
Physical Attacks and Countermeasures

lyakhovs@oregonstate.edu

CS 579/ECE 599, February 28, 2022



Oregon State
University

About Me: Stan Lyakhov

- TA for this course
- Graduate student at OSU in Dr. Immler's lab
- Formerly Multiparty Computation (MPC) with Mike Rosulek
- Work primarily on voltage fault injection and side channel analysis
- Fun fact: I grew up next to Technion (where DFA was proposed)

What is Differential Fault Analysis (DFA)?

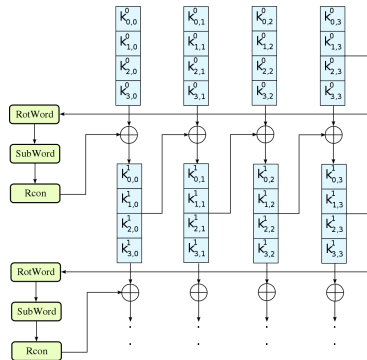
- Goal: extract a secret key by corrupting the state of a cipher
- Introduced for symmetric ciphers by Biham and Shamir in 1997
- Proposed using a laser or “cutting wires”
- Showed attack on DES/3DES (before AES was standardized)

Review: Advanced Encryption Standard

- Three variants (AES128, AES192, AES256) with different key lengths
- Research interest in attacking all three
- Today: focusing on AES128
- AES Visualization:
<https://www.cryptool.org/en/cto/aes-animation>
- AES Step-by-step tool:
<https://www.cryptool.org/en/cto/aes-step-by-step>

AES Key Schedule

- Recall: only the first KeyAdd xors with the key itself
- Key schedule creates subsequent keys using original
- Transformations between keys are:
 - Predicatable
 - **Invertible**
- Can recover original key if we find 10th round key



DFA: Basic Idea

1. Observe output of encryption (no fault)
2. Observe output of an encryption with a strategically placed fault
3. Gain information by looking at the difference/comparing the two

Fault Models: what can an attacker do?

- The fault model dictates what the attacker is capable of
- Consists of two main parts: *location control* and *value control*

Fault Models: what can an attacker do?

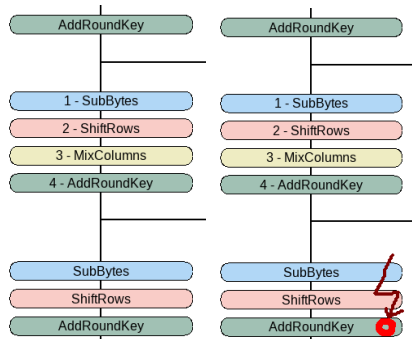
- The fault model dictates what the attacker is capable of
- Consists of two main parts: *location control* and *value control*
- What if we can set a specific value at a specific location?

Fault Models: what can an attacker do?

- The fault model dictates what the attacker is capable of
- Consists of two main parts: *location control* and *value control*
- What if we can set a specific value at a specific location?
 - Target KeyAdd: set some key bit to 0
 - Check if faulty ciphertext changes from non-faulty
 - Need 128 faults (at each bit position)

Simple DFA

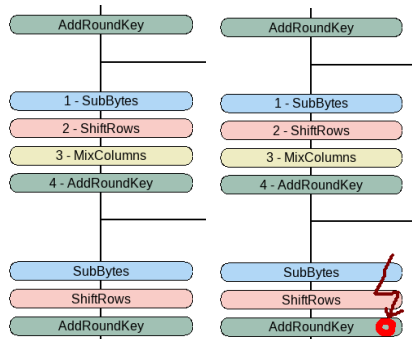
- Inputs: Two similar plaintexts
- For each bit position i of the key:
 - Glitch: Set bit at position i to 0
 - Compare: Are the outputs different?



Simple DFA

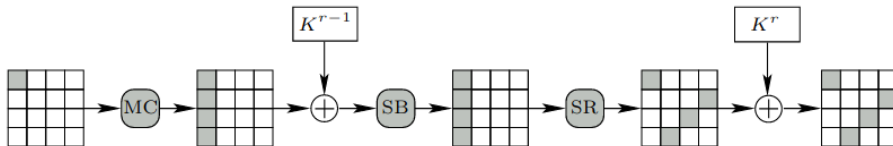
- Inputs: Two similar plaintexts
- For each bit position i of the key:
 - Glitch: Set bit at position i to 0
 - Compare: Are the outputs different?

- Faulty ciphertexts required: 128
- Pros: simple algorithm
- Cons: unreasonable fault model

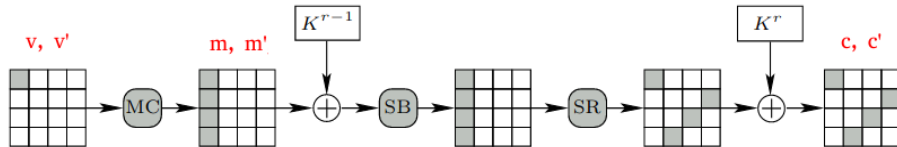


DFA by Piret and Quisquater ('03)

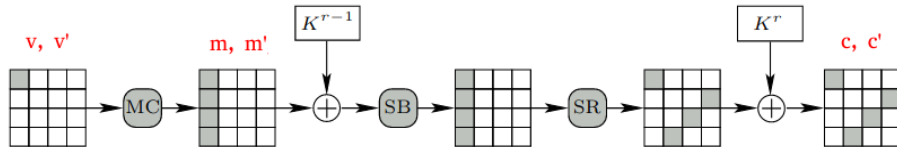
- Fault model: any **single byte** in round 9 changed to any value
- Extract 4 bytes of key (K^r) at a time



DFA by Piret and Quisquater ('03)

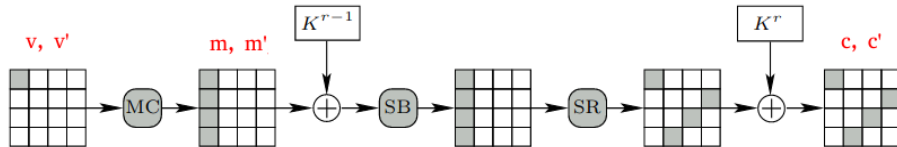


DFA by Piret and Quisquater ('03)



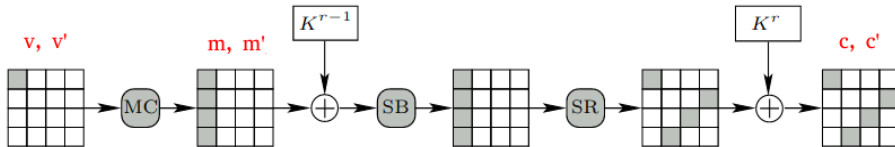
- *Inputs (1 byte apart): $v \oplus \text{glitch} = v'$*
- **MixColumns**

DFA by Piret and Quisquater ('03)



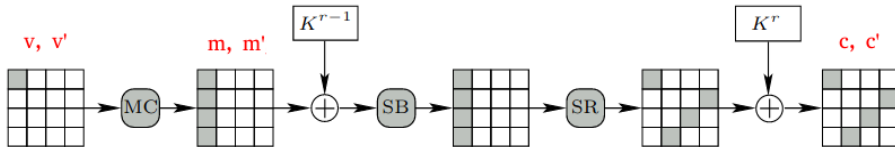
- *Inputs (1 byte apart): $v \oplus \text{glitch} = v'$*
- **MixColumns**
- *Intermediate (unknown): m, m'*

DFA by Piret and Quisquater ('03)



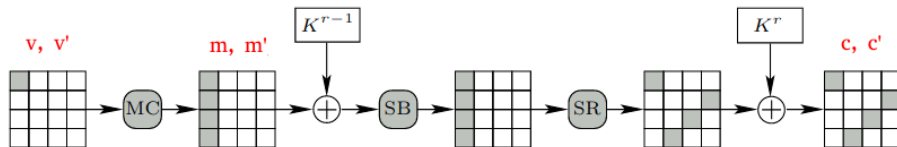
- Inputs (1 byte apart): $v \oplus \text{glitch} = v'$
- **MixColumns**
- Intermediate (unknown): m, m'
- **KeyAdd** K^9
- **SBOX**
- **ShiftRows**
- **KeyAdd** K^{10}

DFA by Piret and Quisquater ('03)



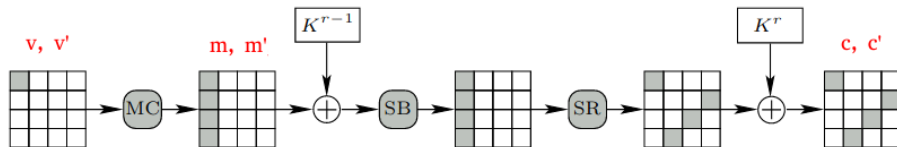
- Inputs (1 byte **apert**): $v \oplus \text{glitch} = v'$
- **MixColumns**
- Intermediate (unknown): m, m'
- **KeyAdd** K^9
- **SBOX**
- **ShiftRows**
- **KeyAdd** K^{10}
- Outputs (known): c, c'

Idea: recover m from c ?



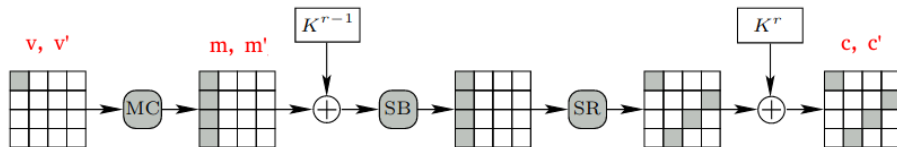
- m (and m') is completely unknown. Can we get from c to m ?

Idea: recover m from c ?



- m (and m') is completely unknown. Can we get from c to m ?
- $m = ISBOX(\text{unshift}(c \oplus K^{10})) \oplus K^9$

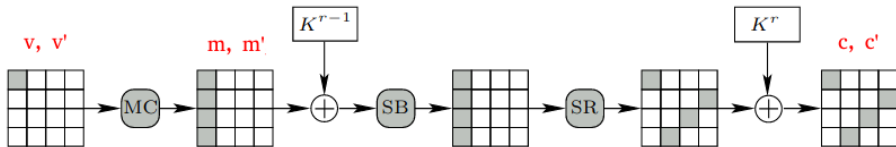
Idea: recover m from c ?



- m (and m') is completely unknown. Can we get from c to m ?
- $m = ISBOX(\text{unshift}(c \oplus K^{10})) \oplus K^9$
- $m' = ISBOX(\text{unshift}(c' \oplus K^{10})) \oplus K^9$

$$\begin{aligned}
 m \oplus m' &= \text{mixcol}(v) \oplus \text{mixcol}(v') \\
 &= \text{mixcol}(v) \oplus \text{mixcol}(v \oplus \text{glitch}) \\
 &= \text{mixcol}(v) \oplus \text{mixcol}(v) \oplus \text{mixcol}(\text{glitch}) \\
 &= \text{mixcol}(\text{glitch})
 \end{aligned}$$

Idea: recover m from c ?



$$\text{mixcol}(\text{glitch}) = m \oplus m'$$


$$= \text{ISBOX}(\text{unshift}(c' \oplus K^{10})) \oplus K^9 \oplus \text{ISBOX}(\text{unshift}(c \oplus K^{10})) \oplus K^9$$

$$= \text{ISBOX}(\text{unshift}(c' \oplus K^{10})) \oplus \text{ISBOX}(\text{unshift}(c \oplus K^{10}))$$

Computing all possibilities of `mixcol(glitch)`

- Let $x = 1$ to 255
- Note:** all multiplication is galois field multiplication!

$$\text{glitch} = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{mixcol} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

$$m \oplus m' = \text{mixcol} \times \text{glitch} = \begin{bmatrix} 2 \times x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ 3 \times x & 0 & 0 & 0 \end{bmatrix}$$


Only 255 possibilities for `mixcol(glitch)`! Can precompute

Plan of attack

- Guess K^{10}
- $m \oplus m' = ISBOX(\text{unshift}(c' \oplus K^{10})) \oplus ISBOX(\text{unshift}(c \oplus K^{10}))$
- If $m \oplus m'$ does not equal to $\text{mixcol}(\text{glitch})$ for some x
- K^{10} cannot be the correct key!

$$m \oplus m' = \text{mixcol}(\text{glitch}) = \begin{bmatrix} 2 \times x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ 3 \times x & 0 & 0 & 0 \end{bmatrix}$$


Plan of attack: Problem

- Guess K^{10}
- ...

Plan of attack: Problem

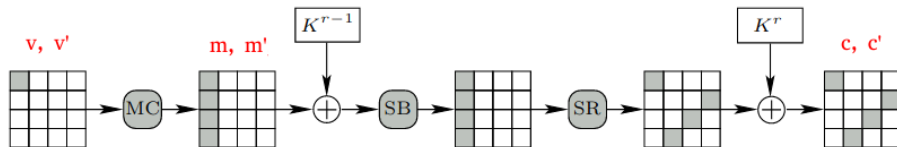
- Guess K^{10}
- ...
- **Problem:** K^{10} is 128 bits long! Are we guessing the whole thing?

Plan of attack: Problem

- Guess K^{10}
- ...
- **Problem:** K^{10} is 128 bits long! Are we guessing the whole thing?
- **Solution:** No, only guess part of K^{10} (the 4 bytes that differ) 

$$\begin{bmatrix} 2 \times x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ x & 0 & 0 & 0 \\ 3 \times x & 0 & 0 & 0 \end{bmatrix}$$

Attacking Partial K^{10}



- $m_0 \oplus m'_0 = ISBOX(c_0 \oplus K_0) \oplus ISBOX(c'_0 \oplus K_0)$
- $m_1 \oplus m'_1 = ISBOX(c_{13} \oplus K_{13}) \oplus ISBOX(c'_{13} \oplus K_{13})$
- $m_2 \oplus m'_2 = ISBOX(c_{10} \oplus K_{10}) \oplus ISBOX(c'_{10} \oplus K_{10})$
- $m_3 \oplus m'_3 = ISBOX(c_7 \oplus K_7) \oplus ISBOX(c'_7 \oplus K_7)$

Revised Plan of Attack

- Guess K_0, K_{13}, K_{10}, K_7
- Let $m_c \oplus m'_c = [m_0 \oplus m'_0, m_{13} \oplus m'_{13}, m_{10} \oplus m'_{10}, m_7 \oplus m'_7]^T$
- If $m_c \oplus m'_c$ does not equal to **first column** of $\text{mixcol}(\text{glitch})$ for some x
- K_0, K_{13}, K_{10}, K_7 can't all be the correct bytes!

$$\text{mixcol}(\text{glitch})[0] = \begin{bmatrix} 2 \times x \\ x \\ x \\ 3 \times x \end{bmatrix}$$

- We started with 2^{32} possible bytes
- After filtering impossible values, we are left with around $2^{10} \approx 1036$ possibilities

Revised Plan of Attack

- Guess K_0, K_{13}, K_{10}, K_7
- Let $m_c \oplus m'_c = [m_0 \oplus m'_0, m_{13} \oplus m'_{13}, m_{10} \oplus m'_{10}, m_7 \oplus m'_7]^T$
- If $m_c \oplus m'_c$ does not equal to **first column** of $\text{mixcol}(\text{glitch})$ for some x
- K_0, K_{13}, K_{10}, K_7 can't all be the correct bytes!

$$\text{mixcol}(\text{glitch})[0] = \begin{bmatrix} 2 \times x \\ x \\ x \\ 3 \times x \end{bmatrix}$$

- We started with 2^{32} possible bytes
- After filtering impossible values, we are left with around $2^{10} \approx 1036$ possibilities
- We can narrow those down to just 1 (more on that later)

Analysis: Revised Plan of Attack

- Given c and c' , we can remove from consideration most values of 4 impacted bytes
- Have to brute force $\{K_0, K_{13}, K_{10}, K_7\}$, which means $(2^8)^4 = 2^{32}$ possibilities
- Not impossible, but can we do better?

Analysis: Revised Plan of Attack

- Given c and c' , we can remove from consideration most values of 4 impacted bytes
- Have to brute force $\{K_0, K_{13}, K_{10}, K_7\}$, which means $(2^8)^4 = 2^{32}$ possibilities
- Not impossible, but can we do better?
- Yes! Filter 2 bytes, then 3 bytes, and then 4 bytes at the end

Simple Attack Outline

Simple Attack: Filtering

- Let $m_{c2} = [m_0 \oplus m'_0, m_{13} \oplus m'_{13}]^T$. Iterations: 256×256
- Let $m_{c3} = [\text{filtered}(m_{c2}), m_{10} \oplus m'_{10}]^T$. Iterations: $|\text{filtered}(m_{c2})| \times 256$
- Let $m_{c4} = [\text{filtered}(m_{c3}), m_7 \oplus m'_7]^T$. Iterations: $|\text{filtered}(m_{c3})| \times 256$
- Our key candidates $\{K_0, K_{13}, K_{10}, K_7\}_i$ are the keybytes used in $\text{filtered}(m_{c4})$

Filter defined as:

$$m_{c2} \stackrel{?}{=} [2 \times x, x]^T$$

$$m_{c3} \stackrel{?}{=} [2 \times x, x, x]^T$$

$$m_{c4} \stackrel{?}{=} [2 \times x, x, x, 3 \times x]^T$$

Simple Attack: End

- After filtering, we are left with $|\text{filtered}(m_{c4})| \approx 1036$ possibilities
- Need one more faulty pair: $p^* \implies (c^*, c^{*'})$
- For the remaining candidates, check that $m^* \oplus m^{*'} is in $[2 \times x, x, x, 3 \times x]^T$ as well$

Simple Attack: End

- After filtering, we are left with $|\text{filtered}(m_{c4})| \approx 1036$ possibilities
- Need one more faulty pair: $p^* \implies (c^*, c^{*'})$
- For the remaining candidates, check that $m^* \oplus m^{*'}$ is in $[2 \times x, x, x, 3 \times x]^T$ as well

For each remaining key candidates $\{K_0, K_{13}, K_{10}, K_7\} \in \text{filtered}(m_{c4})$ and $x \in 1$ to 255:

Simple Attack: End

- After filtering, we are left with $|\text{filtered}(m_{c4})| \approx 1036$ possibilities
- Need one more faulty pair: $p^* \implies (c^*, c^{*'})$
- For the remaining candidates, check that $m^* \oplus m^{*'} is in $[2 \times x, x, x, 3 \times x]^T$ as well$

For each remaining key candidates $\{K_0, K_{13}, K_{10}, K_7\} \in \text{filtered}(m_{c4})$ and $x \in 1$ to 255:

$$ISBOX(c_0^* \oplus K_0) \oplus ISBOX(c_0^{*'} \oplus K_0) \stackrel{?}{=} 2 \times x$$

$$ISBOX(c_{13}^* \oplus K_{13}) \oplus ISBOX(c_{13}^{*'} \oplus K_{13}) \stackrel{?}{=} x$$

$$ISBOX(c_{10}^* \oplus K_{10}) \oplus ISBOX(c_{10}^{*'} \oplus K_{10}) \stackrel{?}{=} x$$

$$ISBOX(c_7^* \oplus K_7) \oplus ISBOX(c_7^{*'} \oplus K_7) \stackrel{?}{=} 3 \times x$$

Simple Attack: End

- After filtering, we are left with $|\text{filtered}(m_{c4})| \approx 1036$ possibilities
- Need one more faulty pair: $p^* \implies (c^*, c^{*'})$
- For the remaining candidates, check that $m^* \oplus m^{*'} is in $[2 \times x, x, x, 3 \times x]^T$ as well$

For each remaining key candidates $\{K_0, K_{13}, K_{10}, K_7\} \in \text{filtered}(m_{c4})$ and $x \in 1$ to 255:

$$ISBOX(c_0^* \oplus K_0) \oplus ISBOX(c_0^{*'} \oplus K_0) \stackrel{?}{=} 2 \times x$$

$$ISBOX(c_{13}^* \oplus K_{13}) \oplus ISBOX(c_{13}^{*'} \oplus K_{13}) \stackrel{?}{=} x$$

$$ISBOX(c_{10}^* \oplus K_{10}) \oplus ISBOX(c_{10}^{*'} \oplus K_{10}) \stackrel{?}{=} x$$

$$ISBOX(c_7^* \oplus K_7) \oplus ISBOX(c_7^{*'} \oplus K_7) \stackrel{?}{=} 3 \times x$$

$\approx 98\%$ of the time, the equations will hold for just a single candidate

Simple Attack Algorithm

Simple Attack Algorithm: Preparation

1. Compute $D = \text{mixcol}(\text{glitch}) = [2 \times x, x, x, 3 \times x]^T$ for every $x \in 1$ to 255
2. Encrypt some plaintext p and observe the output c
3. Inject fault in the first byte in the 9th round to get c'
4. Encrypt another plaintext p^* and observe the output c^*
5. Inject another fault in the same place to get $c^{*'}$

Simple Attack Algorithm: Filtering

1. For each $K_0 \in 0$ to 255 and $K_{13} \in 0$ to 255, and $x \in 1$ to 255
 - 1.1 $ISBOX(K_0 \oplus c_0) \oplus ISBOX(K_0 \oplus c'_0) \stackrel{?}{=} D[x, 0]$
 - 1.2 $ISBOX(K_{13} \oplus c_{13}) \oplus ISBOX(K_{13} \oplus c'_{13}) \stackrel{?}{=} D[x, 1]$
 - 1.3 If both are true, add candidates $\{K_0, K_{13}\}$ to group K_{c2}
2. For each $K_{10} \in 0$ to 255, and $\{K_0, K_{13}\} \in K_{c2}$, and $x \in 1$ to 255
 - 2.1 $ISBOX(K_0 \oplus c_0) \oplus ISBOX(K_0 \oplus c'_0) \stackrel{?}{=} D[x, 0]$
 - 2.2 $ISBOX(K_{13} \oplus c_{13}) \oplus ISBOX(K_{13} \oplus c'_{13}) \stackrel{?}{=} D[x, 1]$
 - 2.3 $ISBOX(K_{10} \oplus c_{10}) \oplus ISBOX(K_{10} \oplus c'_{10}) \stackrel{?}{=} D[x, 2]$
 - 2.4 If so, add candidates $\{K_0, K_{13}, K_{10}\}$ to group K_{c3}
3. For each $K_7 \in 0$ to 255, and $\{K_0, K_{13}, K_{10}\} \in K_{c3}$, and $x \in 1$ to 255
 - 3.1 $ISBOX(K_0 \oplus c_0) \oplus ISBOX(K_0 \oplus c'_0) \stackrel{?}{=} D[x, 0]$
 - 3.2 $ISBOX(K_{13} \oplus c_{13}) \oplus ISBOX(K_{13} \oplus c'_{13}) \stackrel{?}{=} D[x, 1]$
 - 3.3 $ISBOX(K_{10} \oplus c_{10}) \oplus ISBOX(K_{10} \oplus c'_{10}) \stackrel{?}{=} D[x, 2]$
 - 3.4 $ISBOX(K_7 \oplus c_7) \oplus ISBOX(K_7 \oplus c'_7) \stackrel{?}{=} D[x, 3]$
 - 3.5 If so, add candidates $\{K_0, K_{13}, K_{10}, K_7\}$ to group K_{c4}

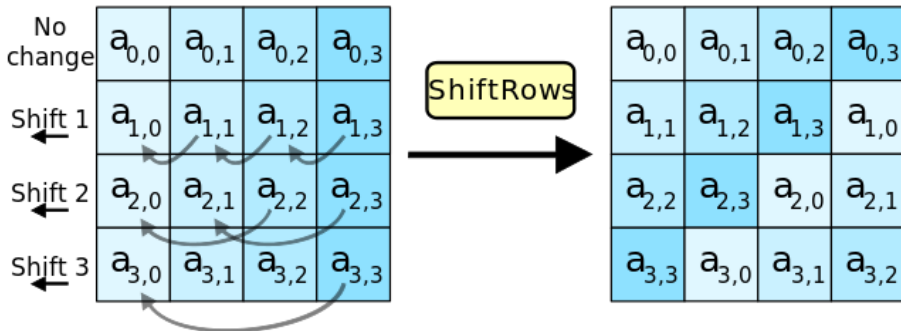
Simple Attack Algorithm: End

1. For each $\{K_0, K_{13}, K_{10}, K_7\} \in K_{c4}$, and $x \in 1$ to 255
 - 1.1 $ISBOX(K_0 \oplus c_0^*) \oplus ISBOX(K_0 \oplus c_0^{*'}) \stackrel{?}{=} D[x, 0]$
 - 1.2 $ISBOX(K_{13} \oplus c_{13}^*) \oplus ISBOX(K_{13} \oplus c_{13}^{*'}) \stackrel{?}{=} D[x, 1]$
 - 1.3 $ISBOX(K_{10} \oplus c_{10}^*) \oplus ISBOX(K_{10} \oplus c_{10}^{*'}) \stackrel{?}{=} D[x, 2]$
 - 1.4 $ISBOX(K_7 \oplus c_7^*) \oplus ISBOX(K_7 \oplus c_7^{*'}) \stackrel{?}{=} D[x, 3]$
 - 1.5 If all are true $\{K_0, K_{13}, K_{10}, K_7\}$, are likely correct keybytes

Converting Simple Attack to Full Attack

Limitations of Simple Attack: leaks 4 out of 16 bytes

- We only showed how to attack the first column to leak 4 bytes.
- Same strategy applies for the rest!
 - We need to inject the glitch in a different place (e.g. in the first entry of the second column)
 - Instead of working with $\{K_0, K_{13}, K_{10}, K_7\}$ we use something else
 - For column 2: $\{K_4, K_1, K_{14}, K_{11}\}$. Can you figure out the rest?



Limitations of Simple Attack: only works on first row glitch

- What if we can't control exactly what single byte is glitched?
 - We can tell which column was affected by comparing c and c'
 - Modify D to include possibilities that any of the 4 bytes in the column can be glitched
 - Size of D : $255 \times 4 = 1020$. Still easy to precompute.

$$\text{glitch}_0 = [1 \text{ to } 255 \quad 0 \quad 0 \quad 0]^T$$

$$\text{glitch}_1 = [0 \quad 1 \text{ to } 255 \quad 0 \quad 0]^T$$

$$\text{glitch}_2 = [0 \quad 0 \quad 1 \text{ to } 255 \quad 0]^T$$

$$\text{glitch}_3 = [0 \quad 0 \quad 0 \quad 1 \text{ to } 255]^T$$

$$D = (\text{mixcol} \times \text{glitch}_0) \cup (\text{mixcol} \times \text{glitch}_1) \\ \cup (\text{mixcol} \times \text{glitch}_2) \cup (\text{mixcol} \times \text{glitch}_3)$$

Last Touches

- Let's say we did this attack on all 4 columns and got all 16 keybytes
- Now we are done right?

Last Touches

- Let's say we did this attack on all 4 columns and got all 16 keybytes
- Now we are done right?



Last Touches

- We recovered the round 10 key. We want the original key

Last Touches

- We recovered the round 10 key. We want the original key
- Recall: AES key schedule is invertible
- Can find original round key based on round 10 key

Summary: DFA by Piret and Quisquater ('03)

- Need 2 different faults in the same column to recover 4 bytes
- 8 faults total for the entire key
- Can we do it with 1 fault per column instead?
 - We are left with about 2^{10} values per each 4 bytes
 - Guessing $(2^{10})^4 = 2^{40}$ is possible but a little painful if you don't have strong computer
 - Still powerful: going from 128 bit security to 40 bit security

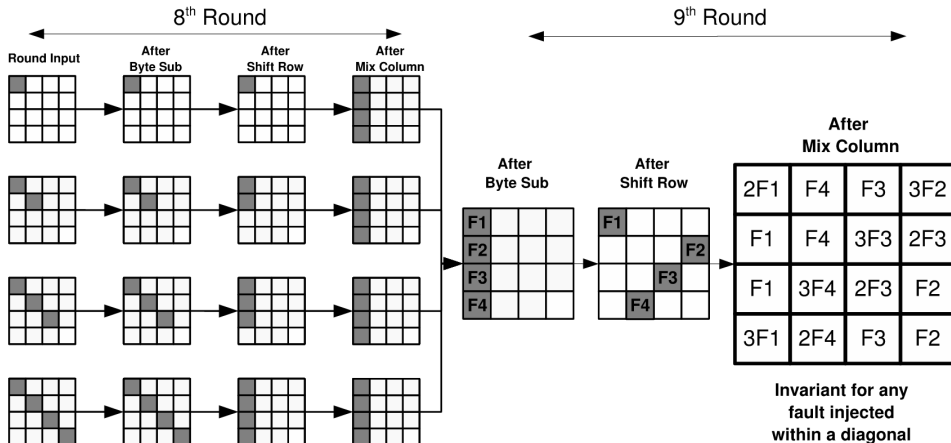
DFA by Saha et al. ('09)

- 1 fault to rule them all: recover key with 1 fault
- Fault injected in 8th round

DFA by Saha et al. ('09)

- 1 fault to rule them all: recover key with 1 fault
- Fault injected in 8th round
- Pro: 1 fault is all you need!
- Con: Must bruteforce 2^{32} keys at the end

DFA by Saha et al. ('09)



References

- Biham and Shamir
<https://link.springer.com/chapter/10.1007/BFb0052259>
- Piret and Quisquater
<https://eprint.iacr.org/2010/440>
- Saha et al.
<https://eprint.iacr.org/2009/581.pdf>