
Outline

1. Introduction
   - Overview of fault attacks
   - Principle of our attack

2. Application to the Data Encryption Standard
   - Data Encryption Standard
   - Attack Simulation
   - Countermeasures

3. Conclusion

FDTC 15' To exploit fault injection on non-injective Sboxes September 13, 2015
Overview of fault attacks

- Safe Error Attacks
  + Just need to know if the calculus has been disturbed or not

- Differential Fault Attacks
  + Work with masked implementations

- Collision Fault Attacks
  + Do not need to encrypt the same plaintext

Take the best of each
Overview of fault attacks

- **Safe Error Attacks**
  - Just need to know if the calculus has been disturbed or not

- **Differential Fault Attacks**
  - Work with masked implementations

- **Collision Fault Attacks**
  - Do not need to encrypt the same plaintext

Take the best of each

FDTC 15'

To exploit fault injection on non-injective Sboxes

September 13, 2015
1. Introduction
   - Overview of fault attacks
   - Principle of our attack

2. Application to the Data Encryption Standard
   - Data Encryption Standard
   - Attack Simulation
   - Countermeasures

3. Conclusion
A non-injective Sbox from $\mathbb{F}_2^3$ to $\mathbb{F}_2^2$:

- Non injectivity:
  - There exist two different inputs $a_1, a_2$ such as $S(a_1) = S(a_2)$
  - There are an input $a$ and a differential $\delta$ such as $S(a \oplus \delta) = S(a)$

N-Differential:
For a given $\delta$, if there exists $a$ such as $S(a \oplus \delta) = S(a)$, $\delta$ is called a N-differential.
A non-injective Sbox from $\mathbb{F}_2^3$ to $\mathbb{F}_2^2$:

- Non injectivity
  - there exist two different inputs $a_1, a_2$ such as $S(a_1) = S(a_2)$
  - there are an input $a$ and a differential $\delta$ such as $S(a \oplus \delta) = S(a)$

N-Differential
For a given $\delta$, if there exists $a$ such as $S(a \oplus \delta) = S(a)$, $\delta$ is called a N-differential.
A non-injective Sbox from $\mathbb{F}_2^3$ to $\mathbb{F}_2^2$:

- there exist two different inputs $a_1, a_2$ such as $S(a_1) = S(a_2)$
- there are an input $a$ and a differential $\delta$ such as $S(a \oplus \delta) = S(a)$

Non injectivity

N-Differential

For a given $\delta$, if there exists a such as $S(a \oplus \delta) = S(a)$, $\delta$ is called a N-differential.
A non-injective Sbox from $\mathbb{F}_2^3$ to $\mathbb{F}_2^2$:

- Non injectivity
  - there exist two different inputs $a_1, a_2$ such as $S(a_1) = S(a_2)$
  - there are an input $a$ and a differential $\delta$ such as $S(a \oplus \delta) = S(a)$

N-Differential

For a given $\delta$, if there exists $a$ such as $S(a \oplus \delta) = S(a)$, $\delta$ is called a **N-differential**.
Introduction

Principle of our attack

Truth table

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a known fault $\delta = 4$

- $S(0 \oplus \delta) = S(4) \neq S(0)$
- $S(1 \oplus \delta) = S(5) \neq S(1)$
- $S(2 \oplus \delta) = S(6) = S(2)$
- $S(3 \oplus \delta) = S(7) \neq S(3)$
Introduction

Principle of our attack

Truth table

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a **known** fault $\delta = 4$

- $S(0 \oplus \delta) = S(4) \neq S(0)$
- $S(1 \oplus \delta) = S(5) \neq S(1)$
- $S(2 \oplus \delta) = S(6) = S(2)$
- $S(3 \oplus \delta) = S(7) \neq S(3)$
Principle of our attack

Truth table

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a known fault $\delta = 4$

$$S(0 \oplus \delta) = S(4) \neq S(0)$$
$$S(1 \oplus \delta) = S(5) \neq S(1)$$
$$S(2 \oplus \delta) = S(6) = S(2)$$
$$S(3 \oplus \delta) = S(7) \neq S(3)$$
Principle of our attack

Truth table

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a **known** fault $\delta = 4$

$$S(0 \oplus \delta) = S(4) \neq S(0)$$

$$S(1 \oplus \delta) = S(5) \neq S(1)$$

$$S(2 \oplus \delta) = S(6) = S(2)$$

$$S(3 \oplus \delta) = S(7) \neq S(3)$$
**Introduction**

**Principle of our attack**

**Truth table**

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

**Example**

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a known fault $\delta = 4$

- $S(0 \oplus \delta) = S(4) \neq S(0)$
- $S(1 \oplus \delta) = S(5) \neq S(1)$
- $S(2 \oplus \delta) = S(6) = S(2)$
- $S(3 \oplus \delta) = S(7) \neq S(3)$
** Principle of our attack **

**Truth table**

<table>
<thead>
<tr>
<th>a</th>
<th>S(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

**Example**

If the calculus is not disturbed by the fault $\delta$, we know:

$$S(a \oplus \delta) = S(a)$$

For a **known** fault $\delta = 4$

- $S(0 \oplus \delta) = S(4) \neq S(0)$
- $S(1 \oplus \delta) = S(5) \neq S(1)$
- $S(2 \oplus \delta) = S(6) = S(2)$
- $S(3 \oplus \delta) = S(7) \neq S(3)$
Principle of our attack

Result

For a known fault $\delta = 4$
If

$$S(a \oplus \delta) = S(a)$$

We deduce:

$$a = 2 \text{ or } a = 6$$

To deduce information about the input we only need to know:

- The fault value $\delta$
- If the calculus is disturbed or not
1 Introduction
   - Overview of fault attacks
   - Principle of our attack

2 Application to the Data Encryption Standard
   - Data Encryption Standard
   - Attack Simulation
   - Countermeasures

3 Conclusion
DES follows a Feistel scheme:

- 64-bit block cipher using a 56-bit key $k$
- 16 times the same round transformation $f$
DES follows a Feistel scheme:

- 64-bit block cipher using a 56-bit key $k$
- 16 times the same round transformation $f$
DES follows a Feistel scheme:

- 64-bit block cipher using a 56-bit key $k$
- 16 times the same round transformation $f$
Round function $f$

- Expansion function
- 48-bit round key $k_r$
- 8 different non-injective Sboxes
- Permutation

To exploit fault injection on non-injective Sboxes
**Round function** $f$

- **Expansion function**
  - 48-bit round key $k_r$
  - 8 different non-injective Sboxes
  - Permutation

To exploit fault injection on non-injective Sboxes
Application to the Data Encryption Standard

Data Encryption Standard

- Round function \( f \)
- Expansion function
- 48-bit round key \( k_r \)
- 8 different non-injective Sboxes
- Permutation

FDTC 15'

To exploit fault injection on non-injective Sboxes

September 13, 2015
Round function $f$

- Expansion function
- 48-bit round key $k_r$
- 8 different non-injective Sboxes
- Permutation
• Round function $f$

- Expansion function
- 48-bit round key $k_r$
- 8 different non-injective Sboxes
- Permutation

To exploit fault injection on non-injective Sboxes September 13, 2015
Application to the Data Encryption Standard

Attack timing

- First or last round
- After the data propagation
- Before Sboxes
- Fault affects only one Sbox
Application to the Data Encryption Standard

Attack timing

- First or last round
- After the data propagation
- Before Sboxes
- Fault affects only one Sbox

FDTC 15'

To exploit fault injection on non-injective Sboxes

September 13, 2015
If the fault value is known

If we know $S(a \oplus \delta) = S(a)$ we deduce information on $a$

During the DES: $a = x \oplus k$, $x$ the Expansion output and $k$ the key

If we know:
- The fault $\delta$
- The Expansion output $x$
- If $S(x \oplus k \oplus \delta) = S(x \oplus k)$ or not

We deduce information on $k$

But it's a too restrictive model

- Fault injection does not have a 100% success rate (missed faults)
- The fault value is rarely constant
If the fault value is known

If we know \( S(a \oplus \delta) = S(a) \) we deduce information on \( a \)

During the DES: \( a = x \oplus k \), \( x \) the Expansion output and \( k \) the key

If we know:
- The fault \( \delta \)
- The Expansion output \( x \)
- If \( S(x \oplus k \oplus \delta) = S(x \oplus k) \) or not

We deduce information on \( k \)

But it's a too restrictive model
- Fault injection does not have a 100% success rate (missed faults)
- The fault value is rarely constant
If the fault value is known

If we know \( S(a \oplus \delta) = S(a) \) we deduce information on \( a \)

During the DES: \( a = x \oplus k \), \( x \) the Expansion output and \( k \) the key

If we know:
- The fault \( \delta \)
- The Expansion output \( x \)
- If \( S(x \oplus k \oplus \delta) = S(x \oplus k) \) or not

We deduce information on \( k \)

But it's a too restrictive model

- Fault injection does not have a 100% success rate (missed faults)
- The fault value is rarely constant
Application to the Data Encryption Standard

Attack with known fault

If the fault value is known

If we know $S(a \oplus \delta) = S(a)$ we deduce information on $a$

During the DES: $a = x \oplus k$, $x$ the Expansion output and $k$ the key

If we know:

- The fault $\delta$
- The Expansion output $x$
- If $S(x \oplus k \oplus \delta) = S(x \oplus k)$ or not

We deduce information on $k$

But its a too restrictive model

- Fault injection does not have a 100% success rate (missed faults)
- The fault value is rarely constant

FDTC 15'

To exploit fault injection on non-injective Sboxes

September 13, 2015
Application to the Data Encryption Standard

Attack with known fault

If the fault value is known

If we know $S(a \oplus \delta) = S(a)$ we deduce information on $a$

During the DES: $a = x \oplus k$, $x$ the Expansion output and $k$ the key

If we know:

- The fault $\delta$
- The Expansion output $x$
- If $S(x \oplus k \oplus \delta) = S(x \oplus k)$ or not

We deduce information on $k$

But it's a too restrictive model

- Fault injection does not have a 100% success rate (missed faults)
- The fault value is rarely constant
Characterization stage

Characterization:
- Fault injection with known key
- We estimate a fault occurrence probability $p$ for each fault value

Attack stage

Attack:
- If the fault has no effect
  - For each $(\delta, p)$
  - For each $k \in [0, 63]$
  - If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
  - $\quad counter[k] + = p$
Characterization stage

Characterization:

- Fault injection with known key
- We estimate a fault occurrence probability $p$ for each fault value

Attack stage

Attack:

If the fault has no effect

1. For each $(\delta, p)$
2. For each $k \in \{0, 63\}$
3. If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
4. $\text{counter}[k] + = p$
Characterization stage

Characterization:
- Fault injection with known key
- We estimate a fault occurrence probability $p$ for each fault value

Attack stage

Attack:
If the fault has no effect
- For each $(\delta, p)$
  - For each $k \in [0, 63]$
    - If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
      - $\text{counter}[k] + = p$
Characterization stage

Characterization:
- Fault injection with known key
- We estimate a fault occurrence probability $p$ for each fault value

Attack stage

Attack:
If the fault has no effect
- For each $(\delta, p)$
  - For each $k \in [0, 63]$
    - If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
      - $\text{counter}[k] += p$
Get information when fault has an effect

If the fault has an effect
  For each $(\delta, p)$
    For each $k \in \mathbb{Z}_{64}$
      If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
        $\text{counter}[k] = p$
Get information when fault has an effect

If the fault has an effect
  . For each $(\delta, p)$
  . . For each $k \in [0, 63]$
  . . . If $S(x \oplus k \oplus \delta) = S(x \oplus k)$
  . . . . $\text{counter}[k] \leftarrow p$
Get information when fault has an effect

If the fault has an effect
 . For each \((\delta, p)\)
 . . For each \(k \in [0, 63]\)
 . . . If \(S(x \oplus k \oplus \delta) = S(x \oplus k)\)
 . . . . . \(\text{counter}[k] = p\)

To exploit fault injection on non-injective Sboxes
Combined algorithm

For each \((\delta, p)\)
  . For each \(k \in [0, 63]\)
  . . If \(S(x \oplus k \oplus \delta) = S(x \oplus k)\)
  . . . If the fault has an effect
  . . . . \(counter[k] -= p\)
  . . . else
  . . . . \(counter[k] += p\)
How it works with masked implementation

- To build a masked Sbox $S'$: $\forall a$
  $$S'(a \oplus z_1) = S(a) \oplus z_2$$

- Then
  $$\text{if } S'(a \oplus z_1 \oplus \delta) = S'(a \oplus z_1)$$
  $$\implies \text{we have } S(a \oplus \delta) \oplus z_2 = S(a) \oplus z_2$$

$$S(a \oplus \delta) = S(a)$$
To build a masked Sbox $S'$: $\forall a$

$$S'(a \oplus z_1) = S(a) \oplus z_2$$

Then

if

$$S'(a \oplus z_1 \oplus \delta) = S'(a \oplus z_1)$$

we have

$$S(a \oplus \delta) \oplus z_2 = S(a) \oplus z_2$$

$$S(a \oplus \delta) = S(a)$$
How it works with masked implementation

- To build a masked Sbox $S'$: $\forall a$

$$S'(a \oplus z_1) = S(a) \oplus z_2$$

- Then

$$S'(a \oplus z_1 \oplus \delta) = S'(a \oplus z_1)$$

we have

$$S(a \oplus \delta) \oplus z_2 = S(a) \oplus z_2$$

$$S(a \oplus \delta) = S(a)$$
1 Introduction
   ■ Overview of fault attacks
   ■ Principle of our attack

2 Application to the Data Encryption Standard
   ■ Data Encryption Standard
   ■ Attack Simulation
   ■ Countermeasures

3 Conclusion
Random plaintexts and random keys
Theoretical fault distribution
Mean of 1000 simulations
**Fault Distribution**

\[
\begin{align*}
HW(\delta) = 0 & \rightarrow p = 0 \\
HW(\delta) = 1 & \rightarrow p = 0 \\
HW(\delta) = 2 & \rightarrow p = 0.013 \\
HW(\delta) = 3 & \rightarrow p = 0.02 \\
HW(\delta) = 4 & \rightarrow p = 0.027 \\
HW(\delta) = 5 & \rightarrow p = 0 \\
HW(\delta) = 6 & \rightarrow p = 0
\end{align*}
\]
Rank of the key when fault number increases
Comparison between the 3 possible models

![Graph showing comparison between fault injection models](image)
1 Introduction
   ■ Overview of fault attacks
   ■ Principle of our attack

2 Application to the Data Encryption Standard
   ■ Data Encryption Standard
   ■ Attack Simulation
   ■ Countermeasures

3 Conclusion
Countermeasures

Fault counter

- Do the calculus twice, compare and increase the counter in case of different results
- When the counter limit is reached: Block the device

- Our attack is theoretically possible
- The success depends on the counter limit

An error correction countermeasure

- Do the calculus three times
- Return the result obtained twice

- The attacker cannot know if a fault has an effect or not
- Our attack is no longer possible
Fault counter

- Do the calculus twice, compare and increase the counter in case of different results
- When the counter limit is reached: Block the device
- Our attack is theoretically possible
- The success depends on the counter limit

An error correction countermeasure

- Do the calculus three times
- Return the result obtained twice
- The attacker cannot know if a fault has an effect or not
- Our attack is no longer possible
Fault counter

- Do the calculus twice, compare and increase the counter in case of different results
- When the counter limit is reached: Block the device

- Our attack is theoretically possible
- The success depends on the counter limit

An error correction countermeasure

- Do the calculus three times
- Return the result obtained twice

- The attacker cannot know if a fault has an effect or not
- Our attack is no longer possible
### Fault counter
- Do the calculus twice, compare and increase the counter in case of different results
- When the counter limit is reached: Block the device
- Our attack is theoretically possible
- The success depends on the counter limit

### An error correction countermeasure
- Do the calculus three times
- Return the result obtained twice
- The attacker cannot know if a fault has an effect or not
- Our attack is no longer possible
Introduction
- Overview of fault attacks
- Principle of our attack

Application to the Data Encryption Standard
- Data Encryption Standard
- Attack Simulation
- Countermeasures

Conclusion
### Comparison

<table>
<thead>
<tr>
<th>Safe Error</th>
<th>DFA</th>
<th>CFA</th>
<th>Our Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works with masked implementation</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to encrypt the same plaintext</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to know the calculus output</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Fault number ≃</td>
<td>100</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>
# Conclusion

## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Safe Error</th>
<th>DFA</th>
<th>CFA</th>
<th>Our Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works with masked implementation</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to encrypt the same plaintext</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to know the calculus output</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault number $\approx$</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>10000</td>
</tr>
</tbody>
</table>
### Conclusion

#### Comparison

<table>
<thead>
<tr>
<th></th>
<th>Safe Error</th>
<th>DFA</th>
<th>CFA</th>
<th>Our Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works with masked implementation</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to encrypt the same plaintext</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not need to know the calculus output</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Fault number ( \sim )</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>10000</td>
</tr>
<tr>
<td>Safe Error</td>
<td>DFA</td>
<td>CFA</td>
<td>Our Attack</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>-----</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>Works with masked implementation</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Does not need to encrypt the same plaintext</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Does not need to know the calculus output</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Fault number $\approx$</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>10000</td>
</tr>
</tbody>
</table>
Conclusion

The End

Any Questions?
Any Questions?