

Implementation Security In Cryptography

Lecture 11: Fault Attacks

Recap

- Till the last lecture
 - Masking...

Today

- Fault Attacks

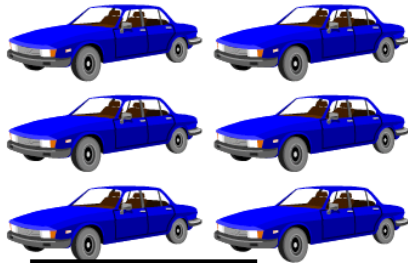
Faults in Our Life...

- Happens....
- We are all human being (Hope there is no AI agent in my class) :P
- We all learn from our faults....
 - The learning is what I am looking for here

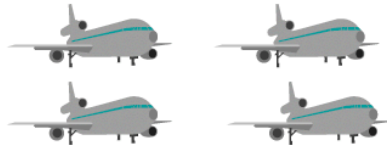
Faults in Our Life...

WHAT IS THIS ABOUT?

Broken toys are not charged to our clients



car = \$3



plane = \$5

Jack

How will you pay?

I'll send \$15
by postal order

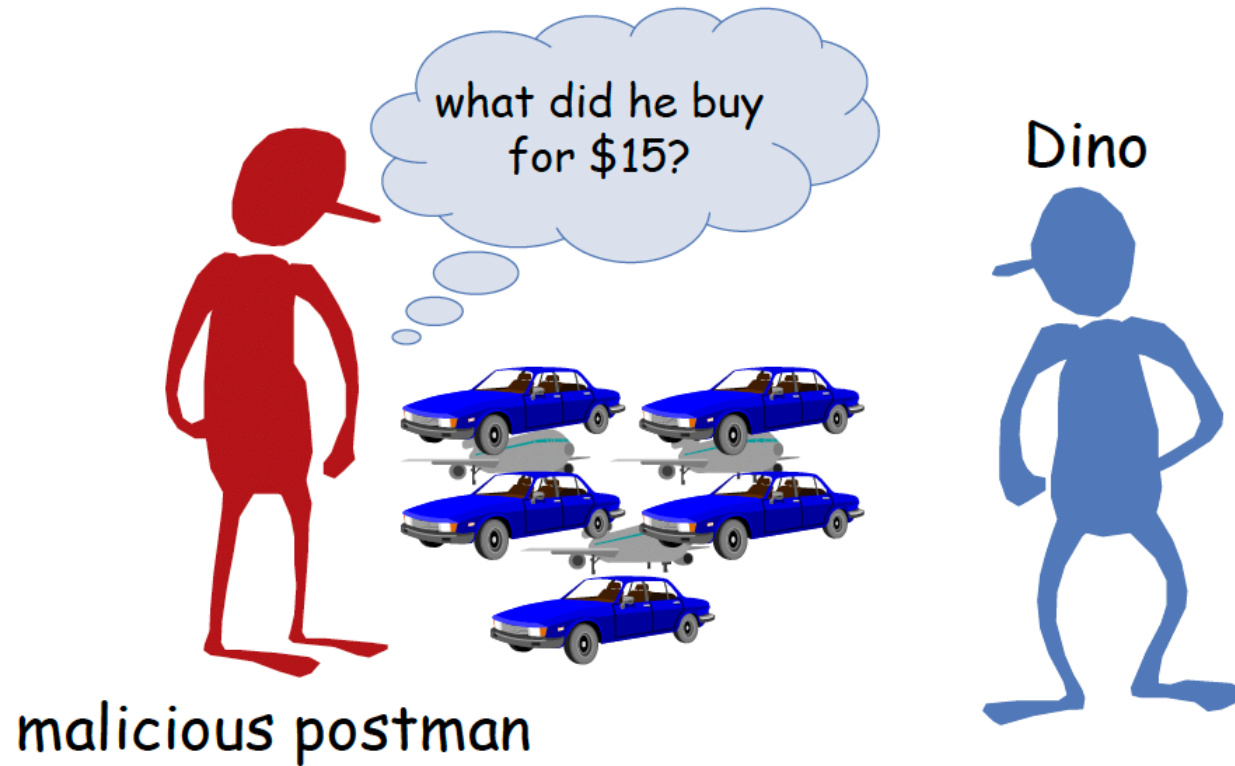
Dino

Dino buys toys from Jack

- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

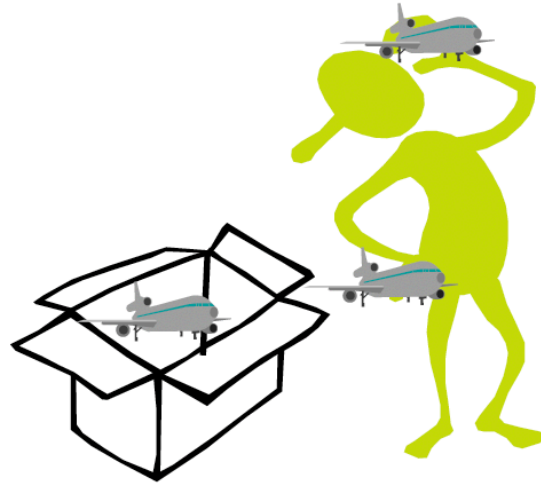
*The postman wants to know
what Dino bought for \$15*



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

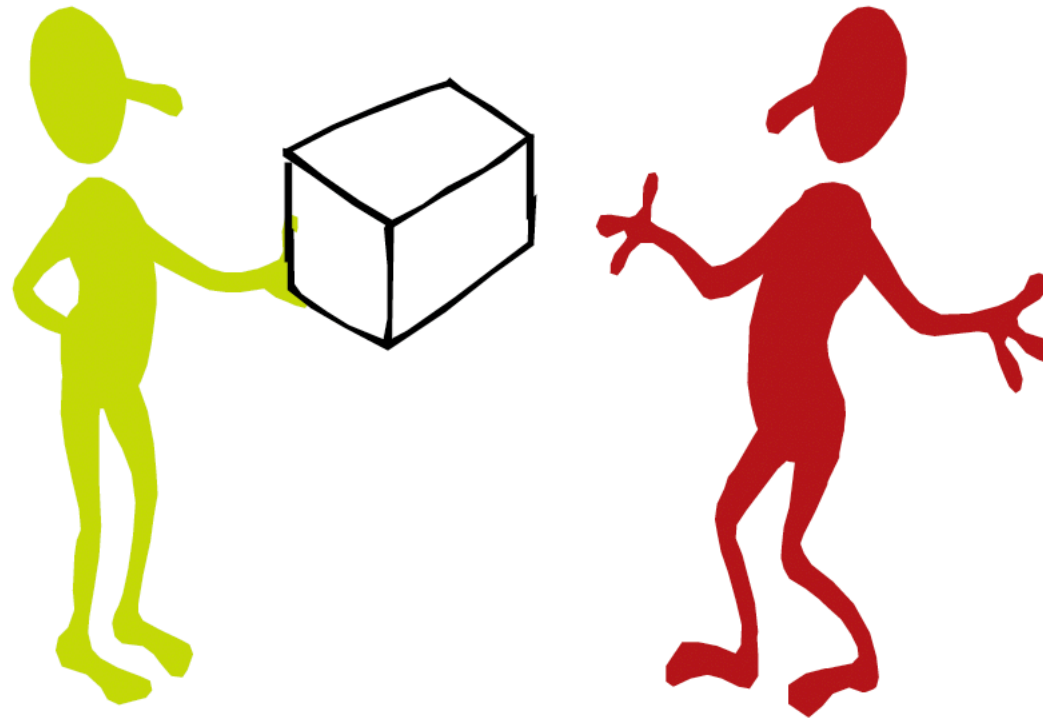
*In the meanwhile Jack prepares
the DHL*



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

and gives it to the postman



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

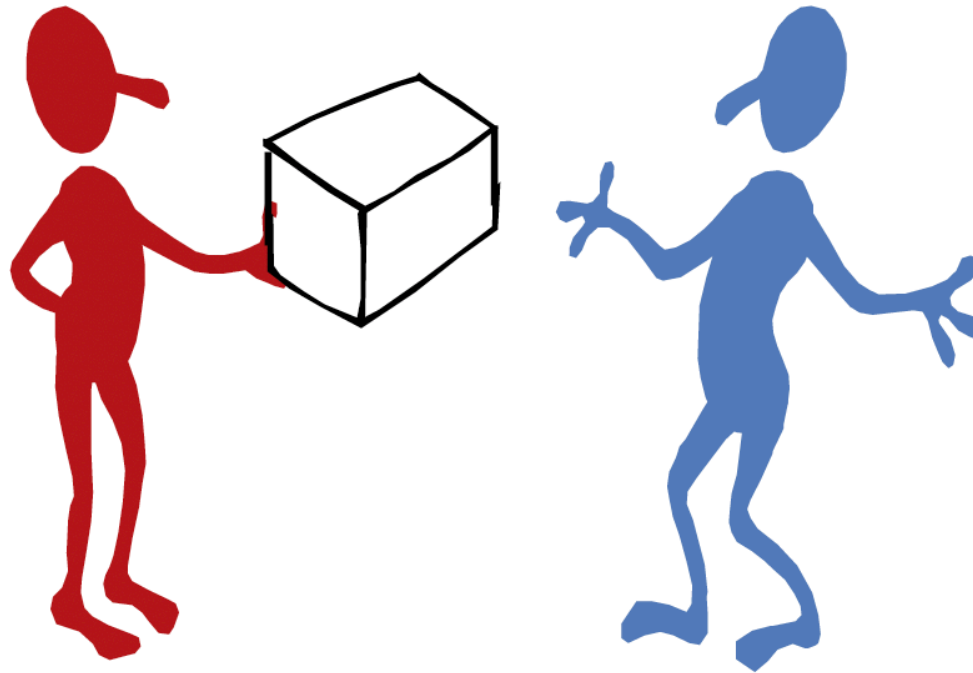
*Who kicks it strong enough to
break one toy*



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

and gives it to Dino



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Life...

a week later he monitors Dino's postal order...



Lesson learned: **Fault attacks** can also extract secrets from tokens!

Hardware faults can have various sources:
voltage glitches, light beams, laser beams...

- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Faults in Our Crypto...

- Introduction of faults in the normal execution of cryptographic algorithms and analysis of faulty output to obtain the key
- First conceived in 1996 by Boneh, Demillo and Lipton
- E. Biham developed Differential Fault Analysis (DFA) of DES
- Today there are numerous examples of fault analysis of block ciphers such as AES under a variety of fault models and fault injection techniques
- Popular Fault Injection Techniques – Clock Glitches, Voltage Glitches, EM and Optical Injection Techniques

What Faults are Up To?

Serious Security: Rowhammer returns to gaslight your computer

Gaslights produce a telltale flicker when nearby lamps are lit; DRAM values do something similar when nearby memory cells are accessed.

Written by Paul Ducklin

JULY 10, 2023

NAKED SECURITY

DATA LEAKAGE

ROWHAMMER

SERIOUS SECURITY

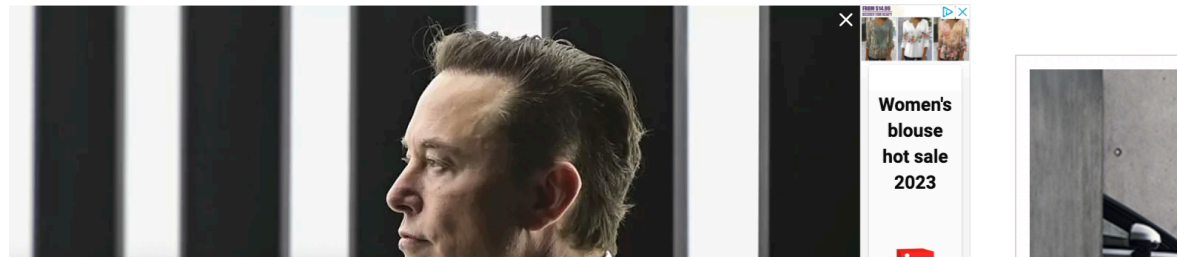
You're probably familiar with the word *gaslighting*, used to refer to people with the odious habit of lying not merely to cover up their own wrongdoing, but also to make it look as though someone else is at fault, even to the point of getting the other person to doubt their own memory, decency and sanity.

You might not know, however, that the term comes from a 1930s psychological thriller play called [Gas Light](#) (spoiler alert) in which a manipulative and murderous husband pretends to spend his evenings out on the town

Technology

Using just a \$25 device a researcher hacked into Elon Musk's Starlink system

What will the technology mogul have to say about this?



Faults in Our Crypto...

Small World

Voltage-Glitch



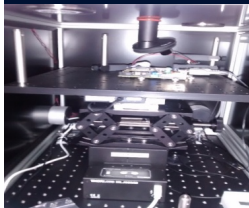
EM-FI



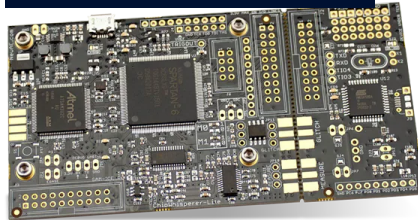
Clock-Glitch



Laser-FI



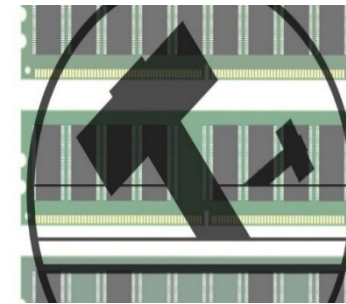
Chipwhisperer



- Tight control on injection timing
 - You can hit the variable you want at a specific instant
- Multi-bit — but can be made single-bit too
- Biased fault distribution
- Mainly transient
- Multiple injection challenging

Big World

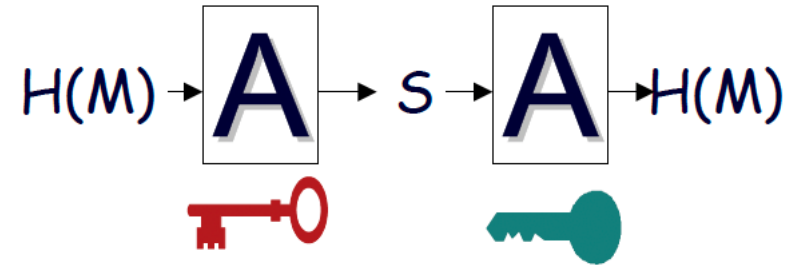
Row-Hammer



- Timing can be controlled but precision is less
 - Target variable must be in RAM for quite sometime
- Single-bit — fairly well controllable
- Mainly persistent
- Multiple injection is easier

A Break From AES...RSA Signature

- Let's talk RSA
- Public key algorithm, — Encryption and Signature
 - We talk about signature.
- **Simple Idea:**
 - Alice generates a secret and a public key.
 - Gives the public key to “Public”
 - Signs a message M with secret key and generates signature C
 - Everyone with the public key can verify that:
 - The C is a valid signature of M
 - C is generated by Alice only and nobody else



RSA Signature in Brief...

- Alice generates two large primes p, q and computes $N = pq$
- It also finds e and d , where $ed \equiv 1 \pmod{\phi(N)}$, this $\phi(N)$ is called Euler's totient function and $\phi(N) = (p - 1)(q - 1)$, as p, q are primes.
- p, q, d is secret key
- N, e is the public key
- **Sign:** $m \in \mathbb{Z}_N^*$, compute $s = m^d \pmod{N}$, and returns $s || m$
- **Verify:** Check if $m' = m$, where $m' = s^e \pmod{N}$
- I have omitted some crucial details here for simplicity. But that's not for what we are going to explain..
 - E.g, m is not the message but is a hash of the message..also there are some paddings needed for security...

RSA-CRT

- CRT stands for **Chinese Remainder Theorem** — check it out!!!
- RSA-CRT is a performance optimisation trick...

$$\left\{ \begin{array}{l} a \equiv 1 \pmod{p} \\ a \equiv 0 \pmod{q} \end{array} \right. \text{ and } \left\{ \begin{array}{l} b \equiv 0 \pmod{p} \\ b \equiv 1 \pmod{q} \end{array} \right. \quad \begin{array}{l} d_p = d \pmod{p-1} \\ d_q = d \pmod{q-1} \end{array} \quad \begin{array}{l} s_p = m^{d_p} \pmod{p} \\ s_q = m^{d_q} \pmod{q} \end{array}$$

$$s = a \times s_p + b \times s_q \pmod{N}$$

Attacking RSA-CRT...

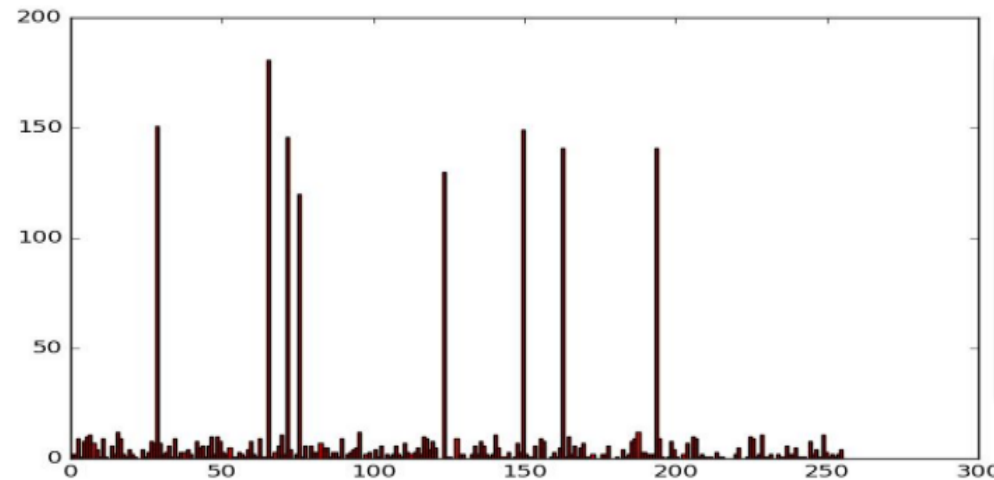
- Again, we want to find the secret key...So attack the signing process..
- Let's say, for a message m , I can repeat the signature generation process.
- What I do:
 - Generate the correct signature $s = \text{sign}(m, d)$
 - On the same message generate a faulty signature $\hat{s} = \text{sign}(m, d)$
 - Fault happens during the computation of the sign.
 - The fault only corrupts the computation of s_p or s_q , but not both

Attacking RSA-CRT...

- Let $\hat{s} = a \times s_p + b \times \hat{s}_q \text{ mod } N$
- Let $s = a \times s_p + b \times s_q \text{ mod } N$
- Let $\Delta = s - \hat{s} = b(s_q - \hat{s}_q)$
- Now, observe that, since $b \equiv 0 \text{ mod } p$
 - $\Delta = b(s_q - \hat{s}_q)$ is divisible by p
 - So, $\Delta = kp$, and $\gcd(s - \hat{s}, n) = p$
- Ok, you got p , you know N , so you know q — game over!!!
- Attack is also possible if you do not use CRT — analysis is slightly different

Let There Be Faults: Fault Model

- Key component of a fault attack
- Attack procedure changes according to the fault model
- Random localized faults
 - Bit/nibble/byte fault
 - Most general model
- Biased faults
 - Device-dependent model
- Instruction skip/modify.
- Constant fault
 - Stuck-at-0/1
- Persistent fault



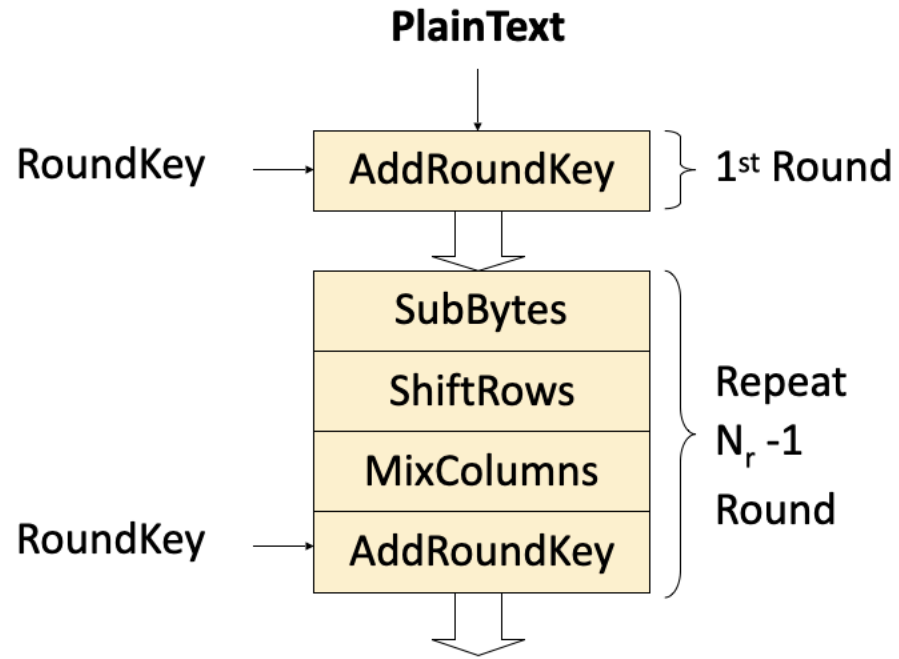
In a fault space of size 256, only 8 faults occur in 98% of the total injections!!!

```
ldi r1 0;  
ld r1 #M1  
ldi r2 0  
ld r2 #M2  
add r1 r2  
str r1
```

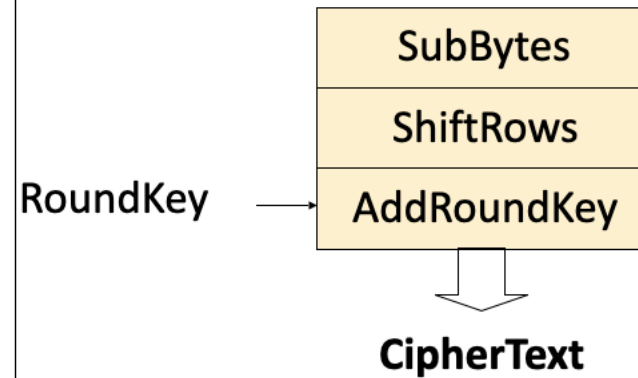


```
ldi r1 0;  
nop  
ldi r2 0  
ld r2 #M2  
add r1 r2  
str r1
```

The AES Story...



First 9 Rounds



Last Round

Looking Inside AES Once Again

State

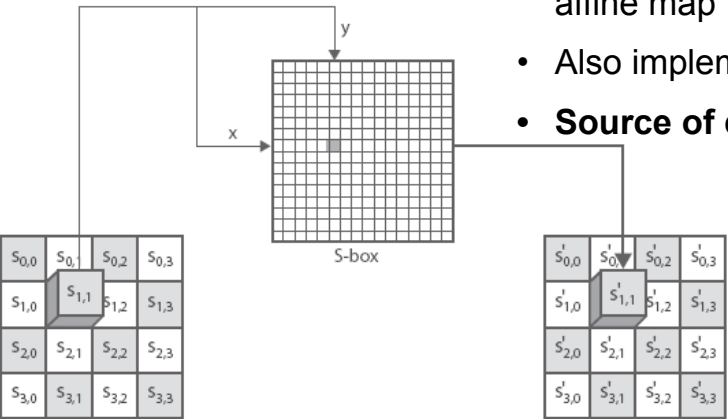
1 byte

s00	s01	s02	s03
s10	s11	s12	s13
s20	s21	s22	s23
s20	s31	s32	s33

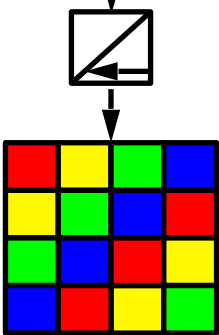
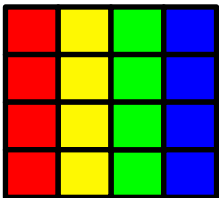
k00	k01	k02	k03
k10	k11	k12	k13
k20	k21	k22	k23
k20	k31	k32	k33

1. SubBytes

- Nonlinear Boolean Function
- Finite field inversion followed by affine map
- Also implemented as a table
- **Source of confusion**

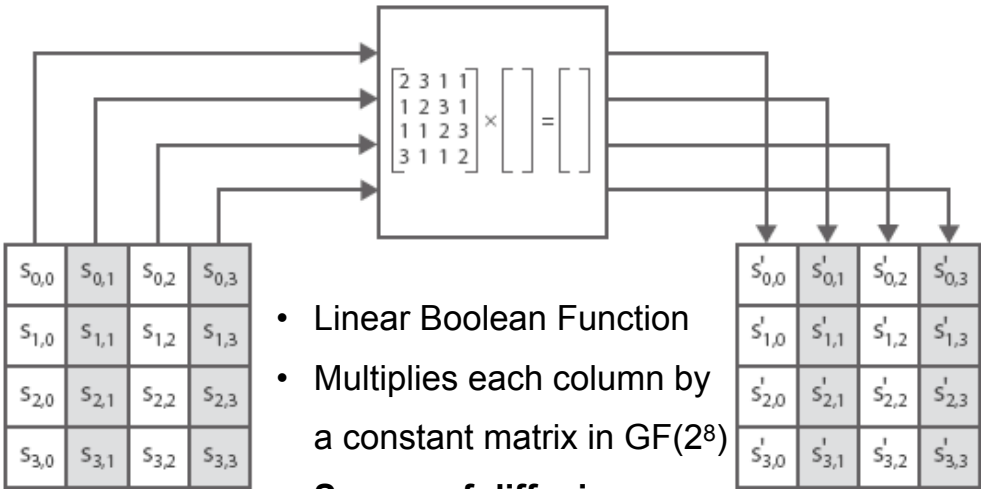


2. ShiftRows



- Linear Boolean Function
- Left circular shift of rows
- **Source of diffusion**

3. MixColumns



- Linear Boolean Function
- Multiplies each column by a constant matrix in $GF(2^8)$
- **Source of diffusion**

4. AddRoundKey

s00	s01	s02	s03
s10	s11	s12	s13
s20	s21	s22	s23
s20	s31	s32	s33

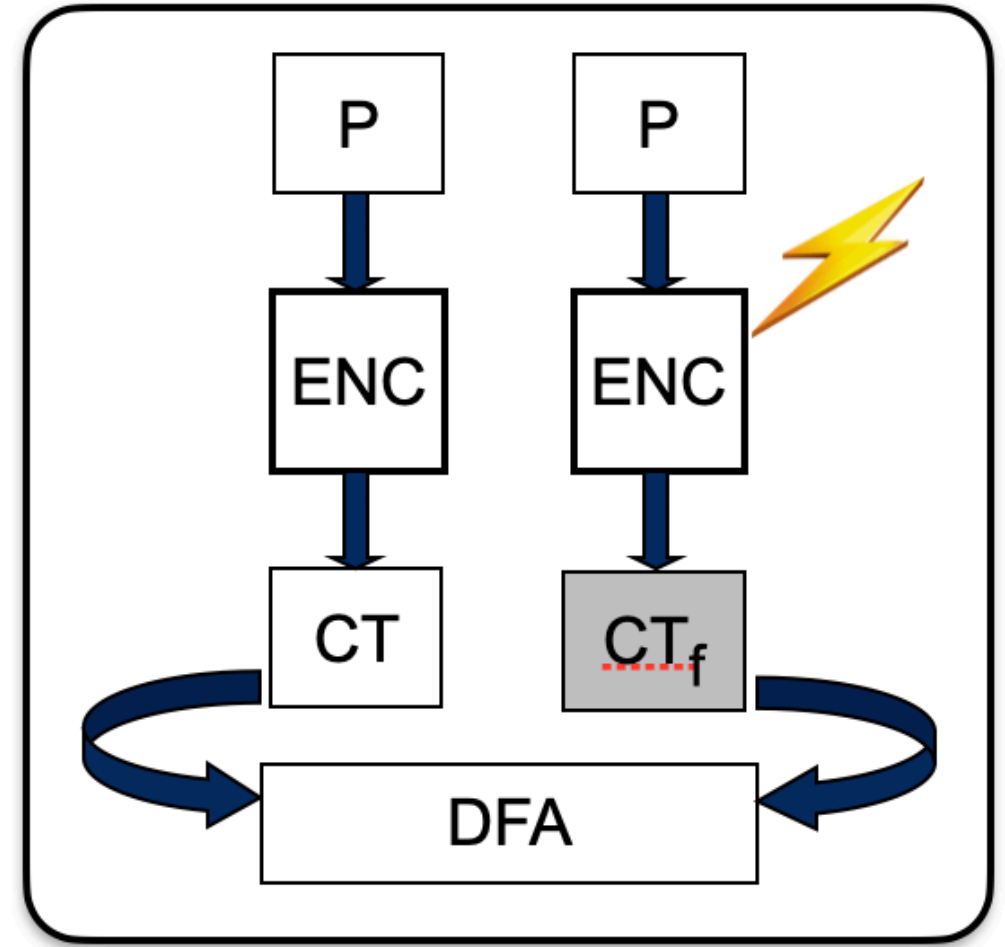


k00	k01	k02	k03
k10	k11	k12	k13
k20	k21	k22	k23
k20	k31	k32	k33

- Linear Boolean Function
- XOR the state with a round key

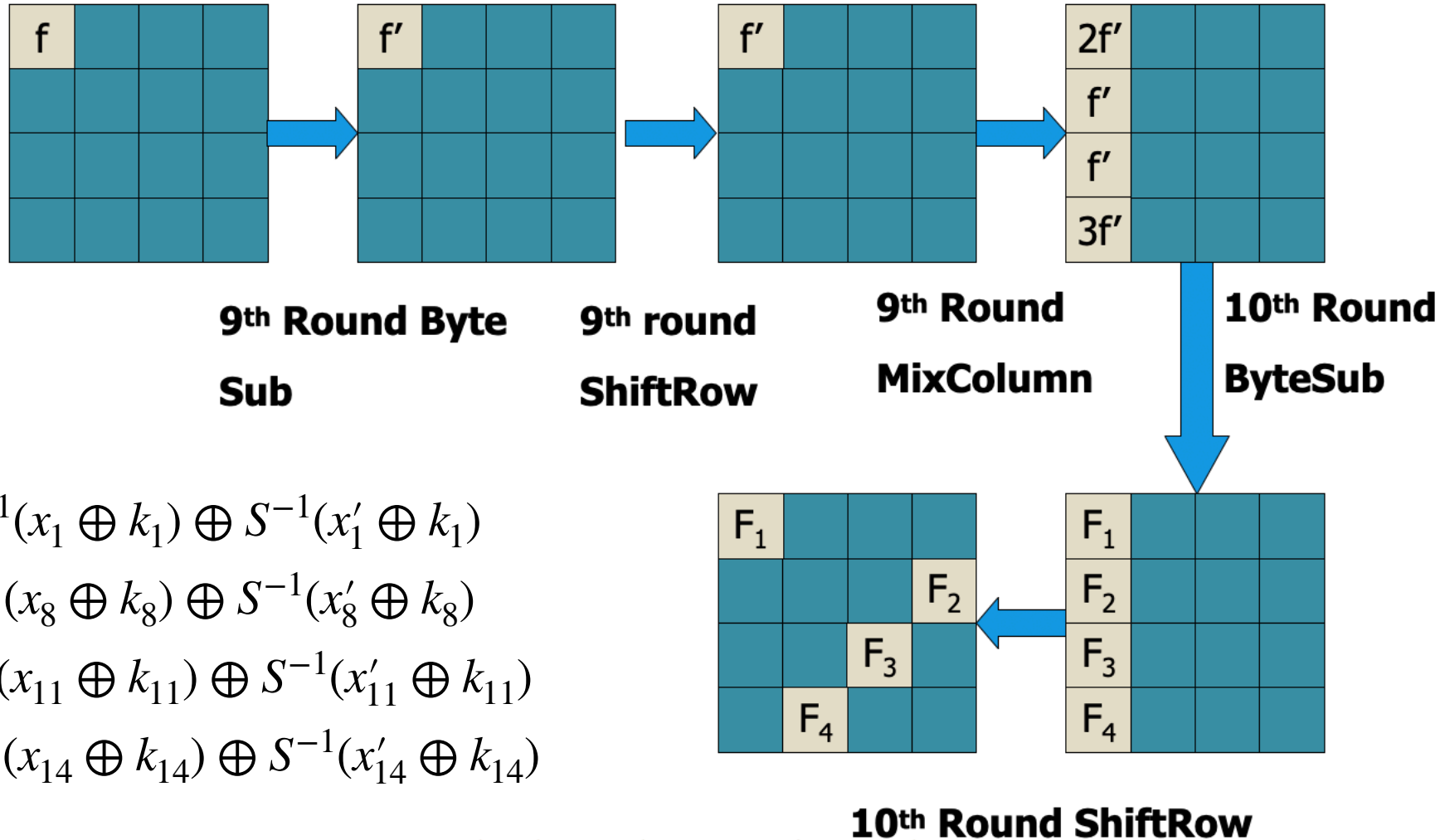
The AES Story...

- The attacker can corrupt one specific round operation of AES.
- One of multiple bytes of the AES state gets corrupted.
- Then what happens?
- We also assume that we have both correct and faulty ciphertext for the same plaintext...
- **Where to inject the fault:**
 - In round functions
 - In key schedule



Let there be a fault at 9th round

Let the correct ciphertext be
 $x = (x_1, x_2, \dots, x_{16})$
 The faulty ciphertext be
 $x' = (x'_1, x'_2, \dots, x'_{16})$



$$2f' = S^{-1}(x_1 \oplus k_1) \oplus S^{-1}(x'_1 \oplus k_1)$$

$$f' = S^{-1}(x_8 \oplus k_8) \oplus S^{-1}(x'_8 \oplus k_8)$$

$$f' = S^{-1}(x_{11} \oplus k_{11}) \oplus S^{-1}(x'_{11} \oplus k_{11})$$

$$3f' = S^{-1}(x_{14} \oplus k_{14}) \oplus S^{-1}(x'_{14} \oplus k_{14})$$

Let there be a fault at 9th round

$$2f' = S^{-1}(x_1 \oplus k_1) \oplus S^{-1}(x'_1 \oplus k_1)$$

$$f' = S^{-1}(x_8 \oplus k_8) \oplus S^{-1}(x'_8 \oplus k_8)$$

$$f' = S^{-1}(x_{11} \oplus k_{11}) \oplus S^{-1}(x'_{11} \oplus k_{11})$$

$$3f' = S^{-1}(x_{14} \oplus k_{14}) \oplus S^{-1}(x'_{14} \oplus k_{14})$$

- On an average there is one solution to the equation: $S^{-1}(X) \oplus S^{-1}(X + \alpha) = \beta$ — why?
 - Two possible solutions can be $X = 0$ and $X = \alpha$. In that case $\beta = \alpha^{-1}$.
 - If not, then we can transform this to $\beta x^2 + \alpha \beta x + \alpha = 0$ — which (may) have 2 more solutions
 - If $X = 0$ or $X = \alpha$, then the equation can have two more solutions in $GF(2^n)$, depending on n is even or odd. Solutions have form $\{0, \alpha, e\alpha, e^2\alpha\}$, with e being a field ($GF(2^n)$) element. — depends on some deep finite field tricks...
- **So, it can have 0, 2, or 4 solutions — on average 1 solutions, we observed....**
- Thus for one value of f' there is 1 value for k_1, k_8, k_{11}, k_{14} which satisfies the equations.
- Thus for all the 2^8 values of f' , there are 2^8 values for k_1, k_8, k_{11}, k_{14} .
- Thus the total size of AES key is 2^{32}

Let there be a fault at 9th round

- With one faulty cipher text:
 - Number of possible values per 4 bytes of the key is around 2^8 .
 - There are 2^{32} possible candidates for 128 bits of the AES key.
 - Brute force key is thus possible!

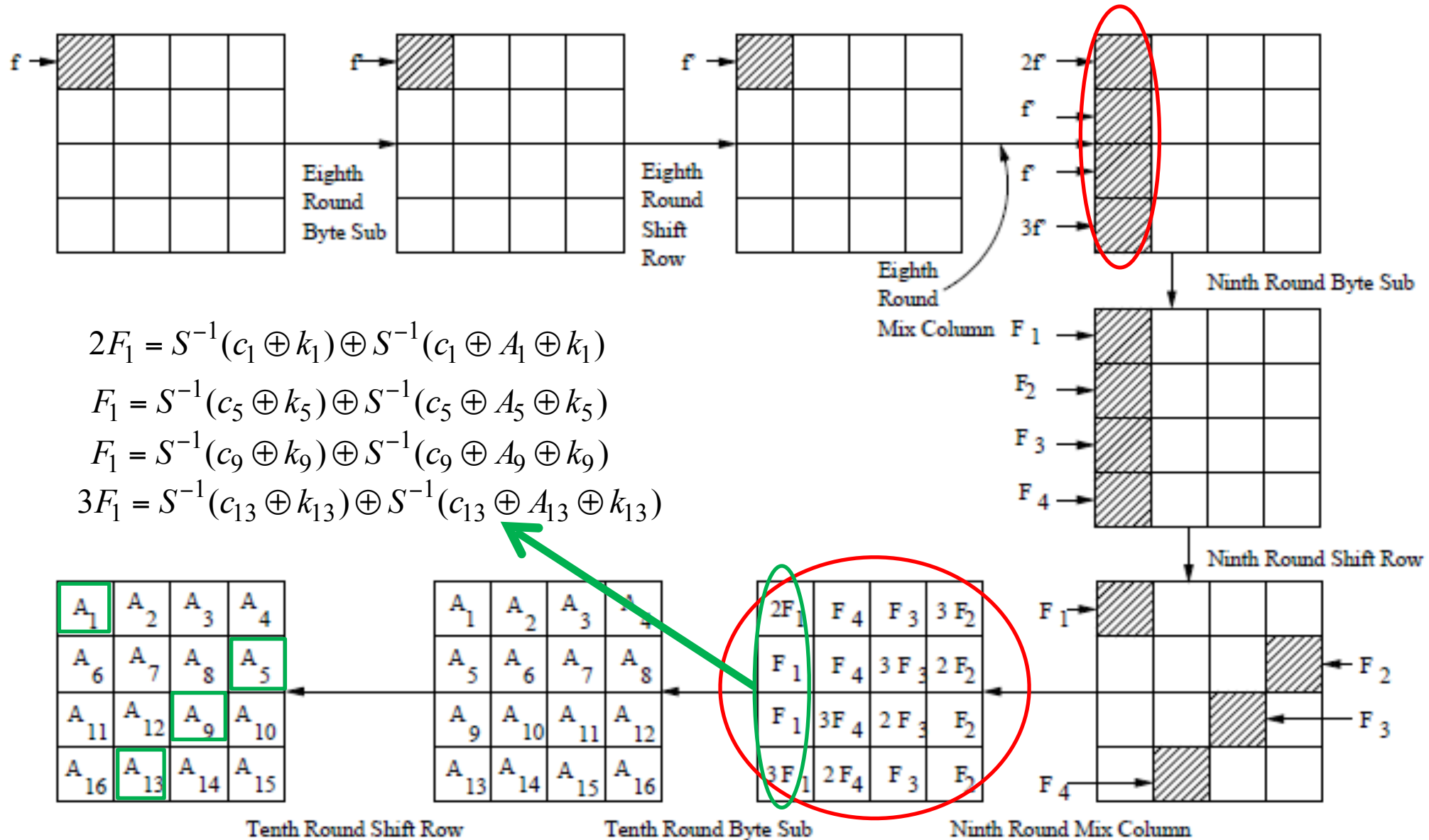
$$2f' = S^{-1}(x_1 \oplus k_1) \oplus S^{-1}(x'_1 \oplus k_1)$$

$$f' = S^{-1}(x_8 \oplus k_8) \oplus S^{-1}(x'_8 \oplus k_8)$$

$$f' = S^{-1}(x_{11} \oplus k_{11}) \oplus S^{-1}(x'_{11} \oplus k_{11})$$

$$3f' = S^{-1}(x_{14} \oplus k_{14}) \oplus S^{-1}(x'_{14} \oplus k_{14})$$

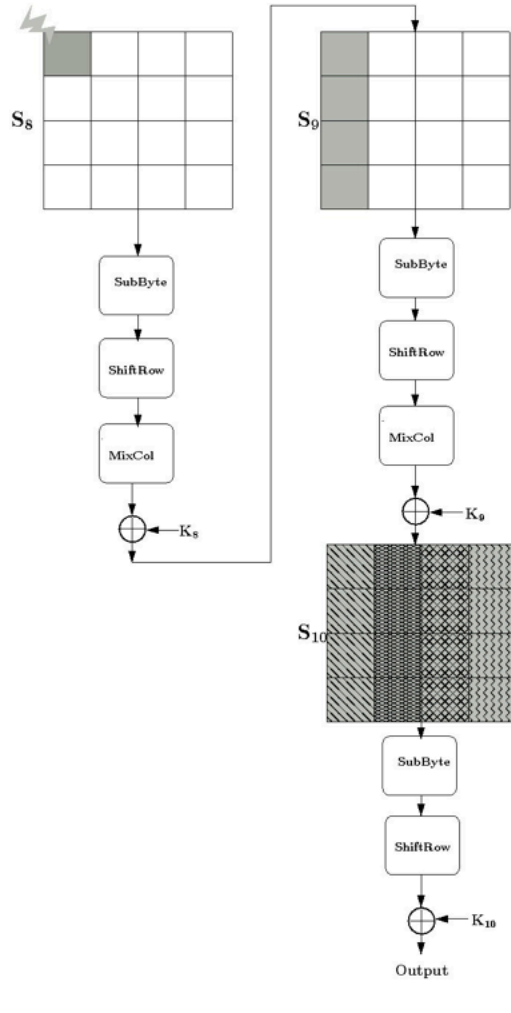
Let's Improve...8th Round Injection



Let's Improve...8th Round Injection...

- Search space reduced in two phases.
- First phase:
 - Find the 2^{32} candidates of 10th round key.
- Second phase
 - Deduce four differential equation from differences $\{2f', f', f', 3f'\}$.
 - Reduce the 2^{32} candidates to 2^8 using the four differential equation.

Let's Improve...8th Round Injection...



- Find 2^{32} candidates K_{10}

Differential Equation

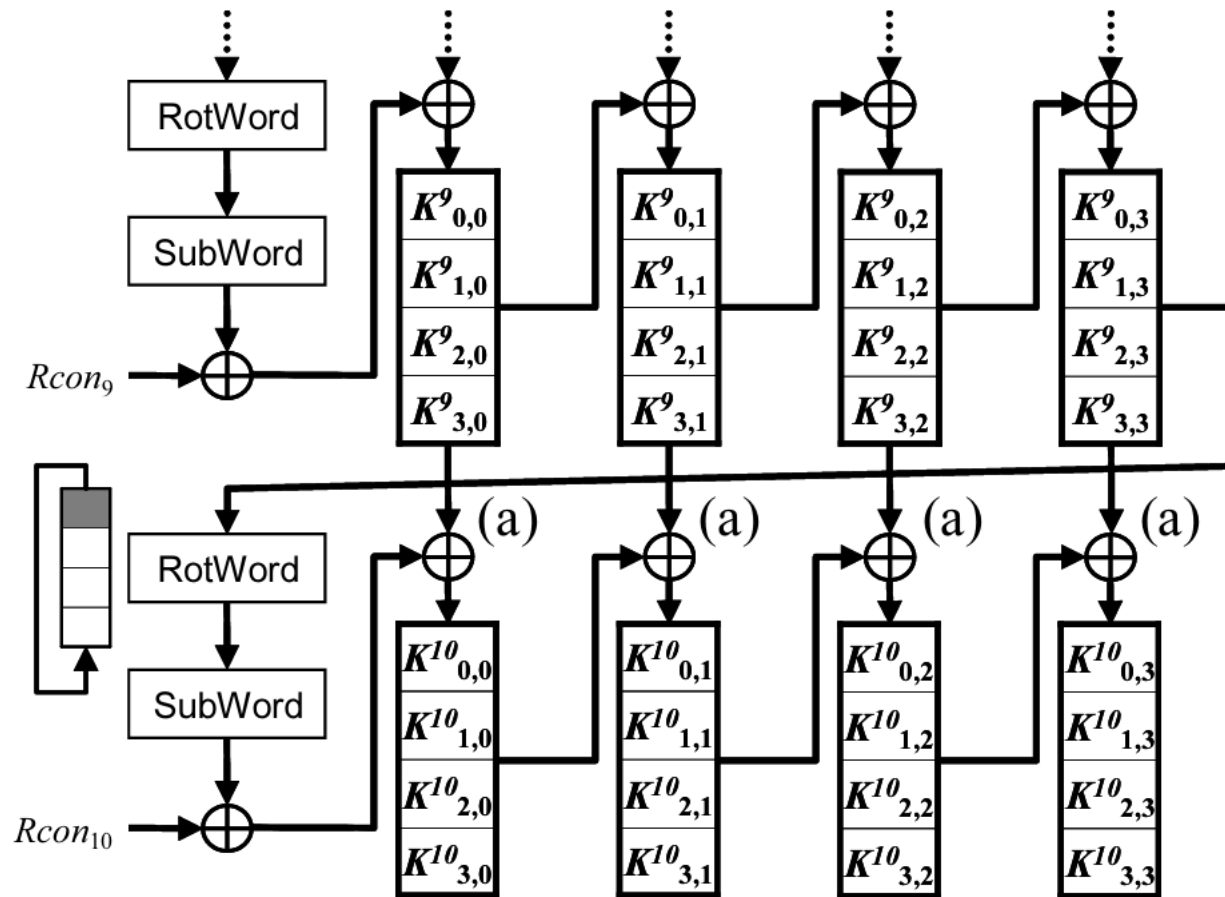
$$2F_1 = S^{-1}(x_0 \oplus k_0) \oplus S^{-1}(x'_0 \oplus k_0)$$

$$F_1 = S^{-1}(x_{13} \oplus k_{13}) \oplus S^{-1}(x'_{13} \oplus k_{13})$$

$$F_1 = S^{-1}(x_{10} \oplus k_{10}) \oplus S^{-1}(x'_{10} \oplus k_{10})$$

$$3F_1 = S^{-1}(x_7 \oplus k_7) \oplus S^{-1}(x'_7 \oplus k_7)$$

AES Key Schedule



Algorithm 2: The AES-128 KeySchedule function.

Input: $(r - 1)^{th}$ round key ($X = x_i$ for $i \in \{1, \dots, 16\}$).

Output: r^{th} round key X .

for $i \leftarrow 0$ **to** 3 **do**

$x_{(i < 2) + 1} \leftarrow x_{(i < 2) + 1} \oplus S(x_{(((i + 1) \wedge 3) < 2) + 4})$

end

$x_1 \leftarrow x_1 \oplus h_r$

for $i \leftarrow 1$ **to** 16 **do**

if $(i - 1) \bmod 4 \neq 0$ **then**

$x_i \leftarrow x_i \oplus x_{i-1}$

end

end

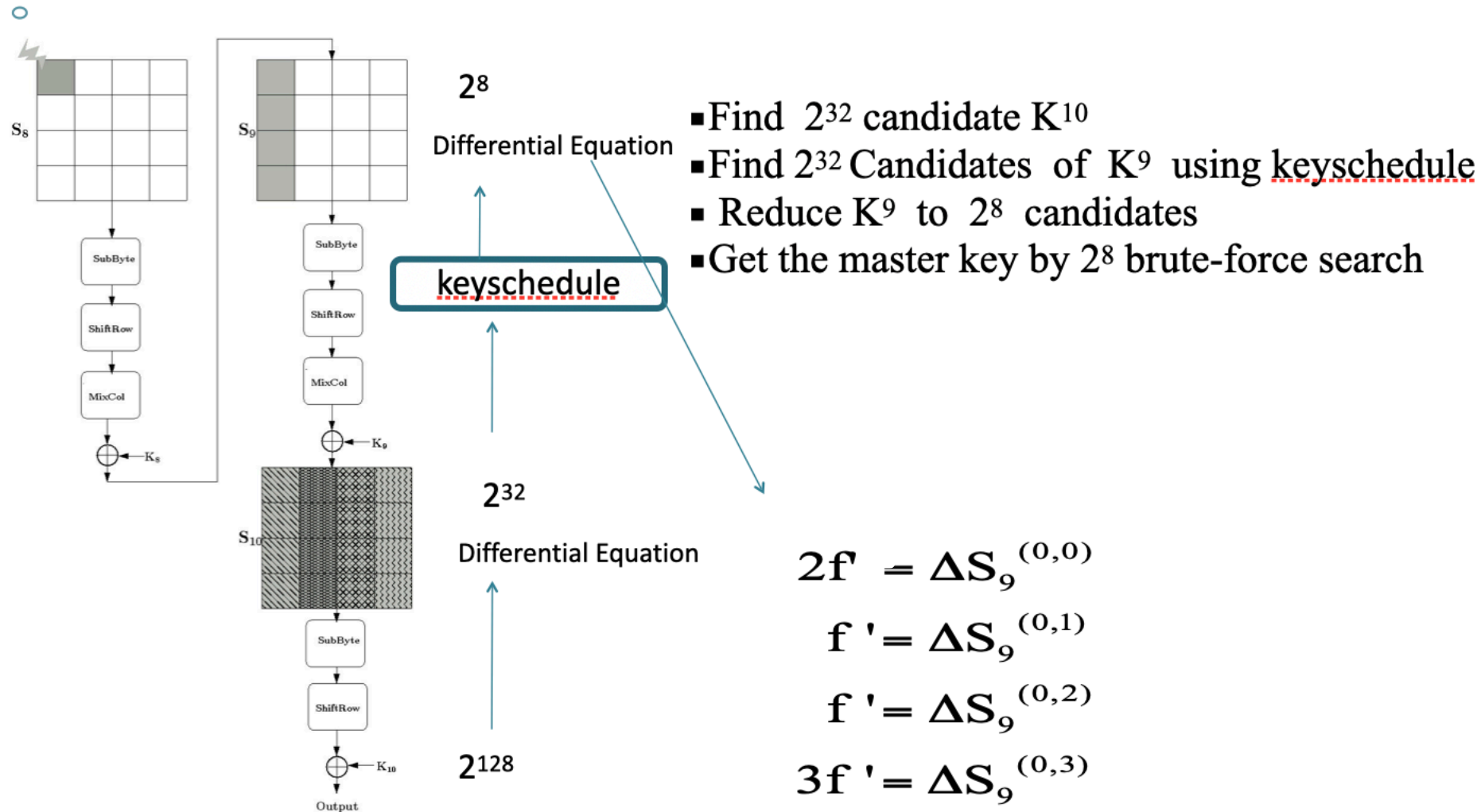
return X

Let's Improve...8th Round Injection...

$$\mathbf{K}^9 = \begin{pmatrix} k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10} & k_5 \oplus k_1 & k_9 \oplus k_5 & k_{13} \oplus k_9 \\ k_2 \oplus S(k_{15} \oplus k_{11}) & k_6 \oplus k_2 & k_{10} \oplus k_6 & k_{14} \oplus k_{10} \\ k_3 \oplus S(k_{16} \oplus k_{12}) & k_7 \oplus k_3 & k_{11} \oplus k_7 & k_{15} \oplus k_{11} \\ k_4 \oplus S(k_{13} \oplus k_9) & k_8 \oplus k_4 & k_{12} \oplus k_8 & k_{16} \oplus k_{12} \end{pmatrix} \cdot \mathbf{K}^{10} = \{k_1, k_2, \dots, k_{16}\}$$

- AES key schedule is invertible

Let's Improve...8th Round Injection...



8th Round Injection...How does the Equation Look?

$$\begin{aligned} 2 f' &= S^{-1} \left(14 \left(S^{-1}(x_1 \oplus k_1) \oplus k'_1 \right) \oplus 11 \left(S^{-1}(x_{14} \oplus k_{14}) \oplus k'_2 \right) \oplus \right. \\ &\quad \left. 13 \left(S^{-1}(x_{11} \oplus k_{11}) \oplus k'_3 \right) \oplus 9 \left(S^{-1}(x_8 \oplus k_8) \oplus k'_4 \right) \right) \oplus \\ &\quad S^{-1} \left(14 \left(S^{-1}(x'_1 \oplus k_1) \oplus k'_1 \right) \oplus 11 \left(S^{-1}(x'_{14} \oplus k_{14}) \oplus k'_2 \right) \oplus \right. \\ &\quad \left. 13 \left(S^{-1}(x'_{11} \oplus k_{11}) \oplus k'_3 \right) \oplus 9 \left(S^{-1}(x'_8 \oplus k_8) \oplus k'_4 \right) \right) \\ &= S^{-1} \left(14 \left(S^{-1}(x_1 \oplus k_1) \oplus ((k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10})) \right) \oplus \right. \\ &\quad \left. 11 \left(S^{-1}(x_{14} \oplus k_{14}) \oplus (k_2 \oplus S(k_{15} \oplus k_{11})) \right) \oplus \right. \\ &\quad \left. 13 \left(S^{-1}(x_{11} \oplus k_{11}) \oplus (k_3 \oplus S(k_{16} \oplus k_{12})) \right) \oplus \right. \\ &\quad \left. 9 \left(S^{-1}(x_8 \oplus k_8) \oplus (k_4 \oplus S(k_{13} \oplus k_9)) \right) \right) \oplus \\ &\quad S^{-1} \left(14 \left(S^{-1}(x'_1 \oplus k_1) \oplus ((k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10})) \right) \oplus \right. \\ &\quad \left. 11 \left(S^{-1}(x'_{14} \oplus k_{14}) \oplus (k_2 \oplus S(k_{15} \oplus k_{11})) \right) \oplus \right. \\ &\quad \left. 13 \left(S^{-1}(x'_{11} \oplus k_{11}) \oplus (k_3 \oplus S(k_{16} \oplus k_{12})) \right) \oplus \right. \\ &\quad \left. 9 \left(S^{-1}(x'_8 \oplus k_8) \oplus (k_4 \oplus S(k_{13} \oplus k_9)) \right) \right) \end{aligned}$$

- We have 4 such equations

Let's Improve...8th Round Injection...

- Time complexity of previous attack: $O(2^{32})$
- Time complexity of this attack: $O(2^8)$
- Just with one fault injection!!!

Diagonal Fault Attack

- Multi Byte Faults (more practical)
 - Attacker induces fault at the input of the 8th round in some bytes
 - Fault value should be non-zero but can be arbitrary

Diagonal Fault Attack

b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	b_{33}

Diagonal: A diagonal is a set of four bytes of the state matrix,

where diagonal i is defined as follows:

$$D_i = \{b_{j, (j+i) \bmod 4} \ ; \ 0 \leq j < 4\}$$

According to the above definition and with reference to the state matrix of AES (refer figure) we obtain the following four diagonals.

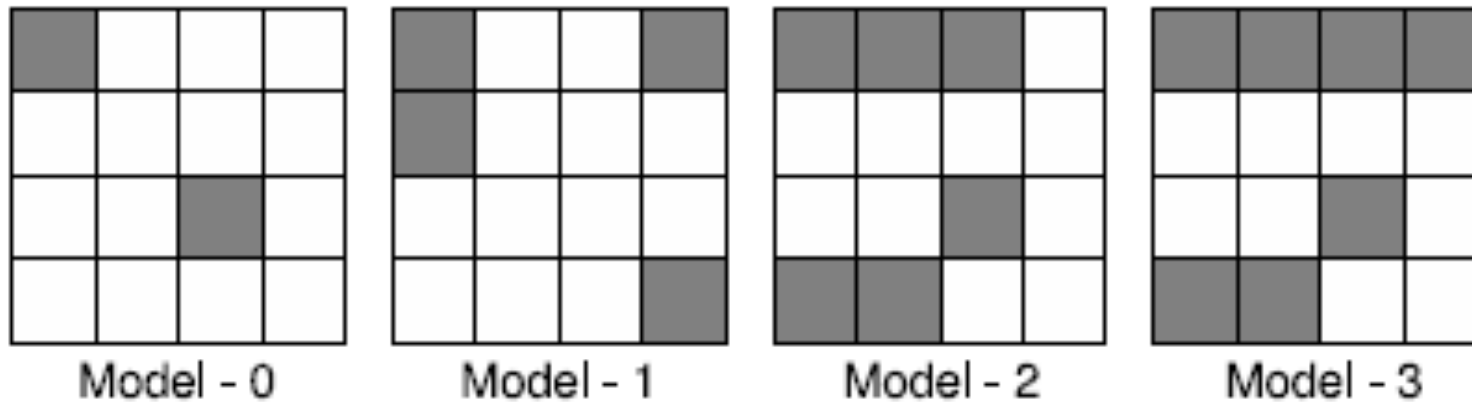
$$D_0 = (b_{00}, b_{11}, b_{22}, b_{33})$$

$$D_1 = (b_{01}, b_{12}, b_{23}, b_{30})$$

$$D_2 = (b_{02}, b_{13}, b_{20}, b_{31})$$

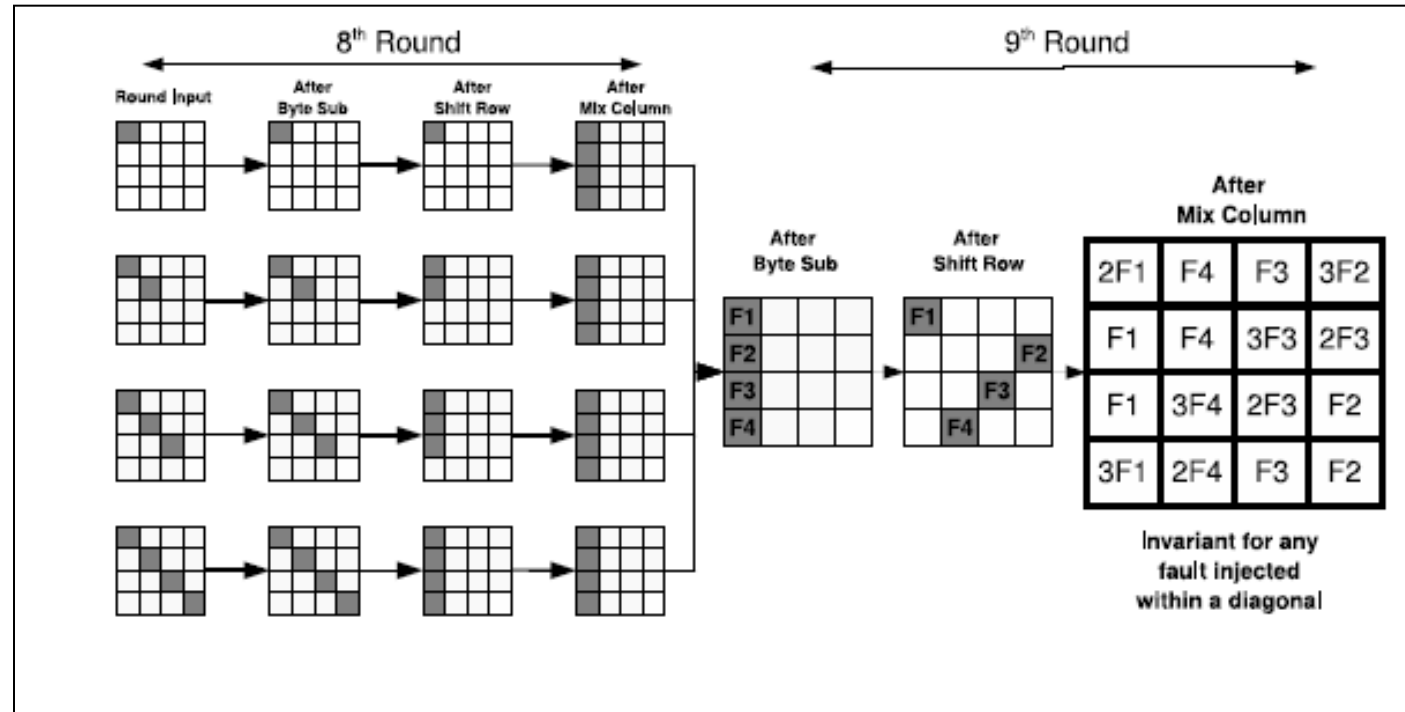
$$D_3 = (b_{03}, b_{10}, b_{21}, b_{32})$$

Diagonal Fault Attack



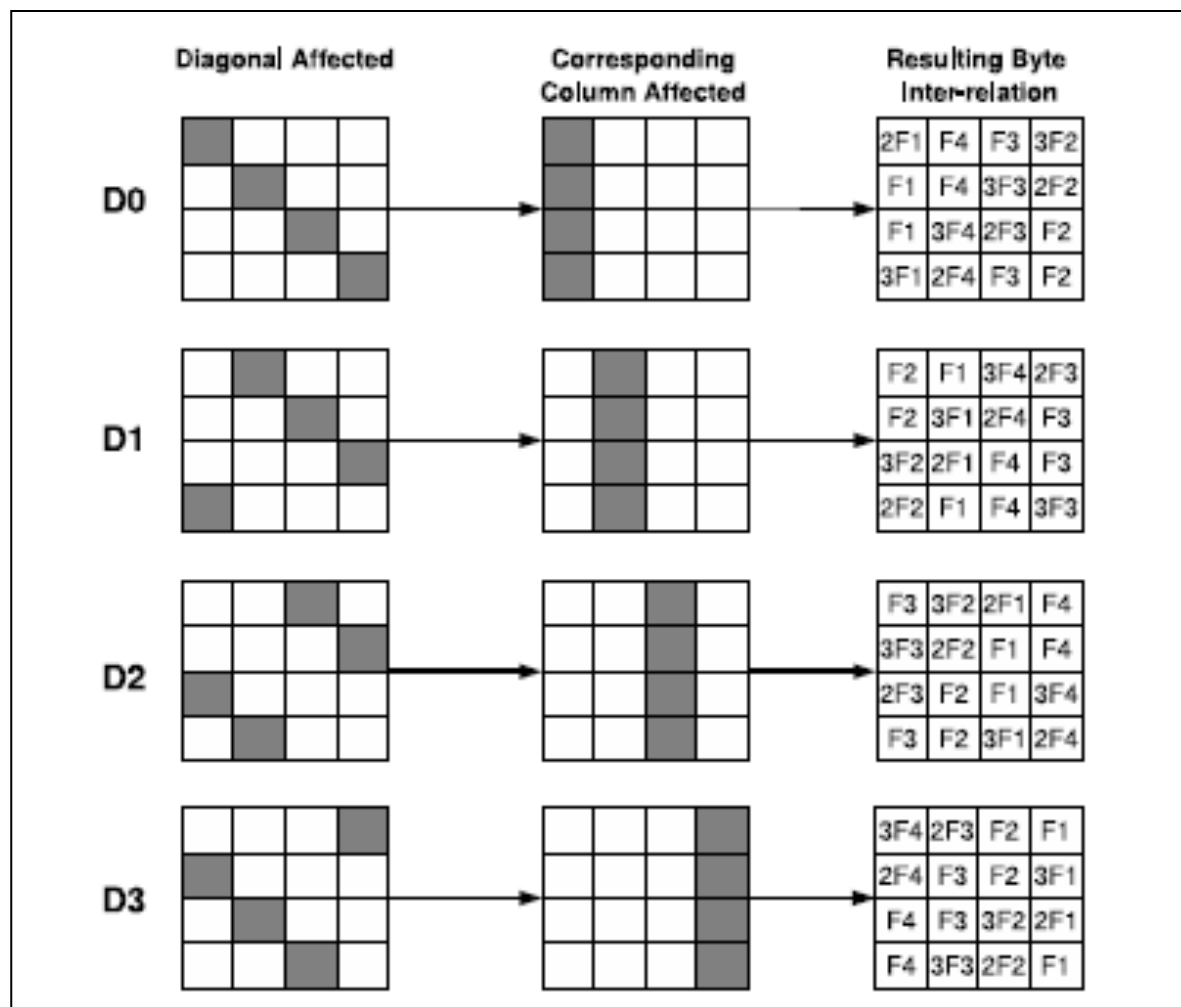
- M0: One Diagonal affected.
- M1: Two Diagonals affected.
- M2: Three Diagonals affected.
- M3: Four Diagonals affected.

Diagonal Fault Attack



- Faults induced in Diagonal D_0 at the input of 8th round AES are all equivalent.

Diagonal Fault Attack



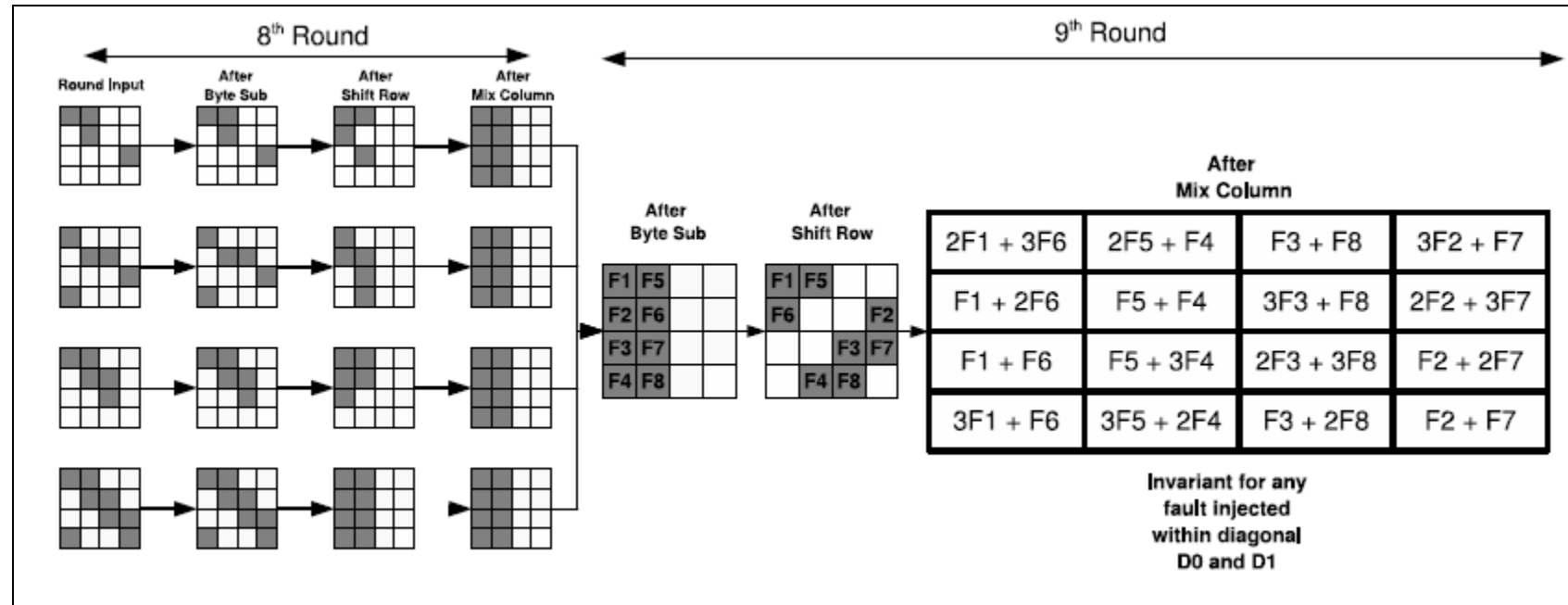
Diagonal Fault Attack

$$\mathbf{CT} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix} \quad \mathbf{CT}' = \begin{pmatrix} x'_1 & x'_2 & x'_3 & x'_4 \\ x'_5 & x'_6 & x'_7 & x'_8 \\ x'_9 & x'_{10} & x'_{11} & x'_{12} \\ x'_{13} & x'_{14} & x'_{15} & x'_{16} \end{pmatrix} \quad \mathbf{K}_{10} = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 \\ k_5 & k_6 & k_7 & k_8 \\ k_9 & k_{10} & k_{11} & k_{12} \\ k_{13} & k_{14} & k_{15} & k_{16} \end{pmatrix}$$

$$\begin{aligned} ISB(x_1 + k_1) + ISB(x'_1 + k_1) &= 2[ISB(x_8 + k_8) + ISB(x'_8 + k_8)] \\ ISB(x_8 + k_8) + ISB(x'_8 + k_8) &= ISB(x_{11} + k_{11}) + ISB(x'_{11} + k_{11}) \\ ISB(x_{14} + k_{14}) + ISB(x'_{14} + k_{14}) &= 3[ISB(x_8 + k_8) + ISB(x'_8 + k_8)] \end{aligned}$$

- There are in total 4 such systems of equations for a diagonal D_0 .
- Each system of equation gives 2^8 keys on an average.
- AES key size gets reduced to 2^{32} .

Diagonal Fault Attack: 2 Diagonals Corrupted

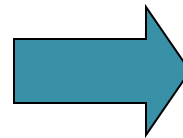


$$a_0 = 2F_1 + 3F_6$$

$$a_1 = F_1 + 2F_6$$

$$a_2 = F_1 + F_6$$

$$a_3 = 3F_1 + F_6$$



$$a_1 + a_3 = a_0$$

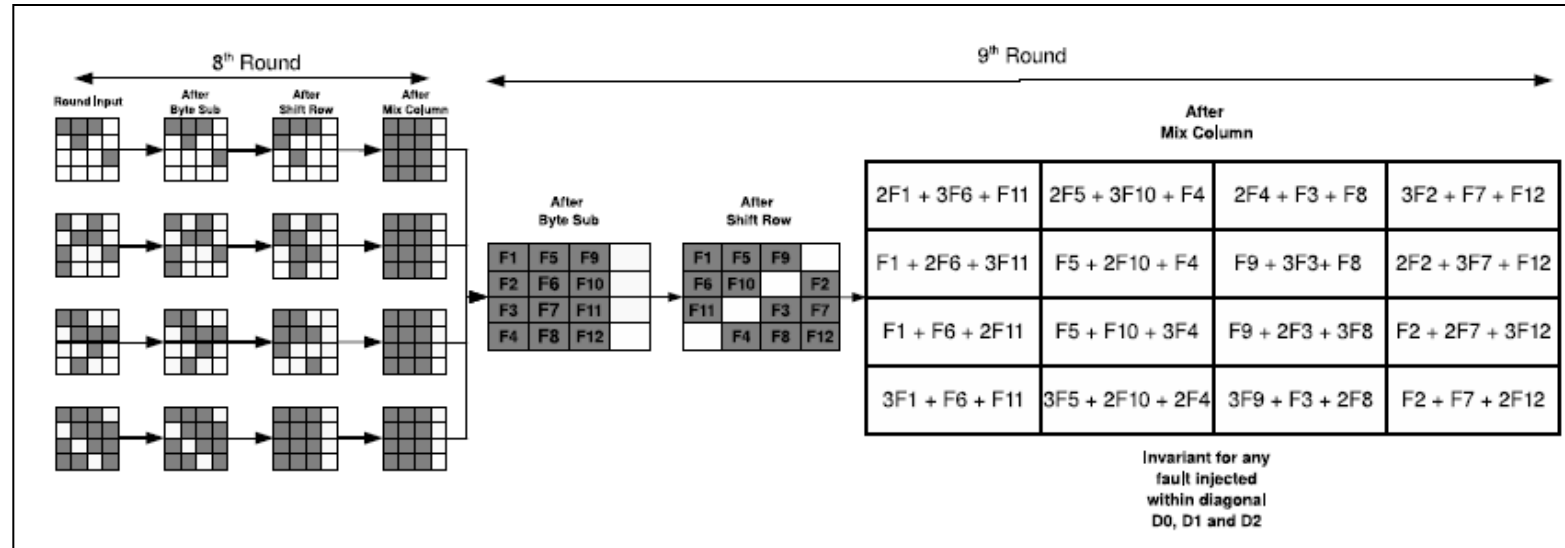
$$2a_1 + 3a_3 = 7a_2$$

Diagonal Fault Attack: 2 Diagonals Corrupted

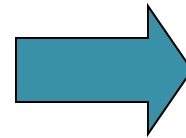
$$\begin{aligned}a_0 &= ISB(x_1 + k_1) + ISB(x'_1 + k_1) \\a_1 &= ISB(x_8 + k_8) + ISB(x'_8 + k_8) \\a_2 &= ISB(x_{11} + k_{11}) + ISB(x'_{11} + k_{11}) \\a_3 &= ISB(x_{14} + k_{14}) + ISB(x'_{14} + k_{14})\end{aligned}$$

- The equation reduces the space of the 4 key bytes of AES to 2^{16}
- Two faulty ciphertexts reduce it to a unique value on an average (experimental result).

Diagonal Fault Attack: 3 Diagonals Corrupted



$$\begin{aligned} a_0 &= 2F_1 + 3F_6 + F_{11} \\ a_1 &= F_1 + 2F_6 + 3F_{11} \\ a_2 &= F_1 + F_6 + 2F_{11} \\ a_3 &= 3F_1 + F_6 + F_{11} \end{aligned}$$



$$11a_0 + 13a_1 = 9a_2 + 14a_3$$

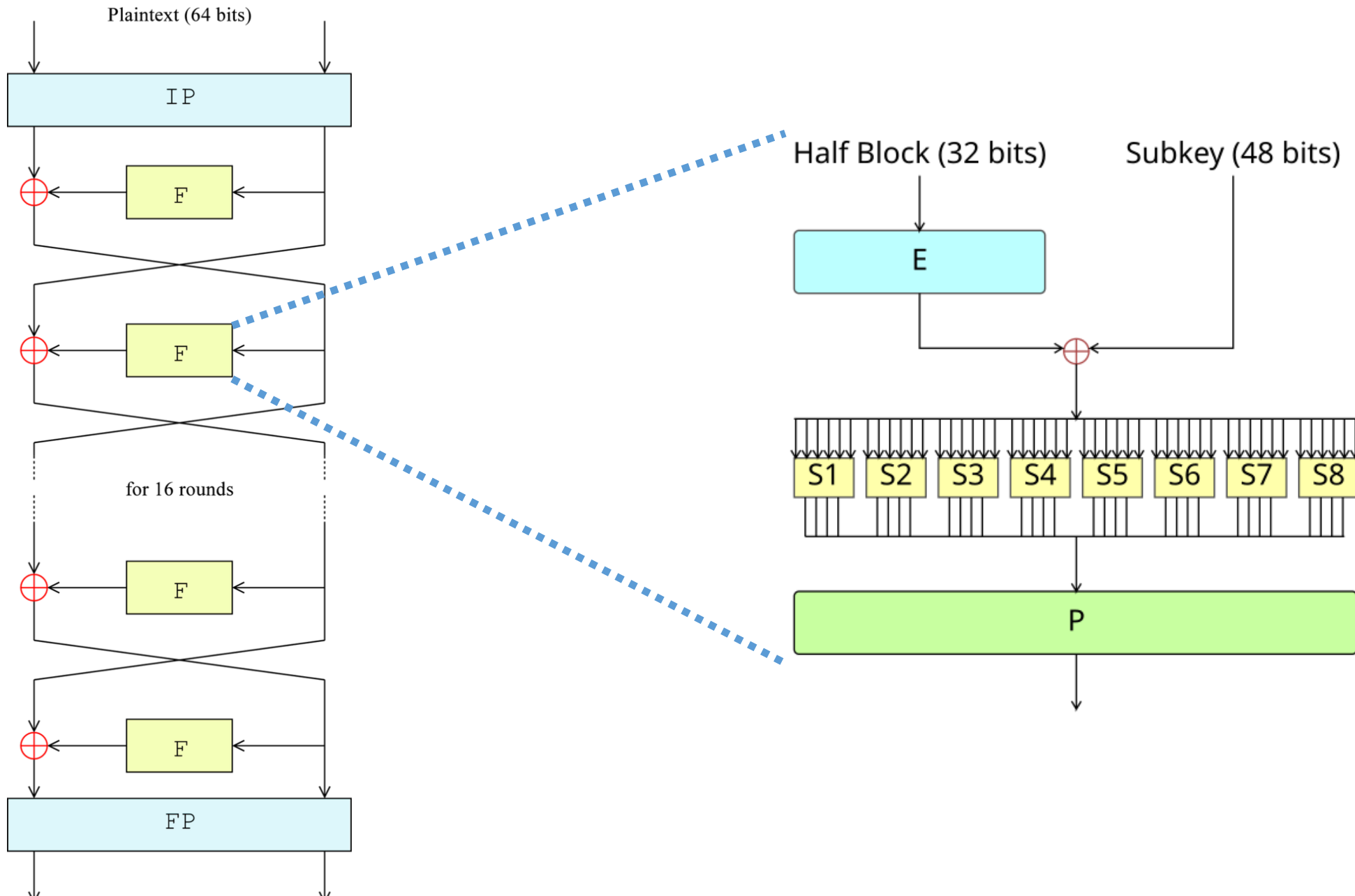
Diagonal Fault Attack: 3 Diagonals Corrupted

$$\begin{aligned}a_0 &= ISB(x_1 + k_1) + ISB(x'_1 + k_1) \\a_1 &= ISB(x_8 + k_8) + ISB(x'_8 + k_8) \\a_2 &= ISB(x_{11} + k_{11}) + ISB(x'_{11} + k_{11}) \\a_3 &= ISB(x_{14} + k_{14}) + ISB(x'_{14} + k_{14})\end{aligned}$$

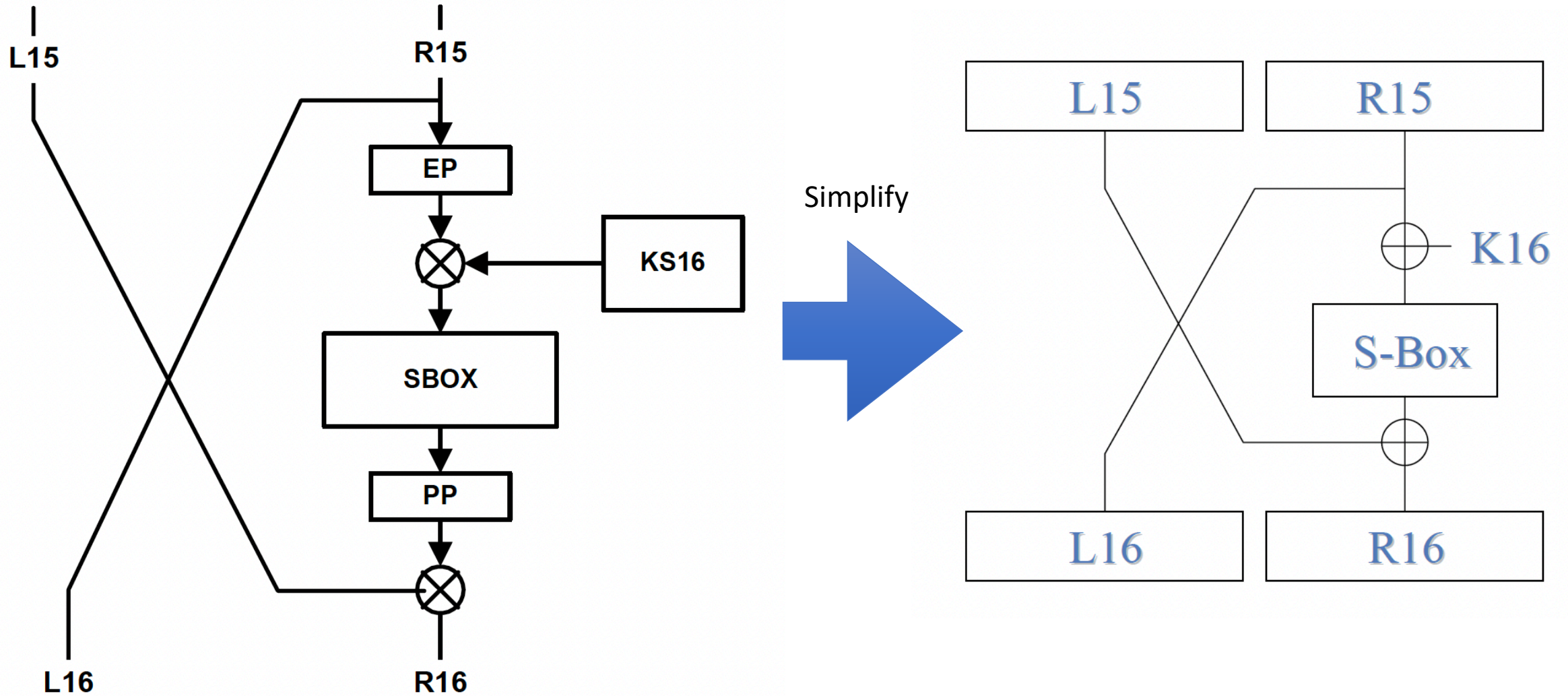
- The equation reduces the space of the 4 key bytes of AES to 2^{24}
- Four faulty ciphertexts reduce it to a unique value on an average (experimental result).

$$\left\{ \begin{array}{l} \text{---} \\ \text{---} \end{array} \right\}$$

Going Beyond AES...Let's Try DES

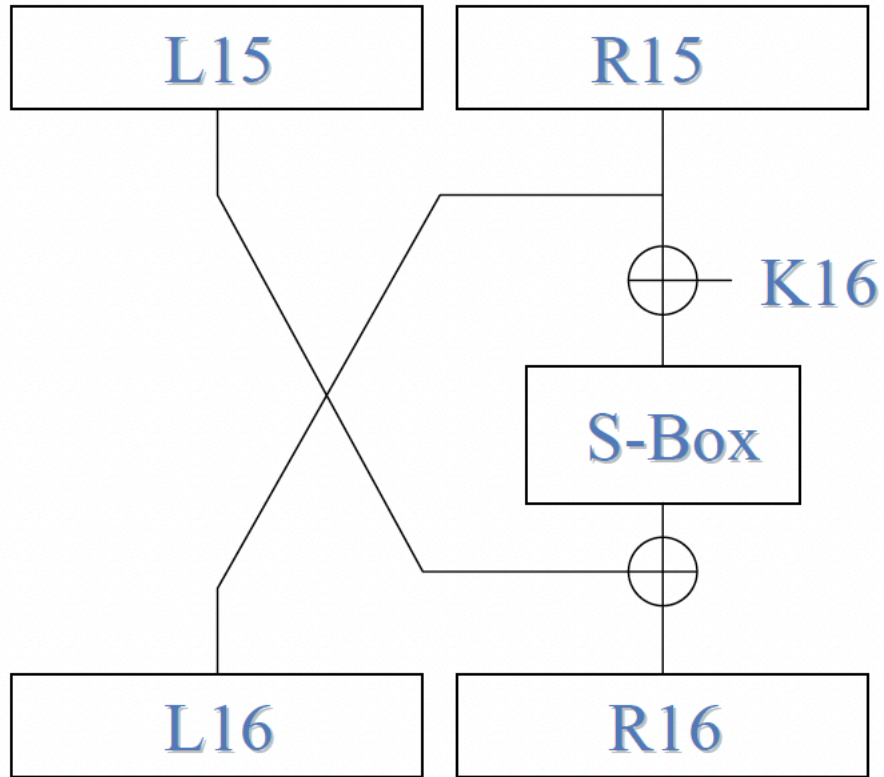


Going Beyond AES...Let's Try DES



- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Going Beyond AES...Let's Try DES

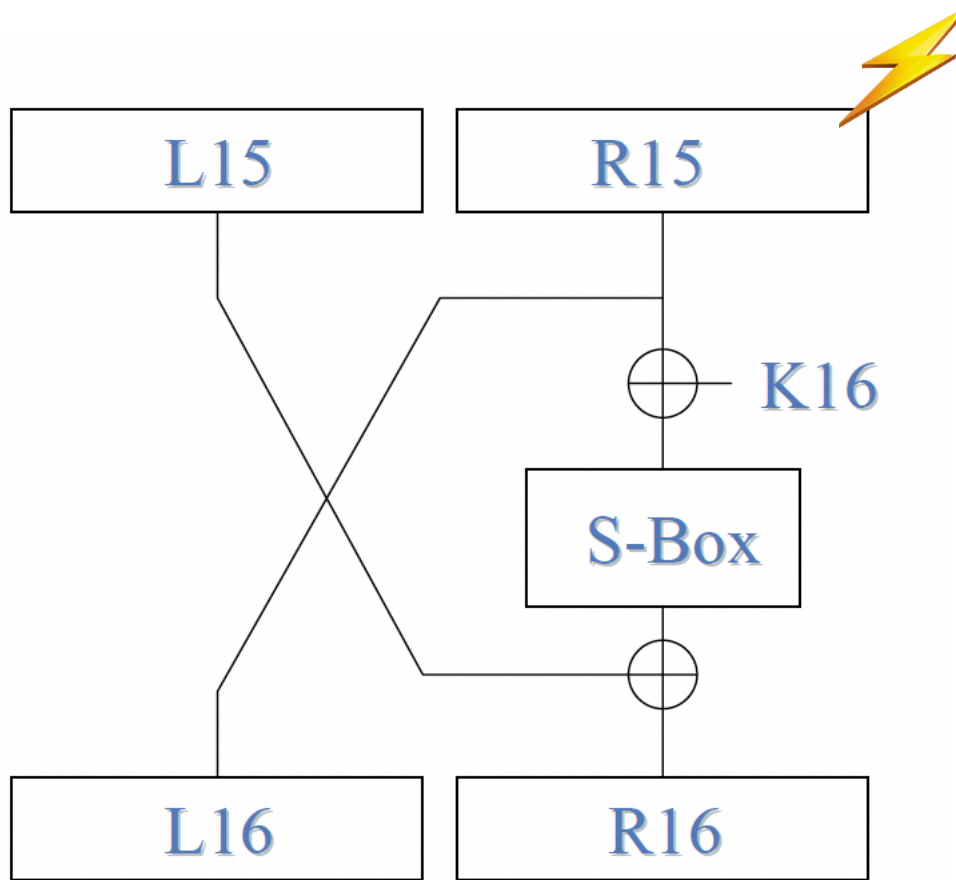


$$L16 = R15$$

$$R16 = S(R15 \oplus K16) \oplus L15$$

- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Going Beyond AES...Let's Try DES



$$L16 = R15$$

$$R16 = S(R15 \oplus K16) \oplus L15$$

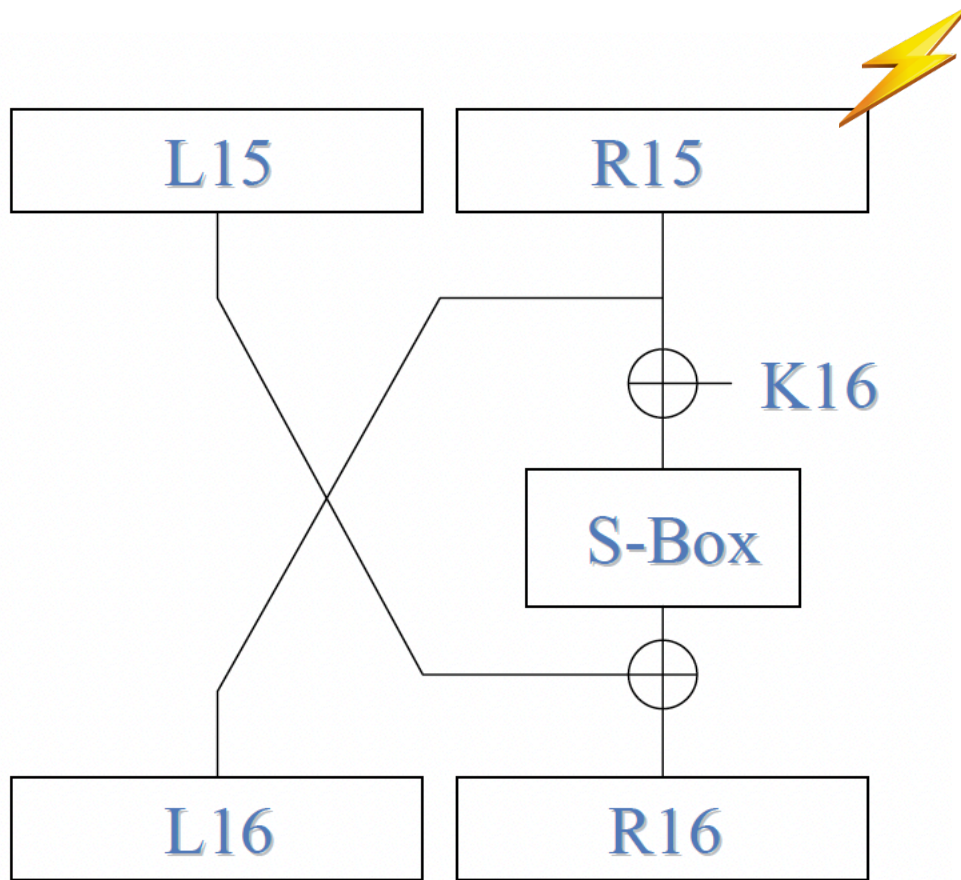
After Fault...

$$L16' = R15'$$

$$R16' = S(R15' \oplus K16) \oplus L15$$

- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

Going Beyond AES...Let's Try DES



$$L16 = R15$$

$$R16 = S(R15 \oplus K16) \oplus L15$$

After Fault...

$$L16' = R15'$$

$$R16' = S(R15' \oplus K16) \oplus L15$$

Calculate the differential...

$$\begin{aligned} R16 \oplus R16' &= S(R15 \oplus K16) \oplus L15 \oplus S(R15' \oplus K16) \oplus L15 \\ &= S(L16 \oplus K16) \oplus S(L16' \oplus K16) \end{aligned}$$

- The Sorcerer's Apprentice's Guide to Fault Attacks, FDTC 2006

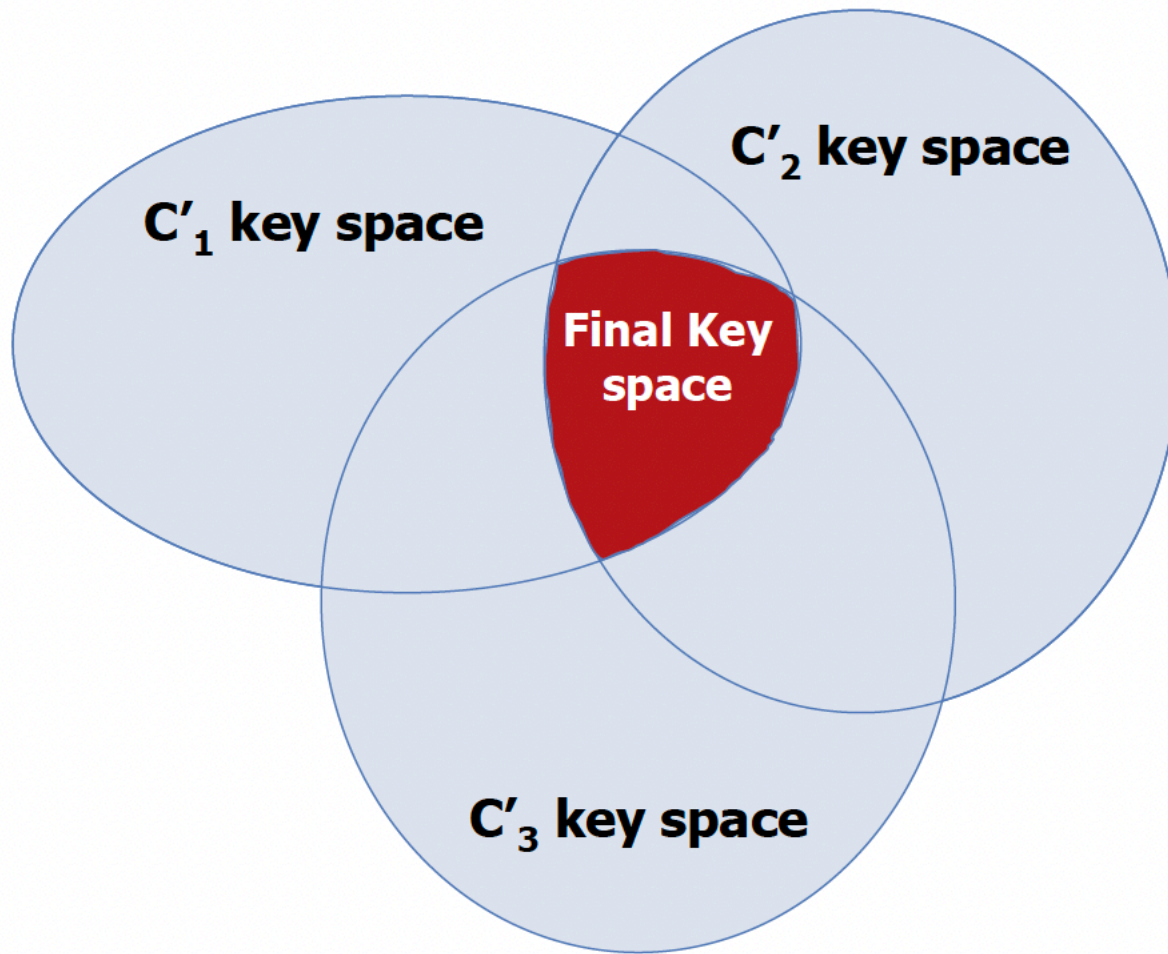
DES Attack

- Note that here both input and output differentials are known...
 - Unlike AES where only a pattern of the input differentials are known
- So, you can solve the equations for k as the only unknown
 - Note that DES S-Boxes are 6×4 ...so 6 bits has to be guessed simultaneously...
 - You can also use the differential distribution table mentioned in the last class to find the number of solutions for each S-Box

DES Attack

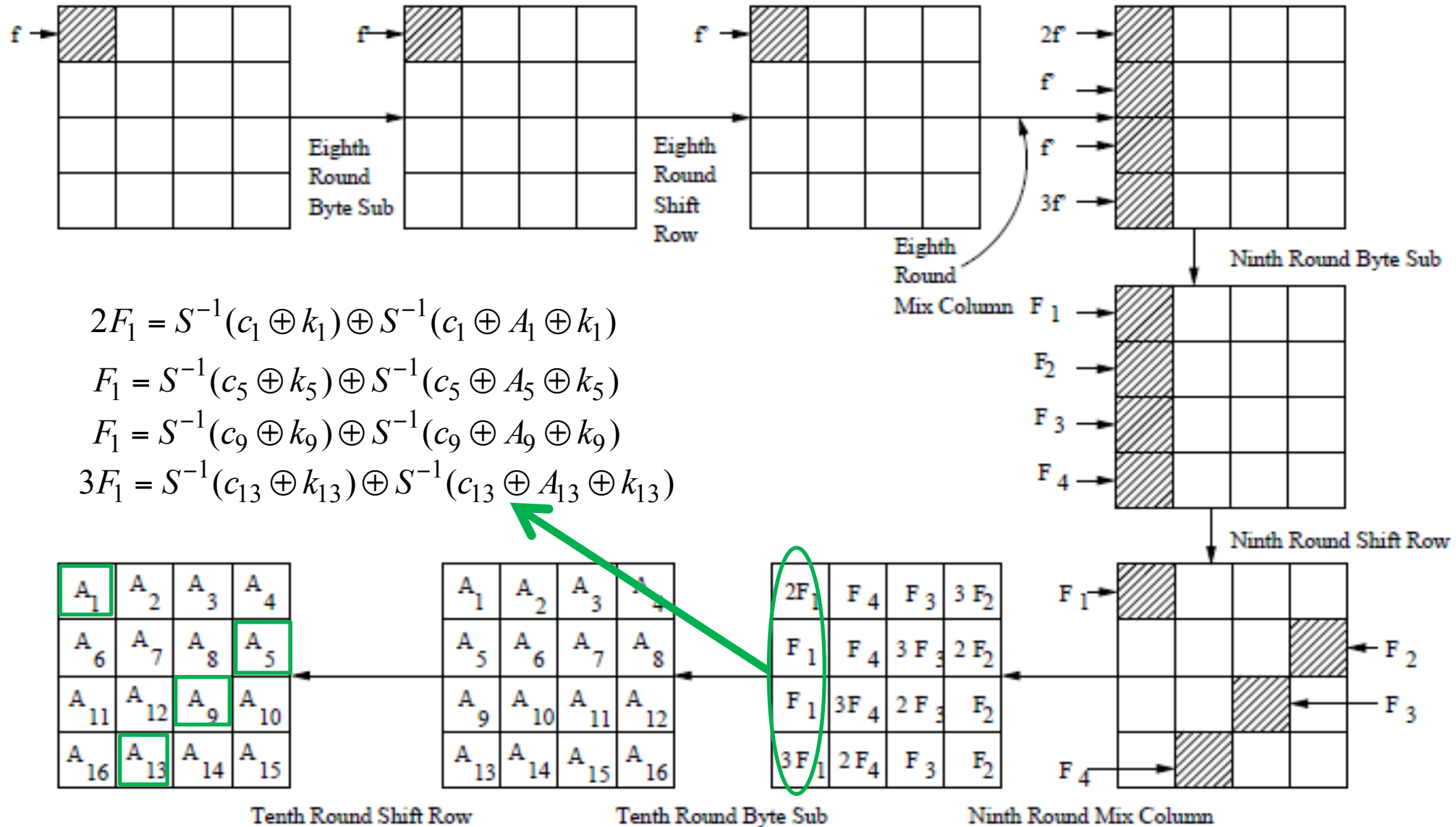
- Each equation gives 4-5 solutions for the key on average...
- So, for 8 S-boxes, roughly $2^{18} ((5)^8)$ suggestions remain for the last round key.
- Another issue with DES — although the key size is 56, round key size is 48.
- So 2^{48} reduces to 2^{18} — total master key can be found with $2^{18} \times 2^8 = 2^{26}$ exhaustive search...

Using Multiple Ciphertexts



- Common strategy for reducing attack complexity
 - Applies irrespective of AES or DES or any other cipher..
 - Let's try for AES...
 - In general, multiplying the number of faulty ciphertext directly reduces the final key space size and therefore the complexity of the exhaustive search

Let's Improve...8th Round Injection



Let's Improve...8th Round Injection

- For 1 column, a single injection reduces the key space from 2^{32} to 2^8 .
- 2^8 is basically the expected number of remaining subkeys
- This is done by satisfying the equation system.
- The probability of a given (4-byte) subkey satisfying this equation is $\frac{2^8}{2^{32}} = 2^{-24}$.
- Now, consider another fault injection
- We can take intersections between the key suggestions provided by these two injections...
- For 2-injections the probability will become $(2^{-24})^2$.
- Expected number of remain subkeys:
 $2^{32} \times (2^{-24})^2 < 1$ — **So practically only one subkey will remain**

$$2F_1 = S^{-1}(c_1 \oplus k_1) \oplus S^{-1}(c_1 \oplus A_1 \oplus k_1)$$

$$F_1 = S^{-1}(c_5 \oplus k_5) \oplus S^{-1}(c_5 \oplus A_5 \oplus k_5)$$

$$F_1 = S^{-1}(c_9 \oplus k_9) \oplus S^{-1}(c_9 \oplus A_9 \oplus k_9)$$

$$3F_1 = S^{-1}(c_{13} \oplus k_{13}) \oplus S^{-1}(c_{13} \oplus A_{13} \oplus k_{13})$$

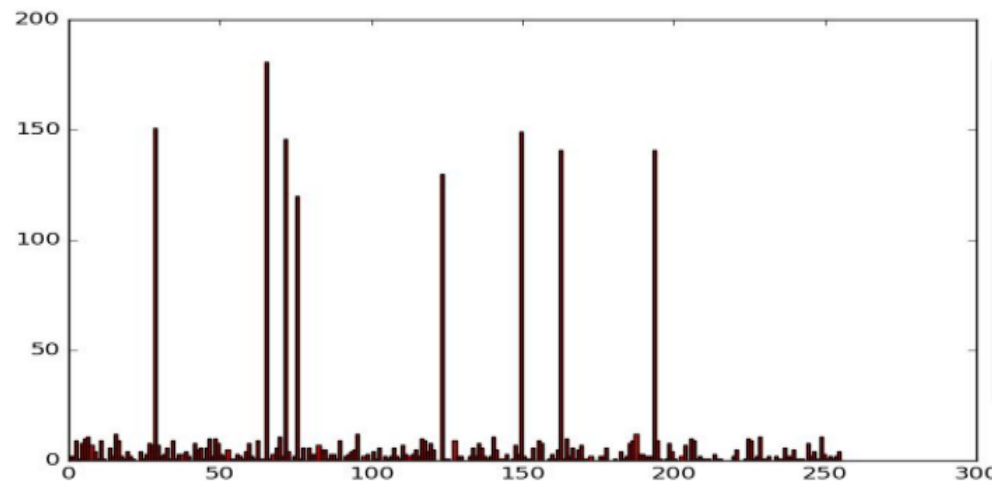
A General Model for Differential Fault Attacks

- **Identify a distinguisher:**
 - Based on fault patterns...
 - **Example:** For AES, 8th round injection, the distinguisher is the linear pattern observed due to fault propagation
 - For DES, the distinguisher is $R16 \oplus R16'$ — a known value and part of ciphertext!!!
 - Distinguisher is mostly an input differential Δ_i .
 - Always try to form equations of the form $S^{-1}(X) \oplus S^{-1}(X + \alpha) = \beta$, where β is a (part of) the distinguisher..
- **Divide-and Conquer:** Find a distinguisher evaluation strategy so that you do not need to guess a lot of keys together.
 - For AES attacks, this distinguisher evaluation complexity was 2^{36} — we have a divide-and conquer strategy which guesses 32-bits of key at once (independently for 4 columns)
 - For DES it is 2^9
- **Calculate the Remaining Key Space:** Estimate the number of solutions to your equation system. That gives you the number of remaining keys after the attack. You have to search these keys exhaustively...

Fault Attacks (FA)

Fault Models

- Key component of a fault attack
- Attack procedure changes according to the fault model
- Random localized faults
 - Bit/nibble/byte fault
 - Most general model
- Biased faults
 - **Device-dependent model**
- Instruction skip/modify.
- Constant fault
 - **Stuck-at-0/1**



In a fault space of size 256, only 8 faults occur in 98% of the total injections!!!

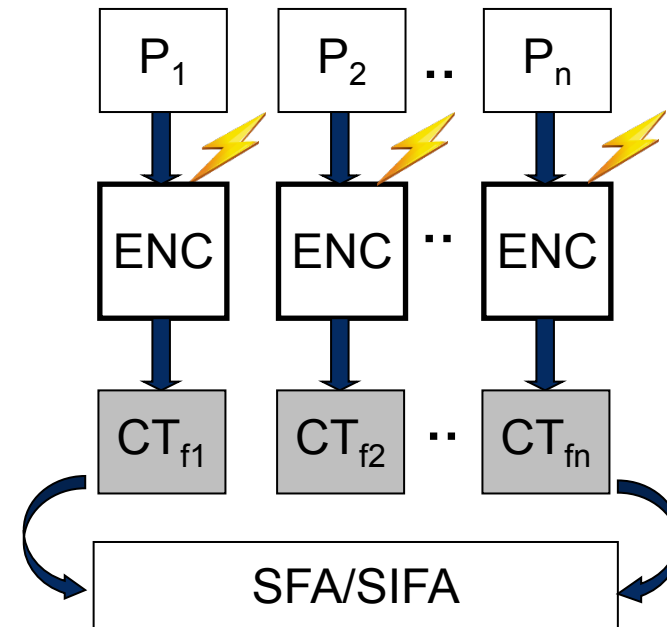
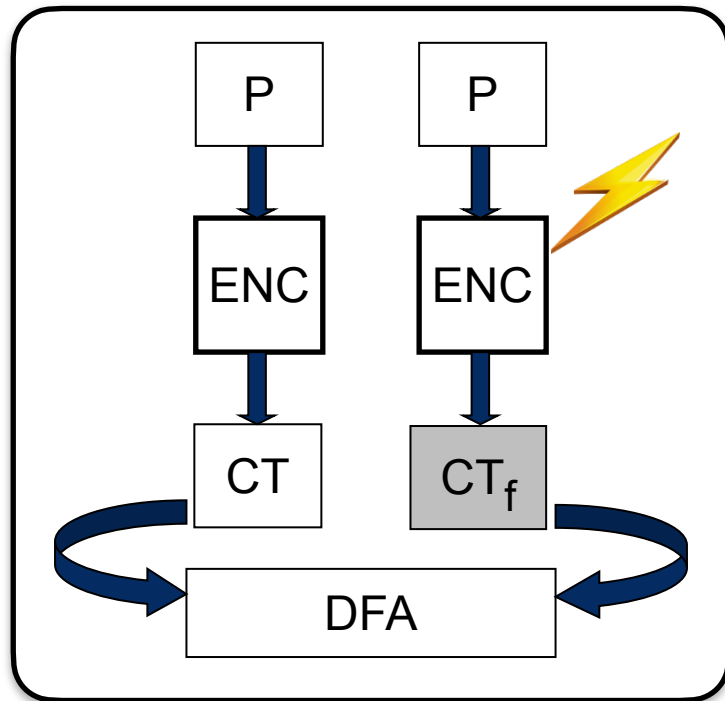
```
ldi r1 0;  
ld r1 #M1  
ldi r2 0  
ld r2 #M2  
add r1 r2  
str r1
```



```
ldi r1 0;  
nop  
ldi r2 0  
ld r2 #M2  
add r1 r2  
str r1
```

Fault Attacks (FA)

- Inject fault in the device during encryption/decryption.
- Analyze the faulty response to extract the secret
 - Differential fault attacks.
 - Biased fault attacks (SFA/SIFA).



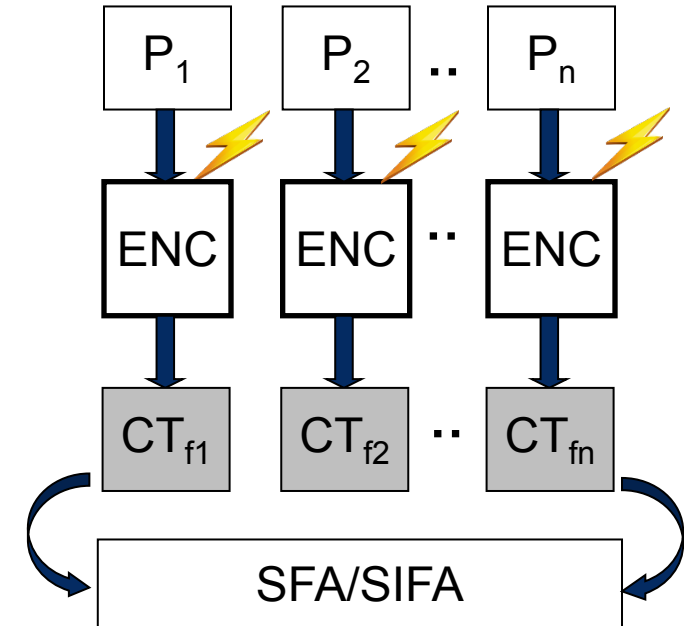
Statistical Fault Analysis (SFA)

The Key Idea:

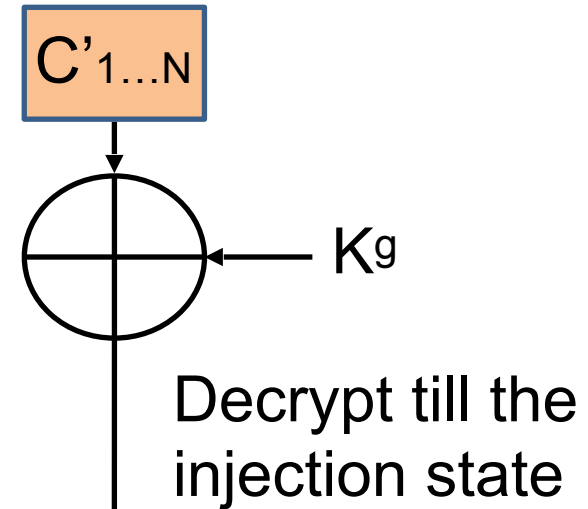
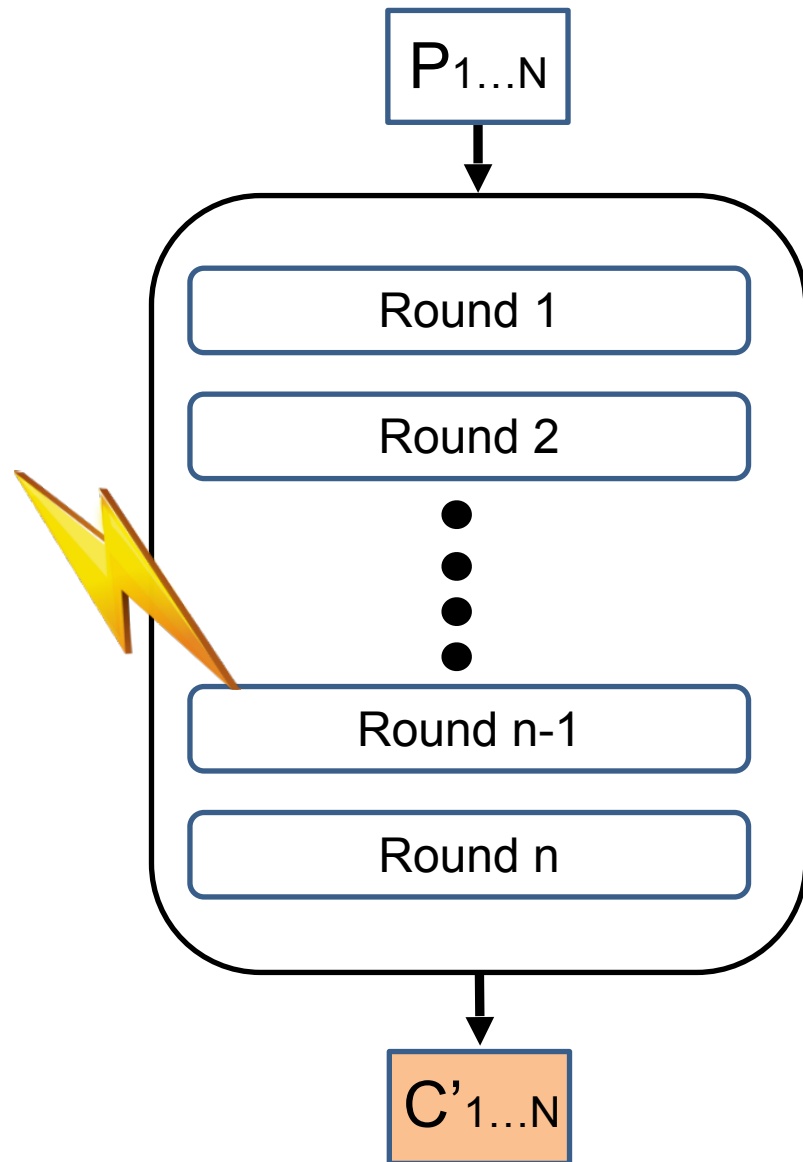
- Biased fault injection makes the state statistically biased.
- This bias is visible only for correct key guess.
- Such attacks require several faulty ciphertexts

Correct State	Faulty State
0 0 0	0 0 0
0 0 1	0 0 0
0 1 0	0 1 0
0 1 1	0 1 0
1 0 0	1 0 0
1 0 1	1 0 0
1 1 0	1 1 0
1 1 1	1 1 0

Example: Consider a stuck-at-0 fault at the LSB bit of the intermediate state. The faulty state can only assume even values.



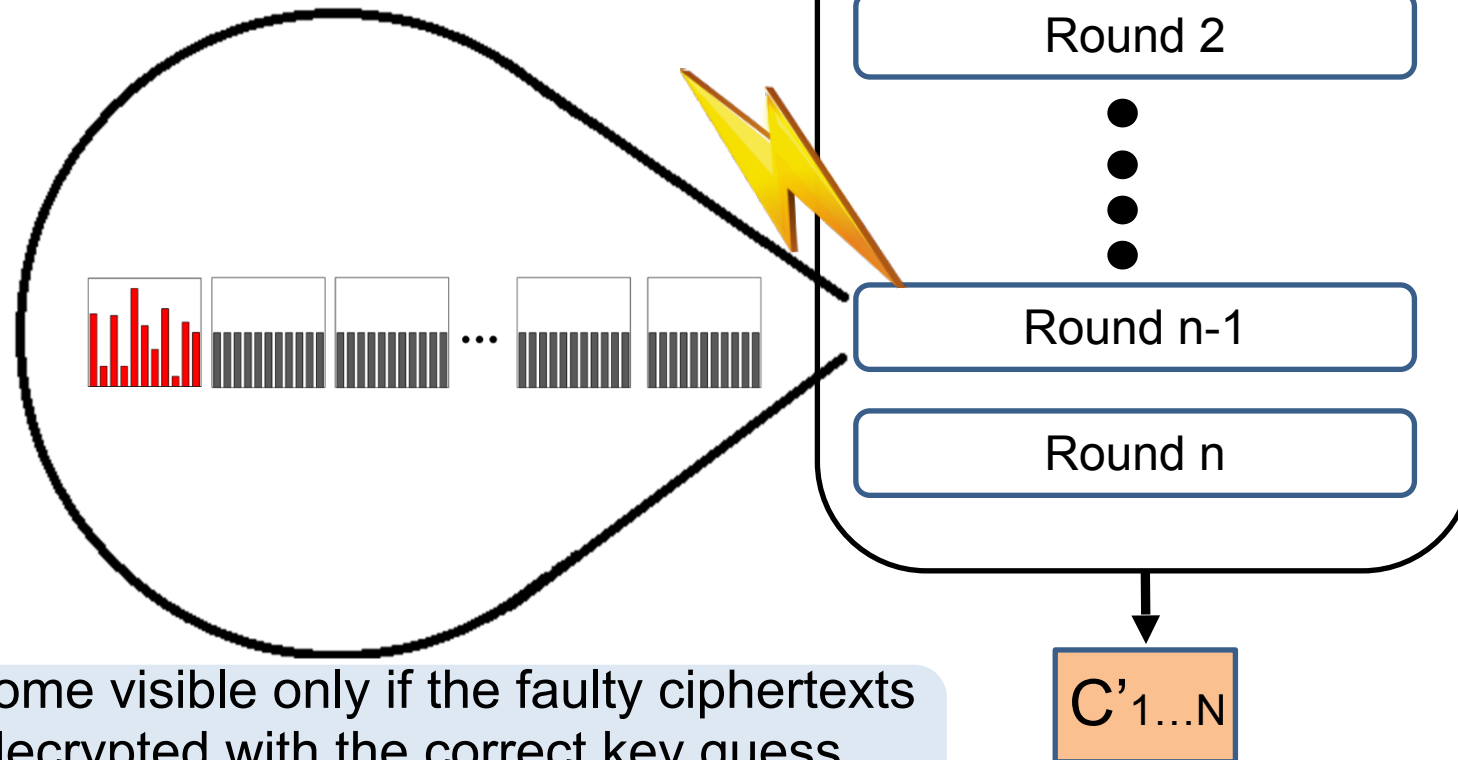
Statistical Fault Analysis (SFA)



$$SEI(K_g) = \sum_{i=0}^{255} \left(\frac{\#\{i \mid S^{-1}(C \oplus K_g) = i\}}{n} - \frac{1}{256} \right)^2$$

K_g with highest SEI is the correct key

Faults make the distribution of the state statistically biased at the injection point.



The bias become visible only if the faulty ciphertexts are partially decrypted with the correct key guess. For wrong guesses the bias disappears.

Statistical Fault Analysis (SFA)

- Similar to SCA attacks in some sense...
- **Key reason**: The bias is only observable while the key guess is correct.
- **Interesting fact**: You do not need to know the exact bias
 - The SEI metric does not require exact statistical distribution of the bias.
- **Issues**: Can only be applied at the last few rounds. Not like DFA which can also be applied at the middle rounds.