Practical Multiple Persistent Fault Analysis

Hadi Soleimany Nasour Bagheri **Hosein Hadipour** Prasanna Ravi Shivam Bhasin
Sara Mansouri

CHES 2022 - Leuven, Belgium

> hsn.hadipour@gmail.com

Outline

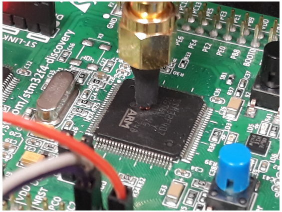
Introduction and the Research Gap

Fault Attacks

Fault attack: An active side-channel attack [BDL97]:

- **Fault injection**: Disturb the operation of a cryptographic device
- **Fault analysis**: Analyze the erroneous outputs to retrieve the secret key

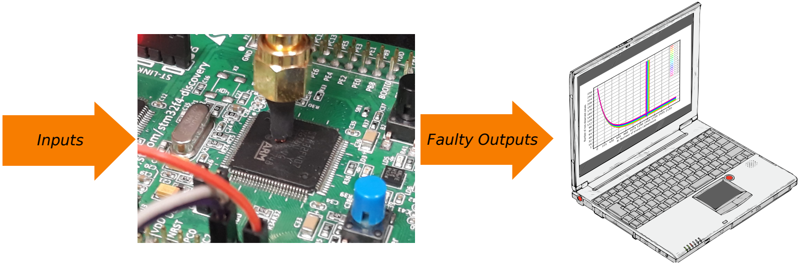
Inputs



Fault Attacks

Fault attack: An active side-channel attack [BDL97]:

- **Fault injection**: Disturb the operation of a cryptographic device
- **Fault analysis**: Analyze the erroneous outputs to retrieve the secret key



Persistent Fault Attack (PFA)

PFA fault model [Zha+18]:

- The injected faults are persistent until the reset of the device
- The injected faults typically alter the stored algorithm constants
- We can inject the faults before the encryption
- We can collect multiple faulty ciphertexts

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

Persistent Fault Attack (PFA)

PFA fault model [Zha+18]:

- The injected faults are persistent until the reset of the device
- The injected faults typically alter the stored algorithm constants
- We can inject the faults before the encryption
- We can collect multiple faulty ciphertexts

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

Persistent Fault Attack (PFA)

PFA fault model [Zha+18]:

- The injected faults are persistent until the reset of the device
- The injected faults typically alter the stored algorithm constants
- We can inject the faults before the encryption
- We can collect multiple faulty ciphertexts

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

Persistent Fault Attack (PFA)

PFA fault model [Zha+18]:

- The injected faults are persistent until the reset of the device
- The injected faults typically alter the stored algorithm constants
- We can inject the faults before the encryption
- We can collect multiple faulty ciphertexts

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

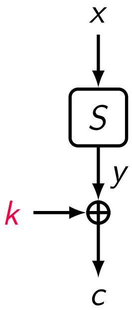
Core Idea of PFA

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

$$c = S(x) + k$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

$$c = S'(x) \oplus k$$



- Filter wrong keys: $S'(x) \neq 0\text{xe} \Rightarrow k \neq 0\text{xe} \oplus c$
- Filter wrong keys: $S'(X[i]) \neq 0\text{xe} \Rightarrow K[i] \neq 0\text{xe} \oplus C[i]$

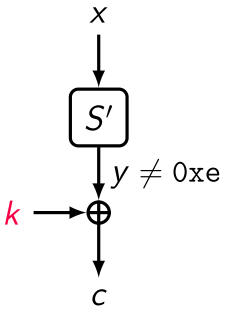
Core Idea of PFA

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

$$c = S(x) + k$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

$$c = S'(x) \oplus k$$



- Filter wrong keys: $S'(x) \neq \texttt{0xe} \Rightarrow k \neq \texttt{0xe} \oplus c$
- Filter wrong keys: $S'(X[i]) \neq \texttt{0xe} \Rightarrow K[i] \neq \texttt{0xe} \oplus C[i]$

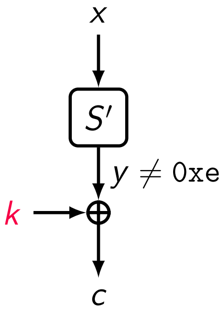
Core Idea of PFA

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

$$c = S(x) + k$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

$$c = S'(x) \oplus k$$



- Filter wrong keys: $S'(x) \neq 0\text{xe} \Rightarrow k \neq 0\text{xe} \oplus c$
- Filter wrong keys: $S'(X[i]) \neq 0\text{xe} \Rightarrow K[i] \neq 0\text{xe} \oplus C[i]$

Core Idea of PFA

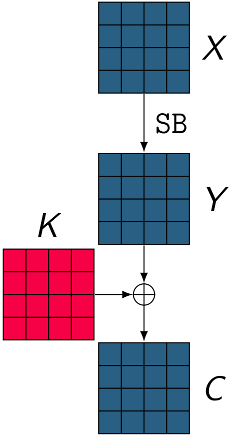
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

$$c = S(x) + k$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}'(x)$	6	4	c	5	0	7	2	4	1	f	3	d	8	a	9	b

$$c = S'(x) \oplus k$$

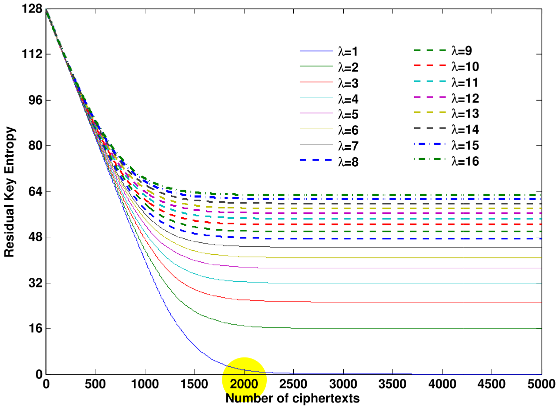
- Filter wrong keys: $S'(x) \neq \mathtt{0xe} \Rightarrow k \neq \mathtt{0xe} \oplus c$
- Filter wrong keys: $S'(X[i]) \neq \mathtt{0xe} \Rightarrow K[i] \neq \mathtt{0xe} \oplus C[i]$

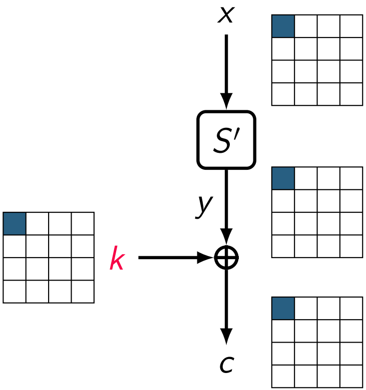


Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Limits of the Original PFA

PFA requires about 2000 faulty ciphertexts per key [Zha+18]



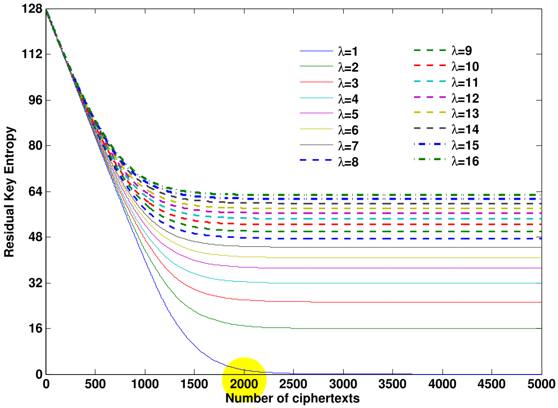


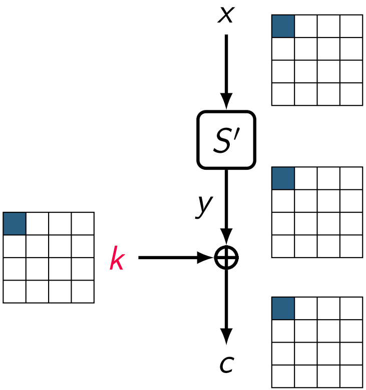
$$N \gtrapprox 2000 \Rightarrow |K| = \lambda^{16}$$

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Limits of the Original PFA

PFA is very time consuming for multiple faults [Zha+18]





$$\lambda = 12 \Rightarrow |K| = 12^{16} \approx 2^{57.36}$$

More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

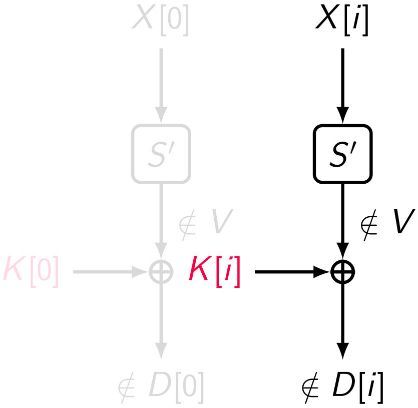
More Limits of PFA and Its Enhanced Versions

- The location of the injected fault is supposed to be known
- For multiple fault injections:
  - We need a known plaintext/ciphertext pair to detect the correct key
- PFA only exploits the fault leakage in the last round
- Enhanced PFA (EPFA) [Xu+21] exploits the fault leakage in multiple rounds
- However, EPFA is not clear about exploiting multiple faults in deeper rounds
- Morever, EPFA still relies on the assumption of knowing the fault location

Our Framework for PFA With Multiple Faults

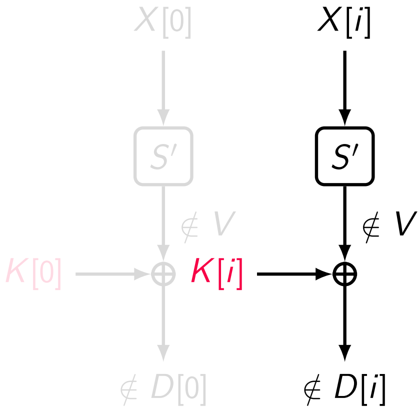
Core Idea

- V: Impossible values in the output of faulty S-box
- $D[i]$: Impossible values in the ith word of ciphertext
- $D[i] = V \oplus K[i]$ for all $i \in \{1, \ldots, 15\}$
- $V = K[0] \oplus D[0]$
- $D[i] = (K[0] \oplus K[i]) \oplus D[0]$
- $\delta[i] = K[0] \oplus K[i]$
- We can derive $\delta[i]$ from $(D[0], D[i])$



Core Idea

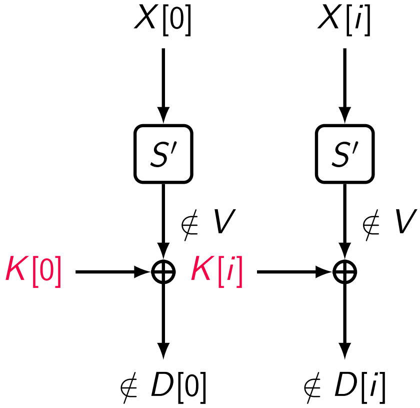
- V: Impossible values in the output of faulty S-box
- $D[i]$: Impossible values in the ith word of ciphertext
- $D[i] = V \oplus K[i]$ for all $i \in \{1, \ldots, 15\}$
- $V = K[0] \oplus D[0]$
- $D[i] = (K[0] \oplus K[i]) \oplus D[0]$
- $\delta[i] = K[0] \oplus K[i]$
- We can derive $\delta[i]$ from $(D[0], D[i])$



Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Core Idea

- V: Impossible values in the output of faulty S-box
- $D[i]$: Impossible values in the ith word of ciphertext
- $D[i] = V \oplus K[i]$ for all $i \in \{1, \ldots, 15\}$
- $V = K[0] \oplus D[0]$
- $D[i] = (K[0] \oplus K[i]) \oplus D[0]$
- $\delta[i] = K[0] \oplus K[i]$
- We can derive $\delta[i]$ from $(D[0], D[i])$

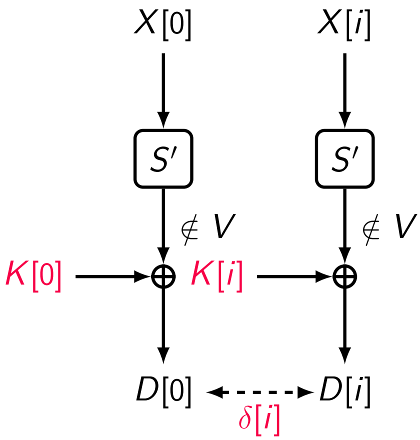


Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

CHES 2022 - Leuven, Belgium

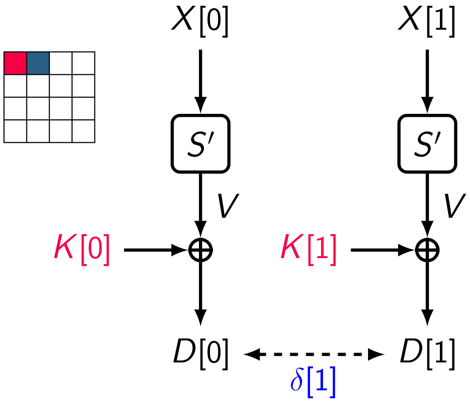
Core Idea

- V: Impossible values in the output of faulty S-box
- $D[i]$: Impossible values in the ith word of ciphertext
- $D[i] = V \oplus K[i]$ for all $i \in \{1, \ldots, 15\}$
- $V = K[0] \oplus D[0]$
- $D[i] = (K[0] \oplus K[i]) \oplus D[0]$
- $\delta[i] = K[0] \oplus K[i]$
- We can derive $\delta[i]$ from $(D[0], D[i])$



Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

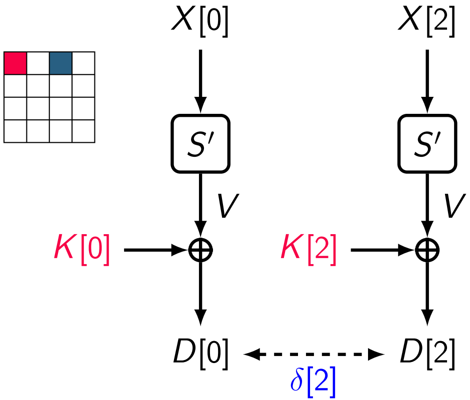
Reducing the Number of Key Candidates to 2^8



$\delta[1] = K[0] \oplus K[1]$

Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

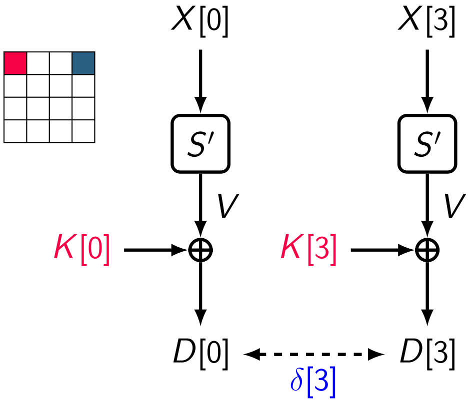
Reducing the Number of Key Candidates to 2^8



$$\delta[1] = K[0] \oplus K[1]$$
$$\delta[2] = K[0] \oplus K[2]$$

Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

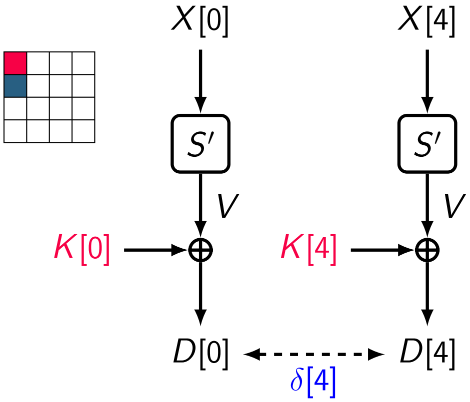
Reducing the Number of Key Candidates to 2^8



$$\delta[1] = K[0] \oplus K[1]$$
$$\delta[2] = K[0] \oplus K[2]$$
$$\delta[3] = K[0] \oplus K[3]$$

Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

Reducing the Number of Key Candidates to 2^8



$\delta[1] = K[0] \oplus K[1]$

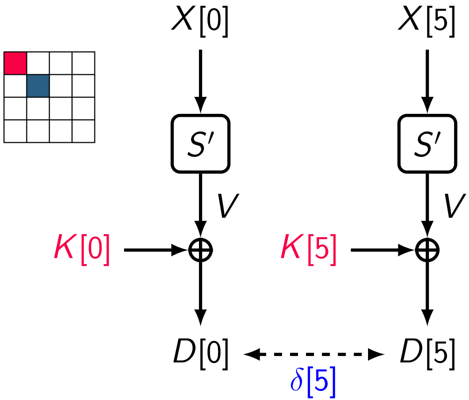
$\delta[2] = K[0] \oplus K[2]$

$\delta[3] = K[0] \oplus K[3]$

$\delta[4] = K[0] \oplus K[4]$

Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

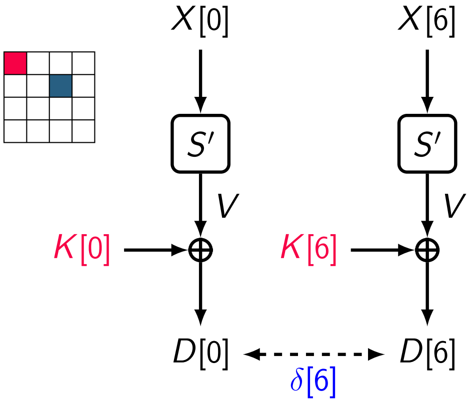
Reducing the Number of Key Candidates to 2^8



$$\delta[1] = K[0] \oplus K[1]$$
$$\delta[2] = K[0] \oplus K[2]$$
$$\delta[3] = K[0] \oplus K[3]$$
$$\delta[4] = K[0] \oplus K[4]$$
$$\delta[5] = K[0] \oplus K[5]$$

- Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

Reducing the Number of Key Candidates to 2^8

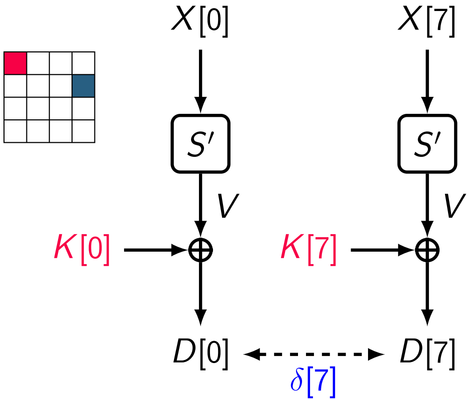


$$\delta[1] = K[0] \oplus K[1]$$
$$\delta[2] = K[0] \oplus K[2]$$
$$\delta[3] = K[0] \oplus K[3]$$
$$\delta[4] = K[0] \oplus K[4]$$
$$\delta[5] = K[0] \oplus K[5]$$
$$\delta[6] = K[0] \oplus K[6]$$

Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Reducing the Number of Key Candidates to 2^8



$\delta[1] = K[0] \oplus K[1]$

$\delta[2] = K[0] \oplus K[2]$

$\delta[3] = K[0] \oplus K[3]$

$\delta[4] = K[0] \oplus K[4]$

$\delta[5] = K[0] \oplus K[5]$

$\delta[6] = K[0] \oplus K[6]$

$\delta[7] = K[0] \oplus K[7]$

$\cdots$

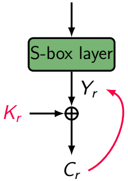
✔ Guess $K[0]$ and determine $K[i]$ for all $i \in \{1, \ldots, 15\}$. So, $|K| = 2^8$ for AES!

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

A Generic Key Recovery Framework

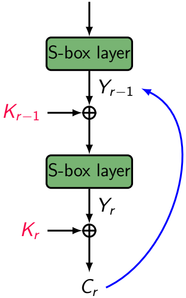
Going Deeper Into the Decryption Rounds

- For each key, compute the impossible values of S-box ($K \Rightarrow V$)
- Go deeper into the decryption to filter more wrong keys
- ⚠ Challenge: the faulty S-box is not invertible
- We use the correct S-box for decryption
- We consider the wrong key assumption



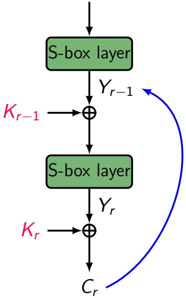
Going Deeper Into the Decryption Rounds

- For each key, compute the impossible values of S-box ($K \Rightarrow V$)
- Go deeper into the decryption to filter more wrong keys
- Challenge: the faulty S-box is not invertible
- We use the correct S-box for decryption
- We consider the wrong key assumption



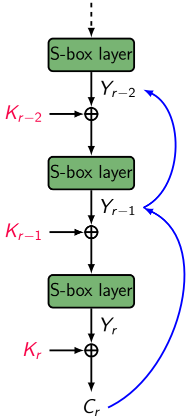
Going Deeper Into the Decryption Rounds

- For each key, compute the impossible values of S-box ($K \Rightarrow V$)
- Go deeper into the decryption to filter more wrong keys
- Challenge: the faulty S-box is not invertible
- We use the correct S-box for decryption
- We consider the wrong key assumption



Going Deeper Into the Decryption Rounds

- For each key, compute the impossible values of S-box ($K \Rightarrow V$)
- Go deeper into the decryption to filter more wrong keys
- ⚠ Challenge: the faulty S-box is not invertible
- We use the correct S-box for decryption
- We consider the wrong key assumption

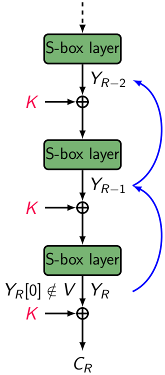


Our Key-recovery Framework

Input: Key candidates

Output: Master key

```
1  for each key candidate K do
2      V ← K[0] ⊕ D[0];
3      cnt[K, V] ← 0;
4      foreach faulty ciphertext do
5          for r = R − 1, ..., 1 do
6              Compute Y_r;
7              foreach cell of Y_r, i.e., Y_r[j] do
8                  if Y_r[j] ∈ V then
9                      Go to line 4
10             cnt[K, V] ← cnt[K, V] + 1;
11 return key with maximum cnt[K, V];
```



$p = \left(1 - \frac{|V|}{256}\right)^{16}$, $\texttt{cnt}_w = N \sum_{r=1}^{R-1} p^r$, $\texttt{cnt}_c = N \sum_{r=1}^{R-1} p^r +$

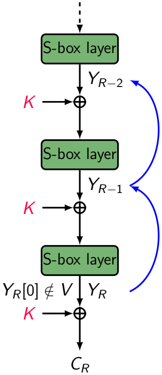
Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Our Key-recovery Framework

Input: Key candidates

Output: Master key

1 **for** *each key candidate* K **do**
2 $\quad V \leftarrow K[0] \oplus D[0]$;
3 $\quad \texttt{cnt}[K, V] \leftarrow 0$;
4 $\quad$ **foreach** *faulty ciphertext* **do**
5 $\quad\quad$ **for** $r = R - 1, \ldots, 1$ **do**
6 $\quad\quad\quad$ Compute Y_r;
7 $\quad\quad\quad$ **foreach** *cell of* Y_r, *i.e.,* $Y_r[j]$ **do**
8 $\quad\quad\quad\quad$ **if** $Y_r[j] \in V$ **then**
9 $\quad\quad\quad\quad\quad$ Go to line 4
10 $\quad\quad\quad$ $\texttt{cnt}[K, V] \leftarrow \texttt{cnt}[K, V] + 1$;
11 **return** key with maximum $\texttt{cnt}[K, V]$;

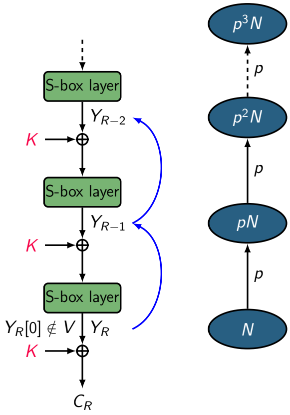


$$p = \left(1 - \frac{|V|}{256}\right)^{16}, \quad \texttt{cnt}_w = N \sum_{r=1}^{R-1} p^r, \quad \texttt{cnt}_c = N \sum_{r=1}^{R-1} p^r +$$

Our Key-recovery Framework

Input: Key candidates
Output: Master key

```
1  for each key candidate K do
2      V ← K[0] ⊕ D[0];
3      cnt[K, V] ← 0;
4      foreach faulty ciphertext do
5          for r = R − 1, . . . , 1 do
6              Compute Y_r;
7              foreach cell of Y_r, i.e., Y_r[j] do
8                  if Y_r[j] ∈ V then
9                      Go to line 4
10             cnt[K, V] ← cnt[K, V] + 1;
11 return key with maximum cnt[K, V];
```



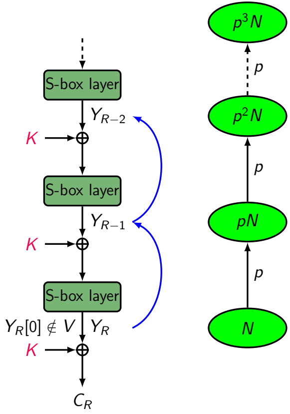
$$p = \left(1 - \frac{|V|}{256}\right)^{16}, \quad \boxed{\texttt{cnt}_w = N \sum_{r=1}^{R-1} p^r}, \quad \texttt{cnt}_c = N \sum_{r=1}^{R-1} p^r +$$

Hadi Soleimany, Nasour Bagheri, **Hosein Hadipour**, Prasanna Ravi, Shivam Bhasin, Sara Mansouri

Our Key-recovery Framework

Input: Key candidates
Output: Master key

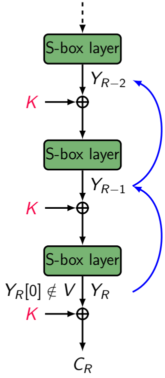
1 **for** *each key candidate* K **do**
2     $V \leftarrow K[0] \oplus D[0]$;
3     $\mathtt{cnt}[K, V] \leftarrow 0$;
4   **foreach** *faulty ciphertext* **do**
5   **for** $r = R - 1, \ldots, 1$ **do**
6   Compute Y_r;
7   **foreach** *cell of* Y_r*, i.e.,* $Y_r[j]$ **do**
8   **if** $Y_r[j] \in V$ **then**
9   Go to line 4
10             $\mathtt{cnt}[K, V] \leftarrow \mathtt{cnt}[K, V] + 1$;
11 **return** key with maximum $\mathtt{cnt}[K, V]$;

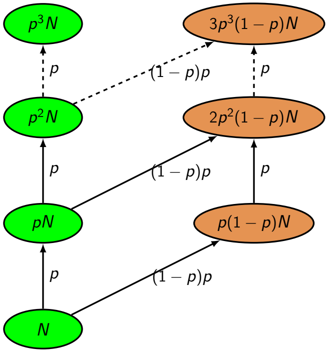


$$p = \left(1 - \frac{|V|}{256}\right)^{16}, \quad \mathtt{cnt}_w = N \sum_{r=1}^{R-1} p^r, \quad \mathtt{cnt}_c = N \sum_{r=1}^{R-1} p^r +$$

Our Key-recovery Framework

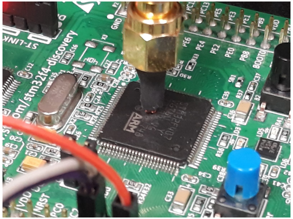
```
    Input: Key candidates
    Output: Master key
1   for each key candidate K do
2       V ← K[0] ⊕ D[0];
3       cnt[K, V] ← 0;
4       foreach faulty ciphertext do
5           for r = R − 1, . . . , 1 do
6               Compute Y_r;
7               foreach cell of Y_r, i.e., Y_r[j] do
8                   if Y_r[j] ∈ V then
9                       Go to line 4
10          cnt[K, V] ← cnt[K, V] + 1;
11  return key with maximum cnt[K, V];
```

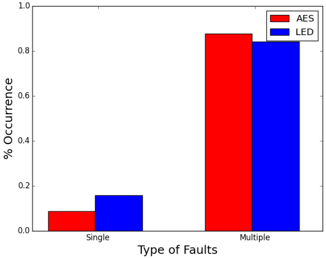




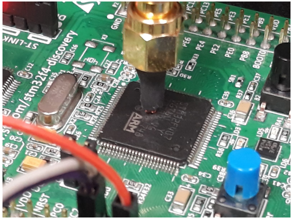
$$p = \left(1 - \frac{|V|}{256}\right)^{16}, \quad \boxed{\texttt{cnt}_w = N\sum_{r=1}^{R-1} p^r}, \quad \texttt{cnt}_c = \boxed{N\sum_{r=1}^{R-1} p^r} + \boxed{N\sum_{r=1}^{R-1} rp^r(1-p)}$$

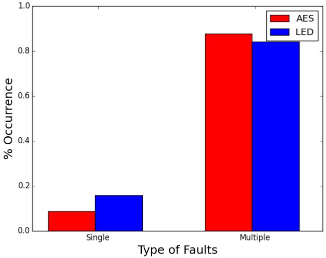
Experimental Verification

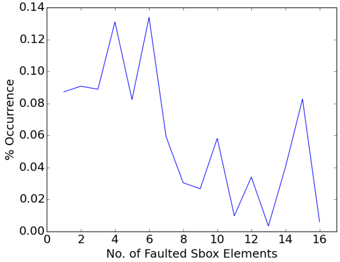




Experimental Verification







$$\lambda = 6,\ N = 1526,\ |K| = 256$$

$$\text{Exp: } \texttt{cnt}_w = 3197.91,\ \texttt{cnt}_c = 6086.93$$

$$\text{The: } \texttt{cnt}_w = 3197.89,\ \texttt{cnt}_c = 6983.73$$

Conclusion

Our Main Contributions

- We removed the assumption of knowing the fault location in PFA
- Our new technique decreases the number of key candidates by a factor of $\approx 2^{50}$
- We exploit the fault leakages in deeper rounds (until the first round)
- Our new technique reduces the number of required ciphertexts (refer to our paper)

Thanks for your attention!

https://github.com/hadipourh/faultyaes

Bibliography I

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. **On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)**. EUROCRYPT 1997. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 37–51. DOI: 10.1007/3-540-69053-0_4.

[Xu+21] Guorui Xu et al. **Pushing the Limit of PFA: Enhanced Persistent Fault Analysis on Block Ciphers**. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 40.6 (2021), pp. 1102–1116. DOI: 10.1109/TCAD.2020.3048280.

[Zha+18] Fan Zhang et al. **Persistent Fault Analysis on Block Ciphers**. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.3 (2018), pp. 150–172. DOI: 10.13154/tches.v2018.i3.150-172.