RECALL OF LAST SESSION

- **Physical implementation / Physical Attacks**
  - A famous example : Mifare break
- **Banking protocols**
  - B0'
  - EMV intro
- **fault attacks**
  - intro
  - RSA case

TODAY SESSION

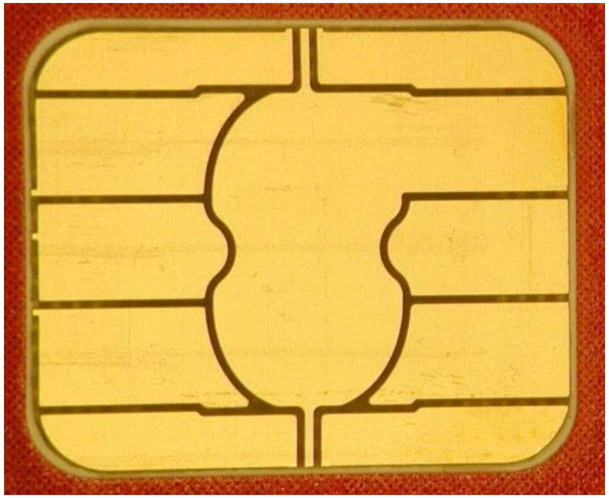
- **Fault attacks**
  - intro
  - RSA
    - Simple FA
    - Fast exponentiation
    - Efficient FA on CRT
  - DFA on symetric ciphers
    - DES
    - AES
  - FA on code execution
- **SCA**

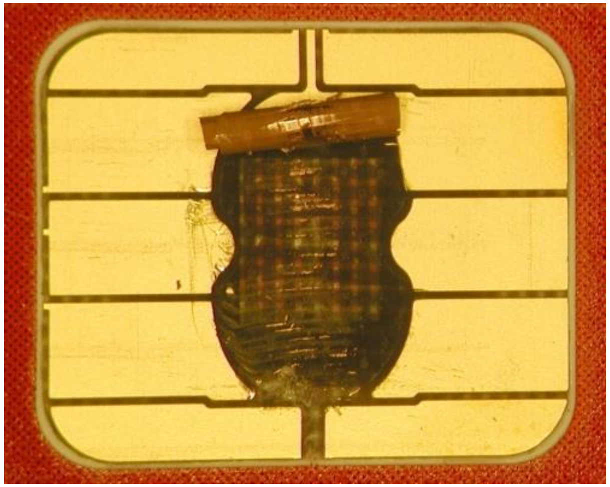
PERTURBATION ATTACKS

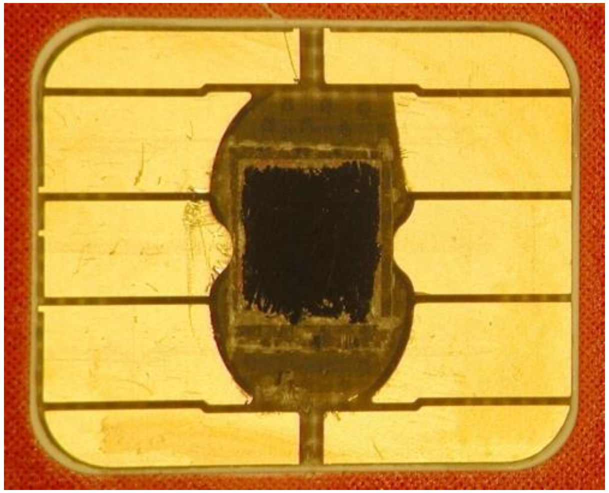
- Target
  - Modify a value read or stored in memory
  - Modify the program flow
    - •Skip an instruction
    - •Invert a test
    - •Generate a jump
  - Specifically attack a cryptographic algorithm
    - •Modify a parameter (secret key, public parameter...)
    - •Generate a fault which can be analyzed after the attack (DFA)
    - •Safe-error attack

LASER BENCH

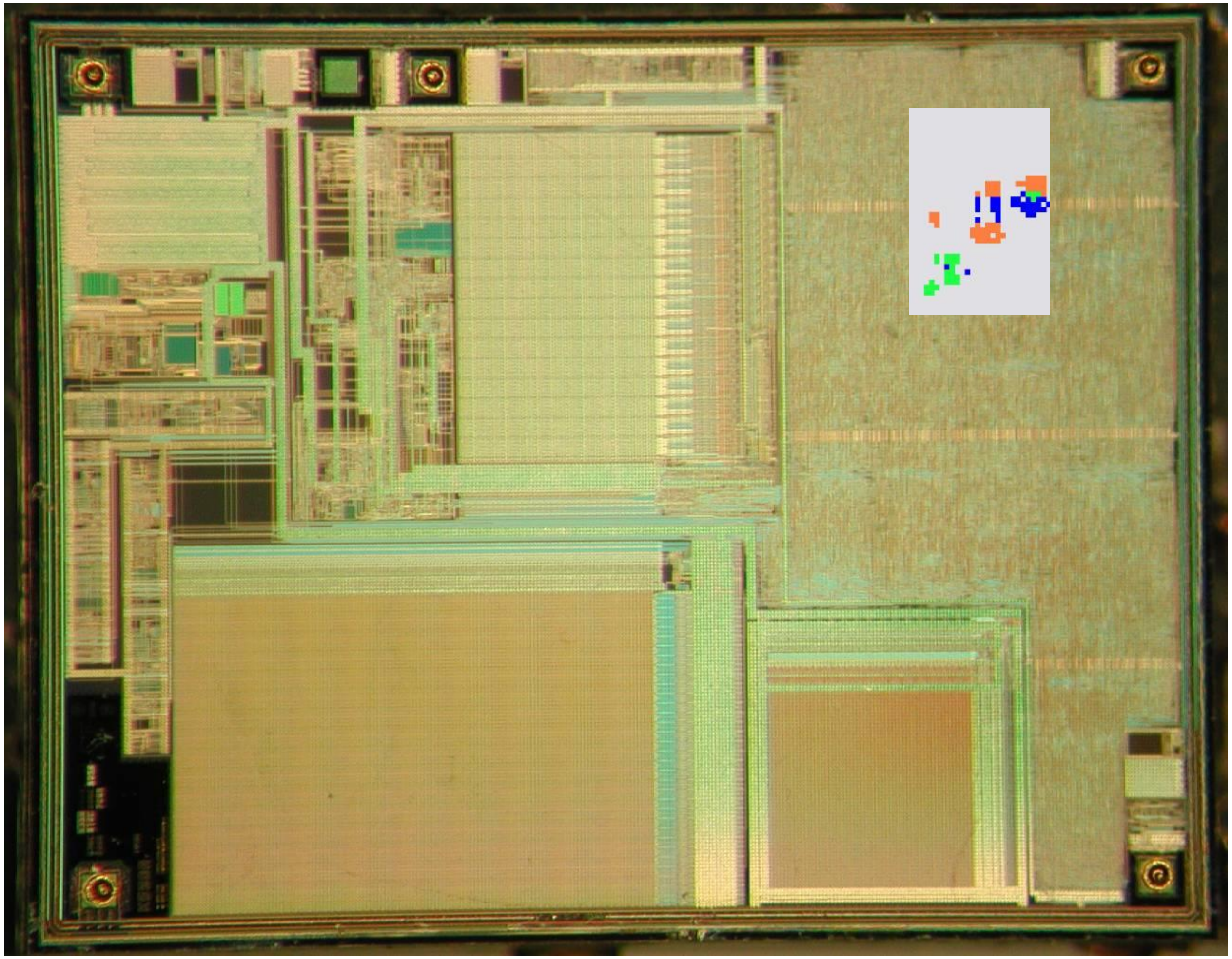
SAMPLES PREPARATION







CARTOGRAPHY



DFA: STANDARD RSA

- $S = x^d \bmod N$
- Private key $d = (d_{t-1}, \ldots, d_i, \ldots, d_1, d_0)$
- Fault on one bit of d

DFA: STANDARD RSA

- Private key $d = (d_{t-1}, \dots, d_i, \dots, d_1, d_0)$
- Fault on one bit of d
- $d' - d =$
  - 2^i if $d' > d$
  - -2^i if $d' < d$
- $S'/S =$
  - m^{2^i} if $d' > d$
  - M^{-2^i} if $d' < d$
- → retrieve of all the bits one by one

DFA: EXAMPLE ON RSA

- **How to efficiently crack RSA: Bellcore attack**
  - CRT recall
  - The attack
  - Example on a simplified RSA
  - Countermeasures
- **Safe error attack**
  - Example on RSA

DFA ON RSA

- Recall of Parameters
  - p, q prime numbers such as $N = p*q$
  - $d_p = d \bmod p-1$, $d_q = d \bmod q-1$, $q_{inv} = q^{-1} \bmod p$
  - $S_p = m^{d_p} \bmod p$, $S_q = m^{d_q} \bmod q$

DFA: CRT-RSA RECALL

- CRT for Chinese Remainder Theorem

- moduli $(m_1, m_2, \ldots, m_k)$ where all mi are mutually prime

- set $M = m_1 \times m_2 \times \ldots \times m_k$

- For all residues $(a_1, a_2, \ldots, a_k)$

- There exists only one x (mod M) such as

$$x \bmod m_1 = a_1$$

$$x \bmod m_2 = a_2$$

$$x \bmod m_k = a_k$$

The theorem also gives a formula to compute it

$$x = (\textstyle\sum_{i=1,k} a_i * v_i * M/n_i)$$

With $v_i = (M/n_i)^{-1} \bmod n_i$

DFA: CRT-RSA (RECALL WITH 2 PRIMES)

- CRT with 2 primes p, q

  $N/p = q$

  $N/q = p$

For any (a1, a2), there exists only one x (mod M) such that

- $x \bmod p = a_1$
- $x \bmod q = a_2$

And $x = a_1 * q * (q^{-1} \bmod p) + a_2 * p * (p^{-1} \bmod q)$

- Why CRT?
  - Speed: 4X times faster
  - Memory: smaller intermediate values

DFA: CRT-RSA (BELLCORE ATTACK)

- And $x = a_1 * q * (q^{-1} \bmod p) + a_2 * p * (p^{-1} \bmod q)$
  - p, q prime numbers such as $N = p*q$
  - $d_p = d \bmod p\text{-}1$, $d_q = d \bmod q\text{-}1$, $q_{inv} = q^{-1} \bmod p$
  - $S_p = m\text{^}d_p \bmod p$ (= m^d mod p)
  - $S_q = m\text{^}d_q \bmod q$ (= m^d mod q)

$$m\text{^}d \bmod N = q(q^{-1} \bmod p)*Sp + p(p^{-1} \bmod q)*Sq$$
$$= S_q + [(S_p - S_q)*q_{inv} \bmod p]*q$$

(Garner's formula)

DFA: CRT-RSA (BELLCORE)

- **Attack**
  - Induce a fault on Sp
- **What happens ?**
  - $S' - S = q*[(S_p{}'-S_q)*q_{inv} \bmod p - (S_p-S_q)*q_{inv} \bmod p]$
  - $q = gcd(N, S' - S)$
  - $q = gcd(N, m-(S'\char`^e \bmod N))$

DFA: CRT-RSA (BELLCORE)

Example : p = 7, q = 13, qinv = 6, d = 3, x=41

$\Rightarrow$ **Compute :**

- **Sp, Sq, S**
- **Induce a fault on Sp (ex: Sp' = 2)**
- **Compute S'**
- **Find q (hint: q=13)**

BELLCORE ATTACK SUMMARY

- **Any** fault on one of the 2 exponentiations
- A pair of x^d mod N and (x^d mod N)*
- Then N can be factored computing a simple gcd

SAFE ERROR ATTACK: A COMPREHENSIVE EXAMPLE

- 1. How to compute efficiently RSA
- 2. A small introduction to Side Channel Attacks
- 3. Countermeasure
- 4. Safe Error Attack

SAFE ERROR ATTACK WITH THE EXAMPLE (1/4)

- How to compute efficiently

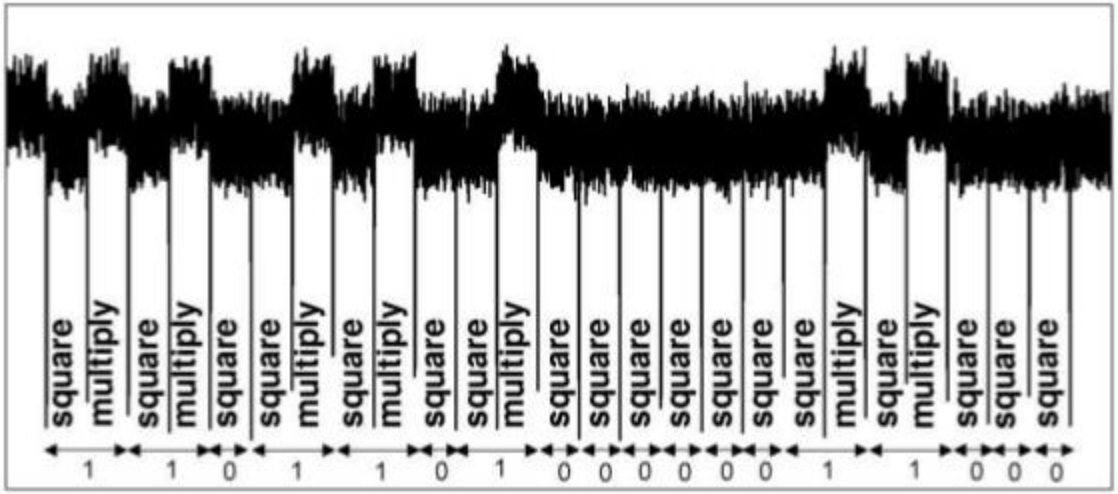
  x^d mod N

```
S = M
for i from 1 to n − 1 do
    S = S * S (mod N)
    if d_i = 1 then
        S = S * M (mod N)
return S
```

Fig. 3. Binary version of Square-and-Multiply Exponentiation Algorithm

SAFE ERROR ATTACK WITH THE EXAMPLE (2/4)

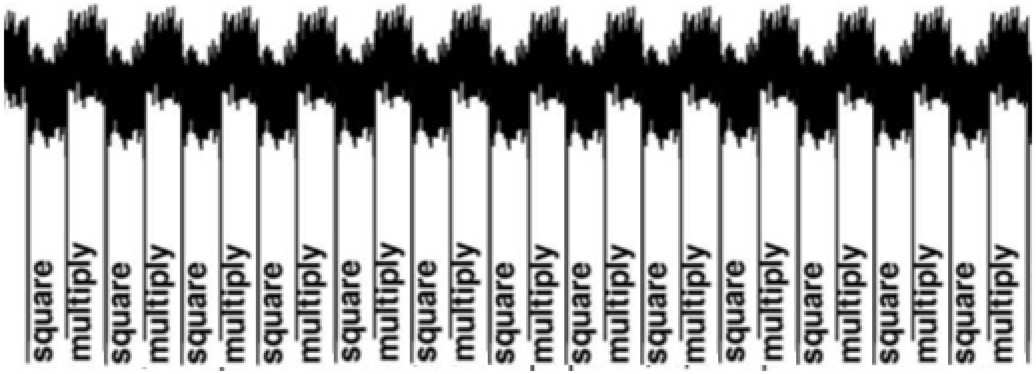
- A very fast introduction to SCA
- The power comsumption of a square is different from the power consumption of a multiplication



SAFE ERROR ATTACK WITH THE EXAMPLE (3/4)

- **Countermeasure**

$S = M \quad R = M$
for i **from** 1 **to** $n-1$ **do**
  $S = S * S \ (\mathrm{mod} N)$
  if $d_i = 1$ **then**
    $S = S * M \ (\mathrm{mod} N)$
  else then
    $R = S * M \ (\mathrm{mod} N)$
return S

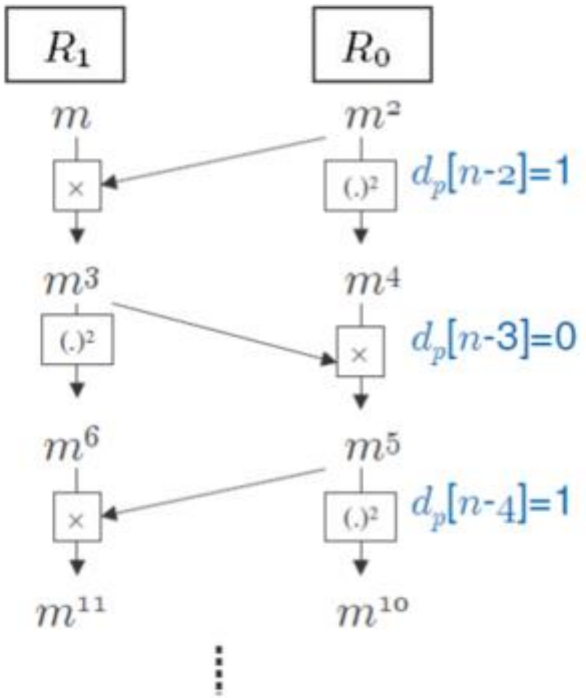


- **Defeating CM**
  - Fault on mult

A SPA/FA EXPONENTIATION: MONTGOMERY LADDER

$R_0 = 1$; $R_1 = M$
for i **from** 0 **to** $n-1$ **do**
  if $d_i = 0$ **then**
    $R_1 = R_0 * R_1 \pmod N$
    $R_0 = R_0 * R_0 \pmod N$
  else [**if** $d_i = 1$] **then**
    $R_0 = R_0 * R_1 \pmod N$
    $R_1 = R_1 * R_1 \pmod N$
return R_0

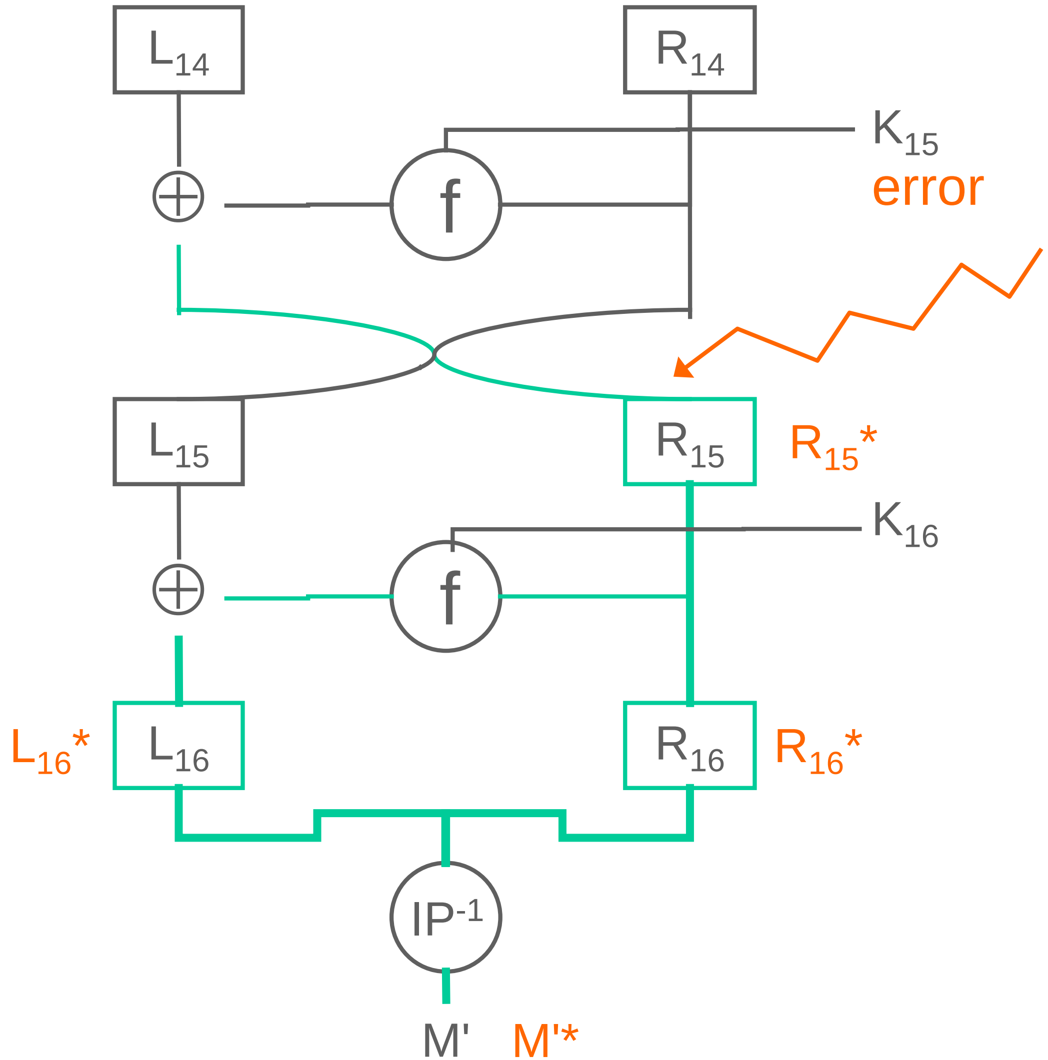
Fig. 4. Balanced Montgomery Powering Ladder



SAFE ERROR ON SYMMETRIC CIPHER

- **Fault on registers**

DFA ON DES



DFA ON DES

With a fault on R15, we have

- $L_{16} = f(R_{15},k_{16})$ xor L_{15}
- $L_{16}^* = f(R_{15}^*,k_{16})$ xor L_{15}
- $R_{15} = R_{16}$ and $R_{15}^* = R_{16}^*$

Thus we obtain the differential

- $f(R_{15}, k_{16})$ xor $f(R_{15}^*, k_{16}) = L_{16}$ xor L_{16}^*
- This equation holds for each Sbox independantly

$fi(R_{15}, k_{16,i})$ xor $fi(R_{15}^*, k_{16,i}) = (L_{16}$ xor $L_{16}^*)i$

DFA ON DES

The attack:

- For a couple (c,c*)

- Apply the following algorithm:

for each i in {1,..,8}, try a guess $k_{16,i}$

Check if the equation holds.

$fi(R_{15}, k_{16,i})$ xor $fi(R_{15}', k_{16,i}) = (R_{16}$ xor $R_{16}')i$

If it holds , $k_{16,i}$ is found
if not the c* is discarded

=> 8 faults are enough to retrieve k16.

DFA ON DES

- **Inject fault on last round of DES**
- **=> This leads to efficient DFA**

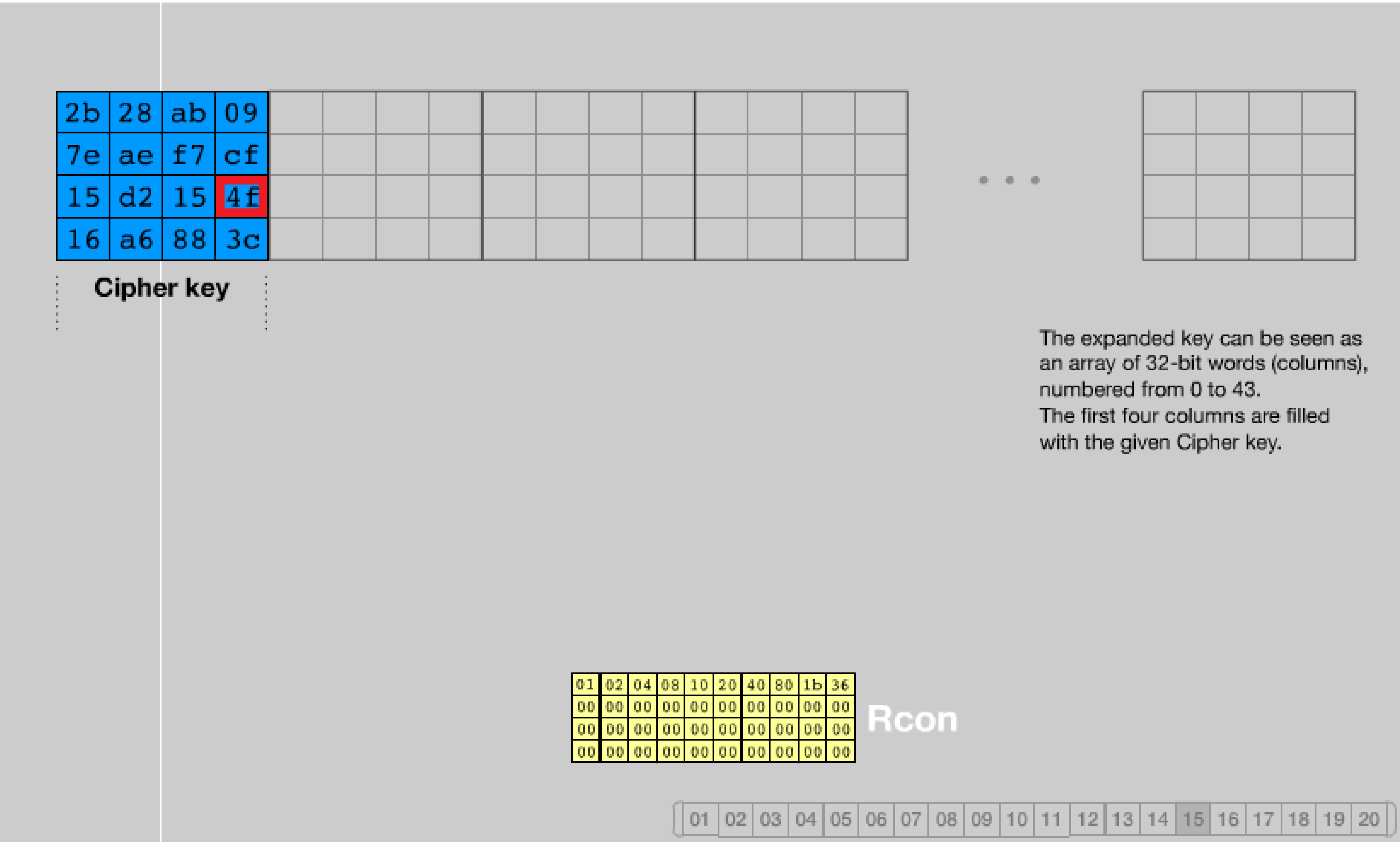
- **Other attacks exist on previous rounds**
  - A bit more complex but also very efficient
  - Attacks exist from round 7 to 15
    - The 12th round attack requires 20 faulties
    - The 11th round attack requires 800 faulties
  - Based on statistical distribution (SEI / Likelihood)

DFA ON AES

- **An easy example on AES**
  - Giraud attack’s : DFA on AES (2007)
  - Fault on key schedule of the last round
  - Fault on M9 / M8

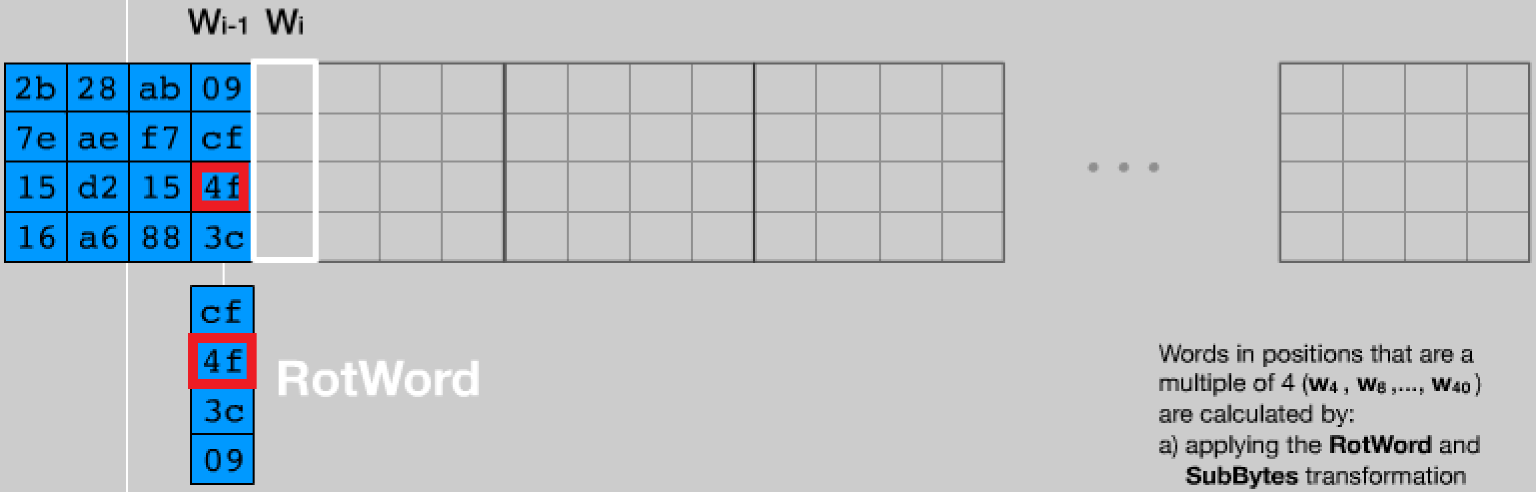
DFA ON AES

Key Schedule



DFA ON AES

Key Schedule



Words in positions that are a multiple of 4 (w_4, w_8,..., w_{40}) are calculated by:

a) applying the **RotWord** and **SubBytes** transformation to the previous word w_{i-1}.

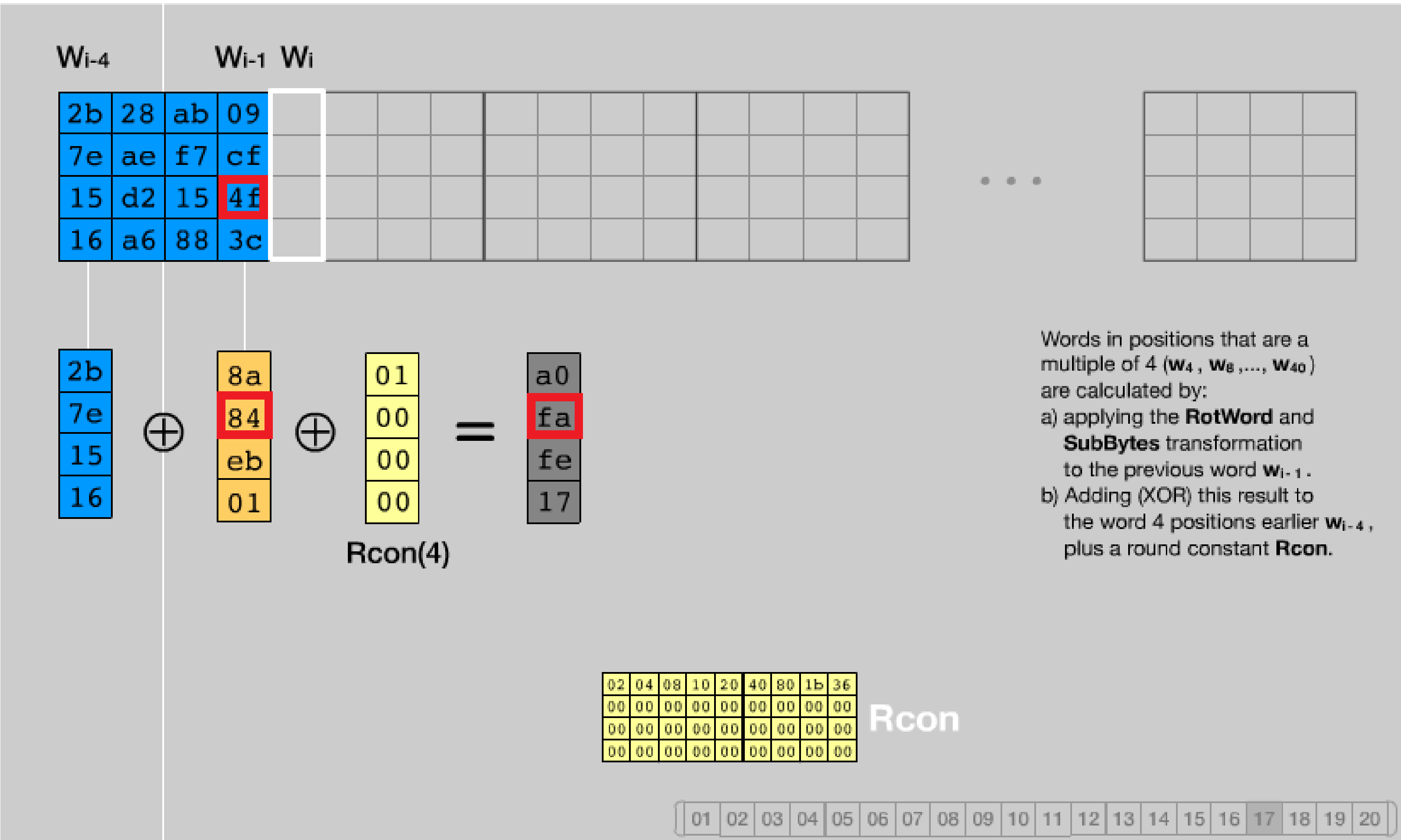
01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Rcon

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20

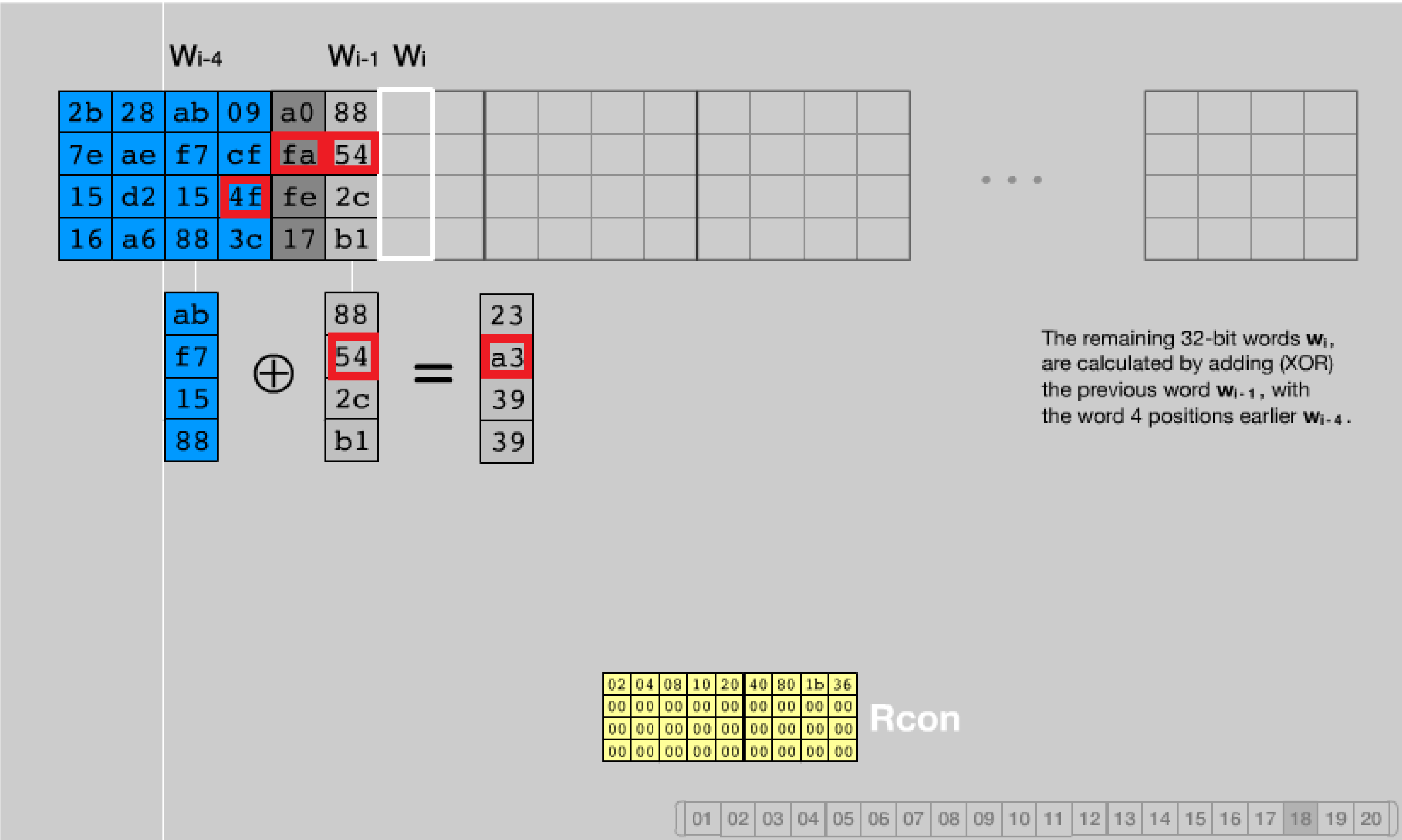
DFA ON AES

Key Schedule



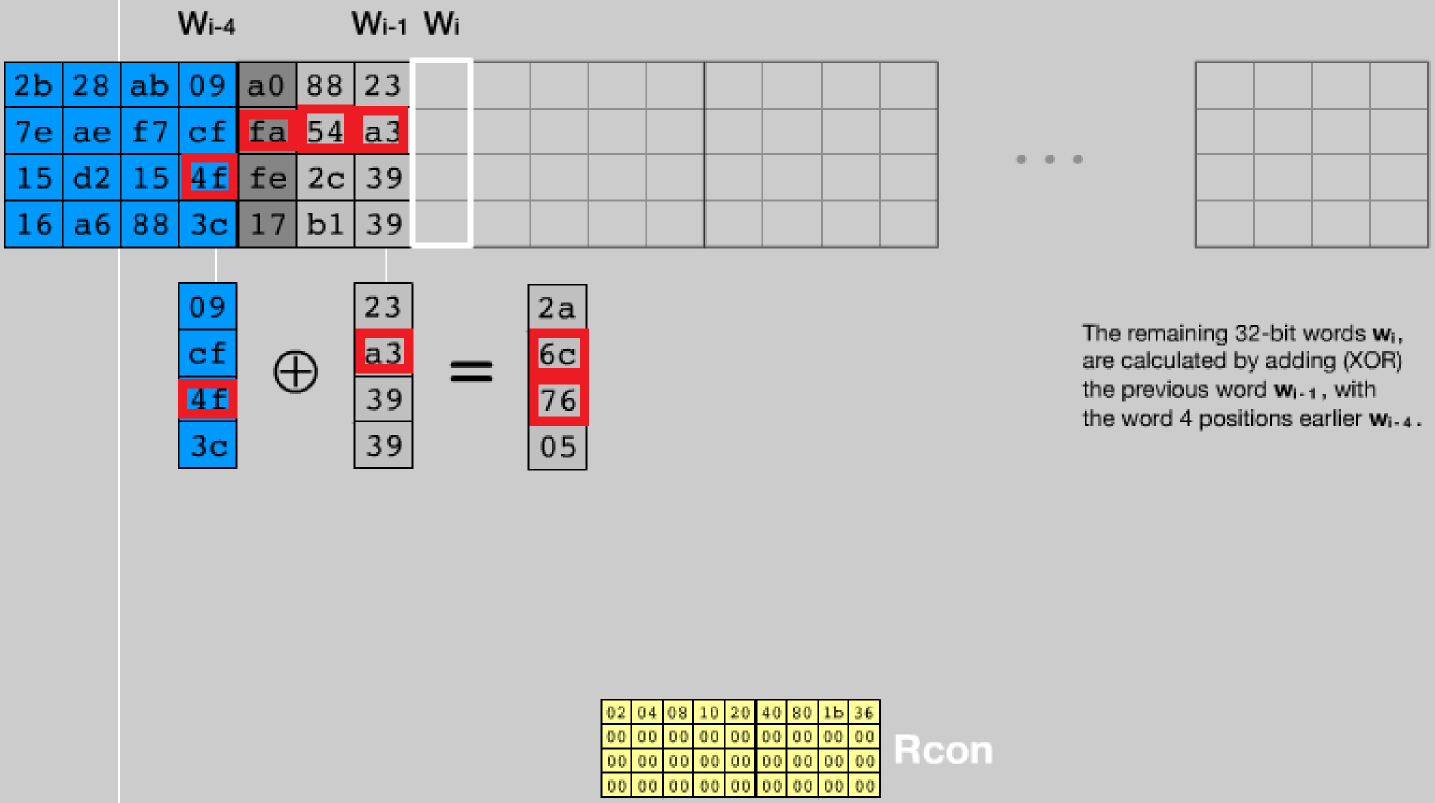
DFA ON AES

Key Schedule



DFA ON AES

Key Schedule



The remaining 32-bit words w_i, are calculated by adding (XOR) the previous word w_{i-1}, with the word 4 positions earlier w_{i-4}.

DFA ON AES

Key Schedule

2b	28	ab	09	a0	88	23	2a
7e	ae	f7	cf	fa	54	a3	6c
15	d2	15	4f	fe	2c	39	76
16	a6	88	3c	17	b1	39	05

. . .

02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00

Rcon

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20

DFA ON AES

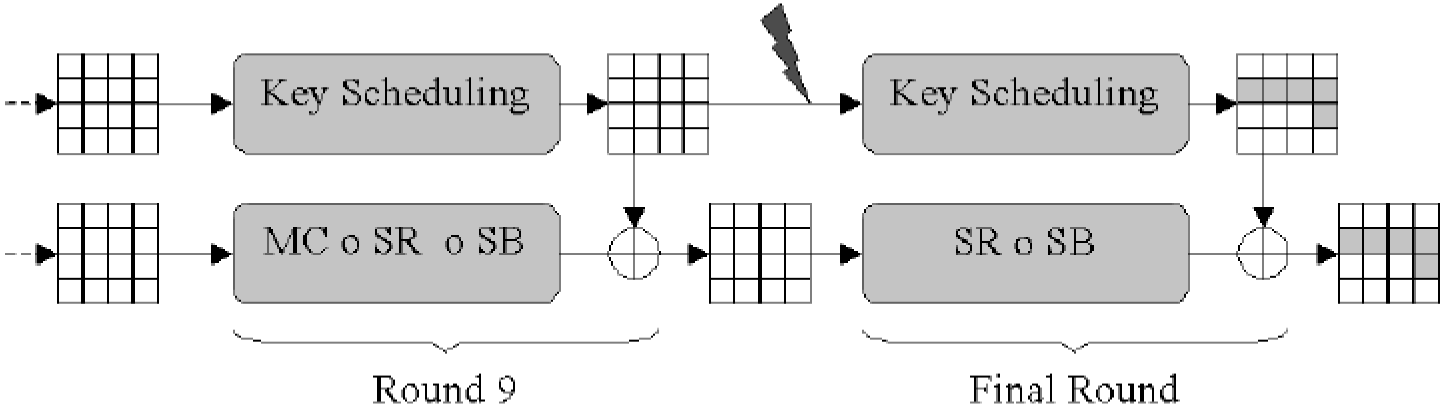


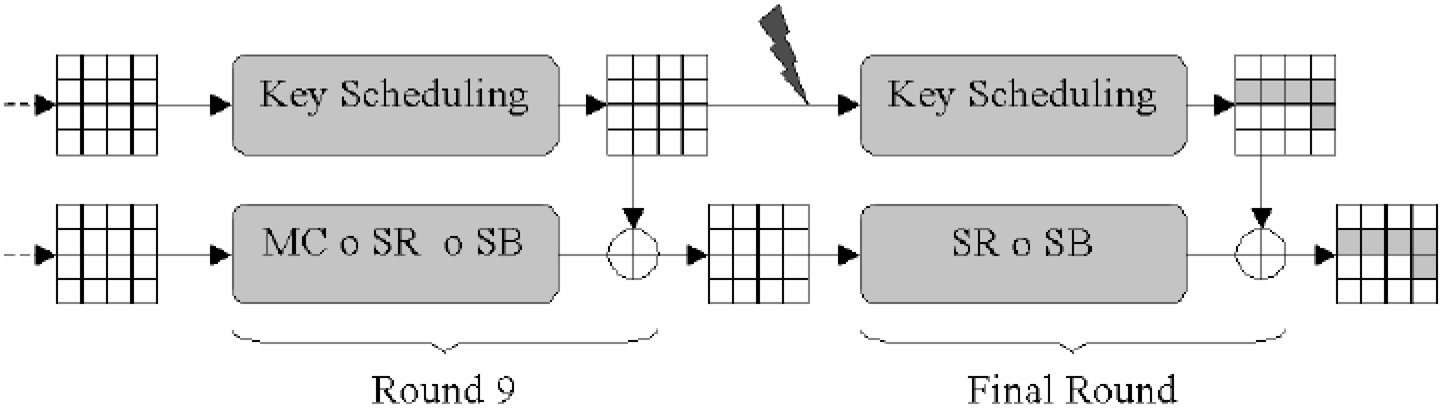
Fig. 3. Fault on the 14^{th} byte of the penultimate round key K^9.

1 fault induced in K9 (specific byte) implies 5 errors in the ciphertext

DFA ON AES

2b	28	ab	09	a0	88	23	2a
7e	ae	f7	cf	fa	54	a3	6c
15	d2	15	4f	fe	2c	39	76
16	a6	88	3c	17	b1	39	05

- **The error can be retrieved**



- **C14 xor D14 = e**
- **(D14 is the value of the faulty ciphertext)**

DFA ON AES

The datapath is unchanged except the last addkey

- => The difference between C and D is only from the last addroundkey
- The equations are

$$C_k \oplus D_k = SubByte(K_j^9) \oplus SubByte(K_j^9 \oplus e_j) \quad (14)$$

We know the value of $C_k \oplus D_k$ and the value of e_j. So, we search the possible values $x \in \{0, ..., 255\}$ which satisfy the equation

$$C_k \oplus D_k = SubByte(x) \oplus SubByte(x \oplus e_j) \quad (15)$$

- 4 bytes of key can be retrieved with this method

DFA ON AES

- **4 bytes is not enough**
- **Several other DFA allows to retrieve the whole key**
  - Giraud attack
  - Piret & Quisquater :
    - 1 faulty byte before the last mixcolumns => 4 bytes in output
    - 1 faulty byte one round before => 2 faulties to retrieve the whole key
- Mukhopadhyay 2011: 1 fault / 2^30 time complexity