Active Physical Attacks

IT Security

2014

Erich Wenger

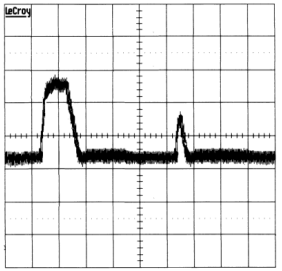
Slides by Stefan Mangard

Active Physical Attacks

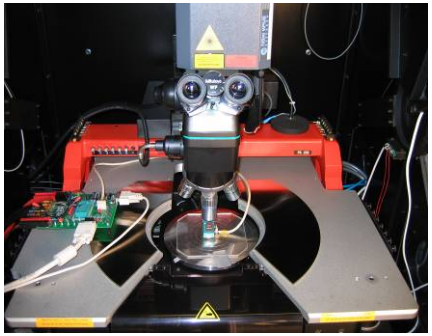
- Digital circuit require certain operating conditions to work properly
  - temperature range
  - operating frequency
  - supply voltage
  - ...
- By changing the operating conditions an attacker can induce a fault in the circuit
- This is the goal of active physical attacks, which are often also called fault attacks

The Two Most Popular Fault Attacks

- Spike/Glitch Attacks (an active non-invasive attack)
  - Disturb the power supply lines or the I/O lines



- Light Attacks (an active semi-invasive attack)
  - Transistors can be switched by light from the front side or backside of the chip; local attacks are done with focused lasers; global attacks can also be done using other light sources



Countermeasures Against Active Physical Attacks

There are essentially two approaches to counteract active physical attacks:

- Sensor-based approaches: The goal is to detect the fault induction (temperature sensor, light sensor, voltage sensor, …)
- Error-detection based approaches: The goal is to detect the error that is the consequence of the fault induction

In this course, we focus on error-detection techniques. This is the more generic approach that can also be implemented in software.

Content of Chapter 10

- Part 1: Fault Attacks on RSA
- Part 2: Fault Attacks on AES

Part 1

Fault Attacks on RSA

Example I: Safe-Error-Attack on a Simple Square-and-Multiply Implementation of RSA

- The square-and-always-multiply algorithm for $m^d \bmod N$

```
Z := 1; [d = (d_{n-1}…d_0)_2]
for i:=n-1 to 0 do
      Z := Z^2 mod N
      if d_i=1 then Z := Z · m;
      if d_i=0 then B := Z · m;
end;
return Z;
```

  can be used as a countermeasure against simple power analysis (see Example I, Part 5, Chapter 9)

- Attack: At step i, disturb the second multiplication
  - if $d_i=0$, then the final signature will be correct,
  - if $d_i=1$, then the final signature will be wrong.
- This allows to reveal the key bit by bit; the attack only needs the information: "Signature ok/wrong".

Example II: Bellcore Attack on RSA with CRT (1/3)

- Assume an RSA signature is created using the Chinese-Remainder Theorem:

$$S_p := m^{dp} \bmod p \quad [= m^{(d \bmod (p-1))} \bmod p]$$
$$S_q := m^{dq} \bmod q \quad [= m^{(d \bmod (q-1))} \bmod q]$$

Calculating S from $S_p \bmod S_p$ (Garner's Formula)

$$S := S_q + q \cdot [(S_p - S_q) \cdot q_{inv} \bmod p]$$
$$q_{inv} := q^{-1} \bmod p$$

Calculating S from $S_p \bmod S_p$ (Textbook method)

$$S := a \cdot S_p + b \cdot S_q \bmod N$$
$$a := q \cdot (q^{-1} \bmod p)$$
$$b := p \cdot (p^{-1} \bmod q)$$

- In any case, the computation of the signature involves the secrets p, q, dp and dq

Example II: Bellcore Attack on RSA with CRT (2/3)

- Attack Description:
  - Disturb one of the two partial exponentiations, such that either S_p or S_q results in a different value S_p' or S_q'.
  - Take the erroneous Signature S' and the correct Signature S and compute
    $$x = \gcd(S'-S, N).$$
  - If x>1 then x=p or x=q.
- Explanation: Assume S_p is disturbed and we have S_p' instead of S_p. Then

$$S'-S \equiv (a \cdot S_p' + b \cdot S_q) - (a \cdot S_p + b \cdot S_q)$$
$$\equiv a \cdot (S_p'-S_p) \bmod N \equiv q \cdot (q^{-1} \bmod p) \cdot (S_p'-S_p) \bmod N$$
$$= q \cdot y.$$

Therefore
$$\gcd(S'-S, N) = \gcd(q \cdot y, N) = q.$$

Example II: Bellcore Attack on RSA with CRT (3/3)

- For the classical Bellcore attack one needs a correct and an erroneous signature of the same message.

- Another possibility is: One knows an erroneous signature S' of the message m and m itself. Then
  $\gcd((S'^e \bmod N) - m, N) = p$ or q.

- Explanation: It holds that

$$S \equiv a \cdot S_p + b \cdot S_q \bmod N \equiv S_p \bmod p \equiv S_q \bmod q$$

Assume S_p is disturbed (S_p' instead of S_p); Then it holds that

$$m \equiv S^e \not\equiv S_p'^e \bmod p$$
$$m \equiv S^e \equiv S_q^e \bmod q$$

Hence, q divides $(S'^e \bmod N) - m$, but not p;

Consequently, $\gcd((S'^e \bmod N) - m, N) = q$

Countermeasures for RSA

- Obvious countermeasures:
  - Compute twice and compare.
  - Test whether the Signature is correct by verifying
    $S^e \bmod N == m$.

  But both countermeasures have the disadvantage that the computation time grows significantly, and more…

- Advanced countermeasures use redundancy mechanisms directly in the computation
  - There are many pitfalls
  - Very active area of research

Part 2

Fault Attacks on AES

Generic Fault Attack on Cryptographic Algorithms

- Assume the attacker can change bits of data values selectively to a defined value (e.g. to 0)

- Trivial attack in this fault model
  - The attacker performs a reference encryption of some unknown plaintext and an unknown key.
  - The attacker repeats this encryption and attacks the implementation by setting one key bit to 0
  - If the output of the encryption changed compared to the reference encryption, the attacked key bit is 1; otherwise it is 0
  - The attacker can repeat this for all bit positions

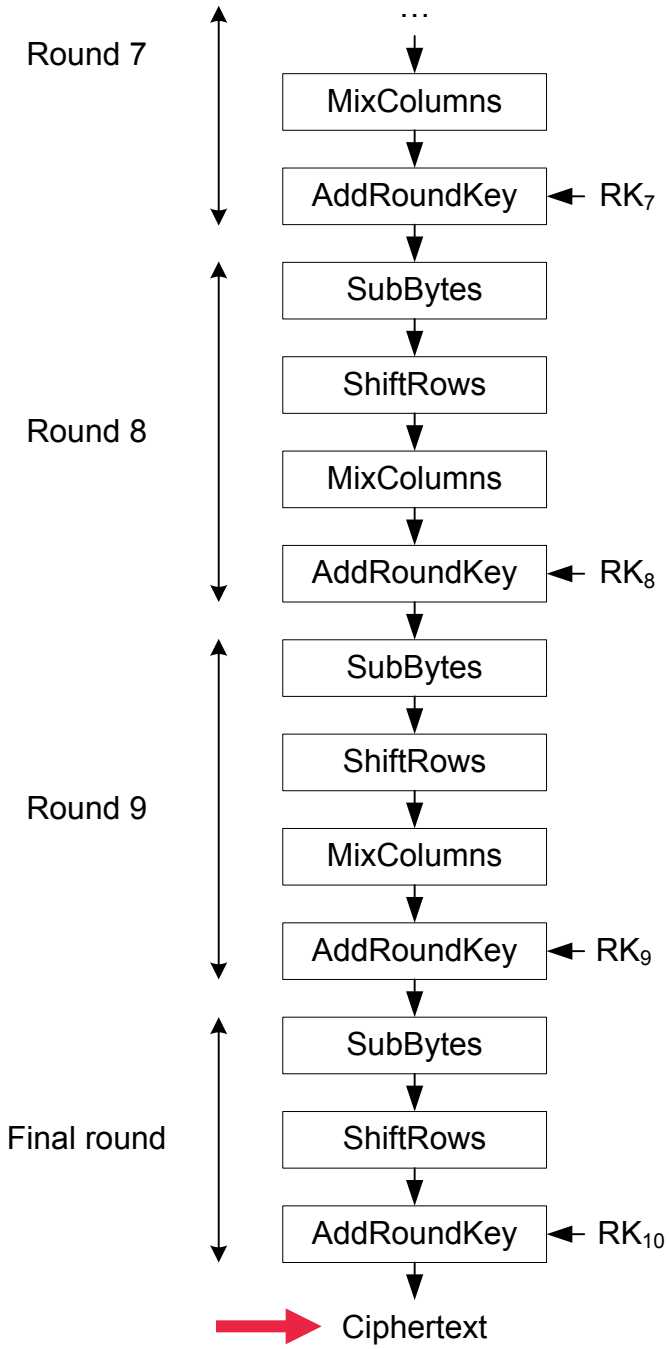
This algorithm is very simple. However, the drawback of this method is that the attacker needs to be able to induce very precise faults

When knowing the attacked algorithm, properties of this algorithm can be exploited to obtain the key in a much simpler way

Fault Attacks on AES

Can an attacker learn something about the key by changing the ciphertext?

NO



Fault Attacks on AES

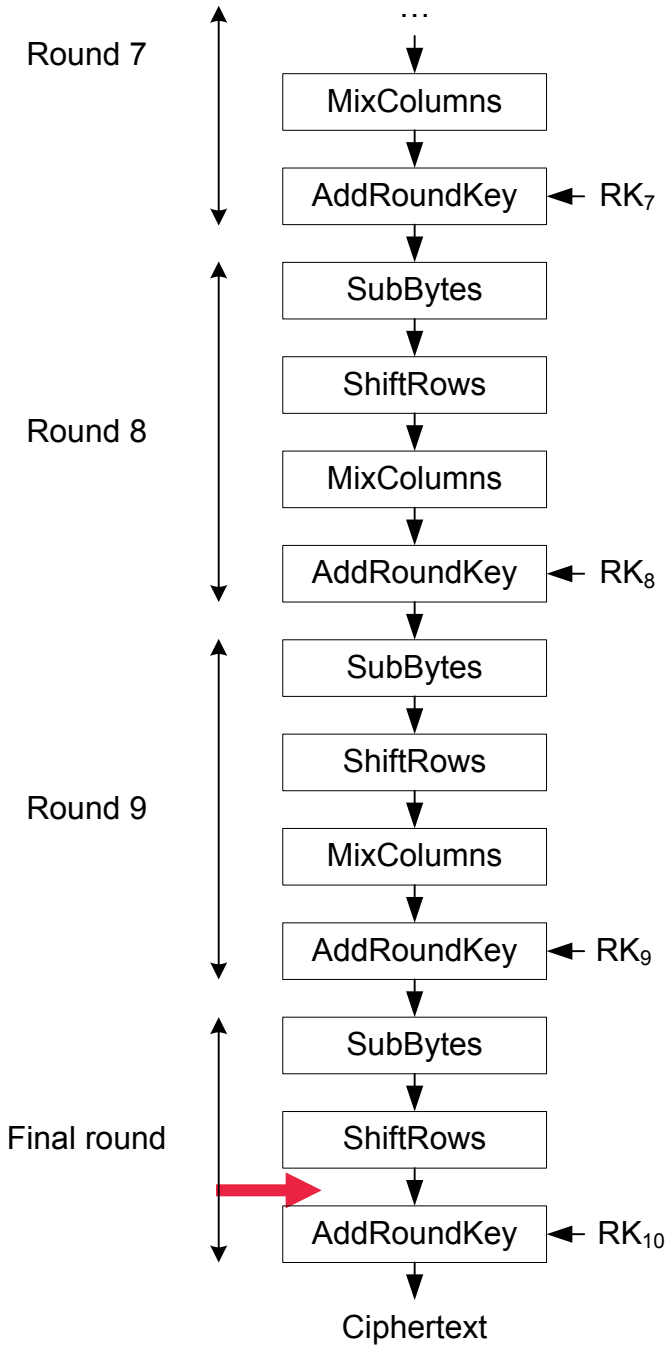
Can an attacker learn something about the key by changing the input of Addroundkey?

It depends on the fault model

Fault model 1:

- The attacker can toggle one or multiple bits:

  The attacker learns nothing, because the ciphertext changes exactly at exactly the same bit positions as the input of AddRoundKey – this independent of the key



Fault Attacks on AES

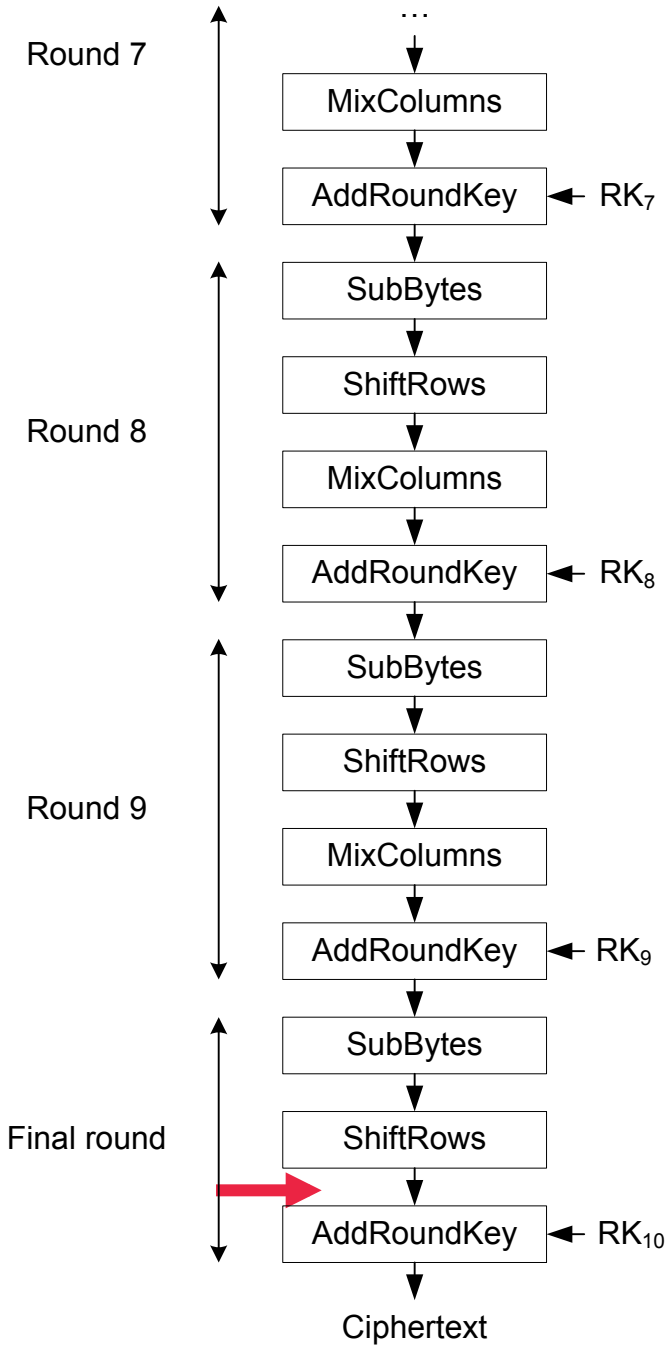
Can an attacker learn something about the key by changing the input of Addroundkey?

It depends on the fault model

Fault model 2:

- The attacker can set bits to certain values:

  The attacker can learn the key just like in the generic fault attack on the key



Fault Attacks on AES

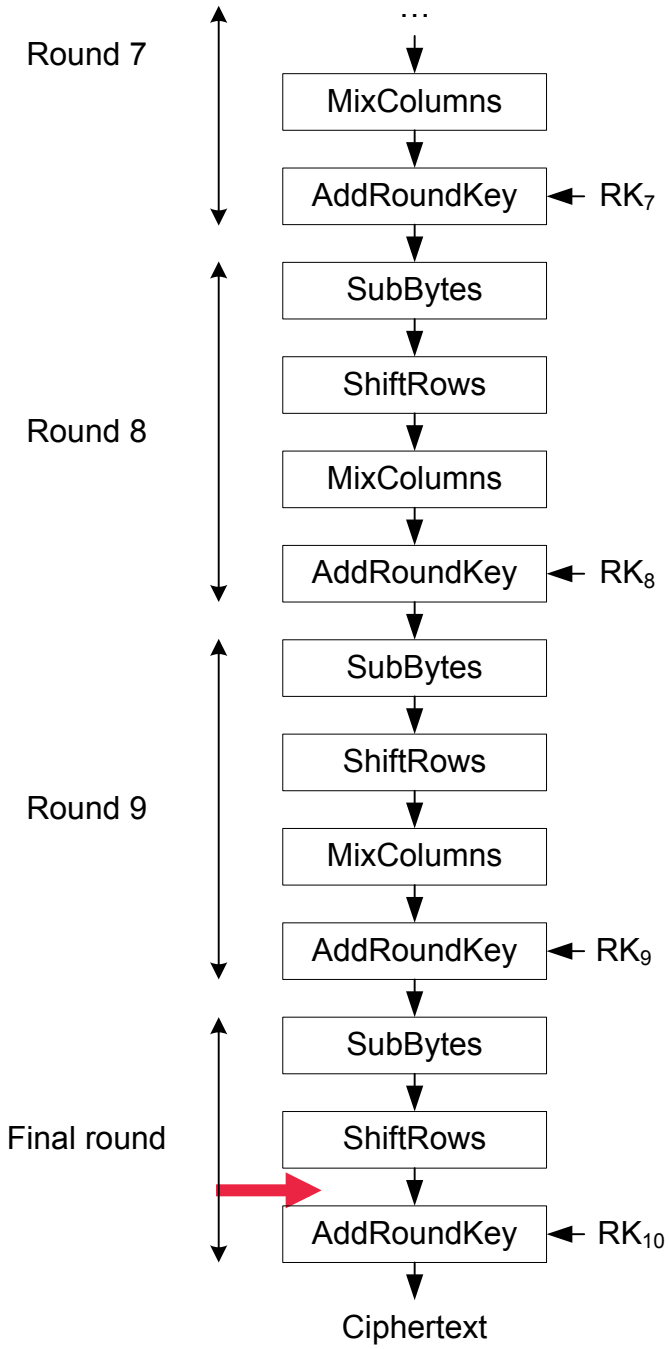
Can an attacker learn something about the key by changing the input of Addroundkey?

It depends on the fault model

Fault model 3:

- The attacker can set bits to unknown random values:

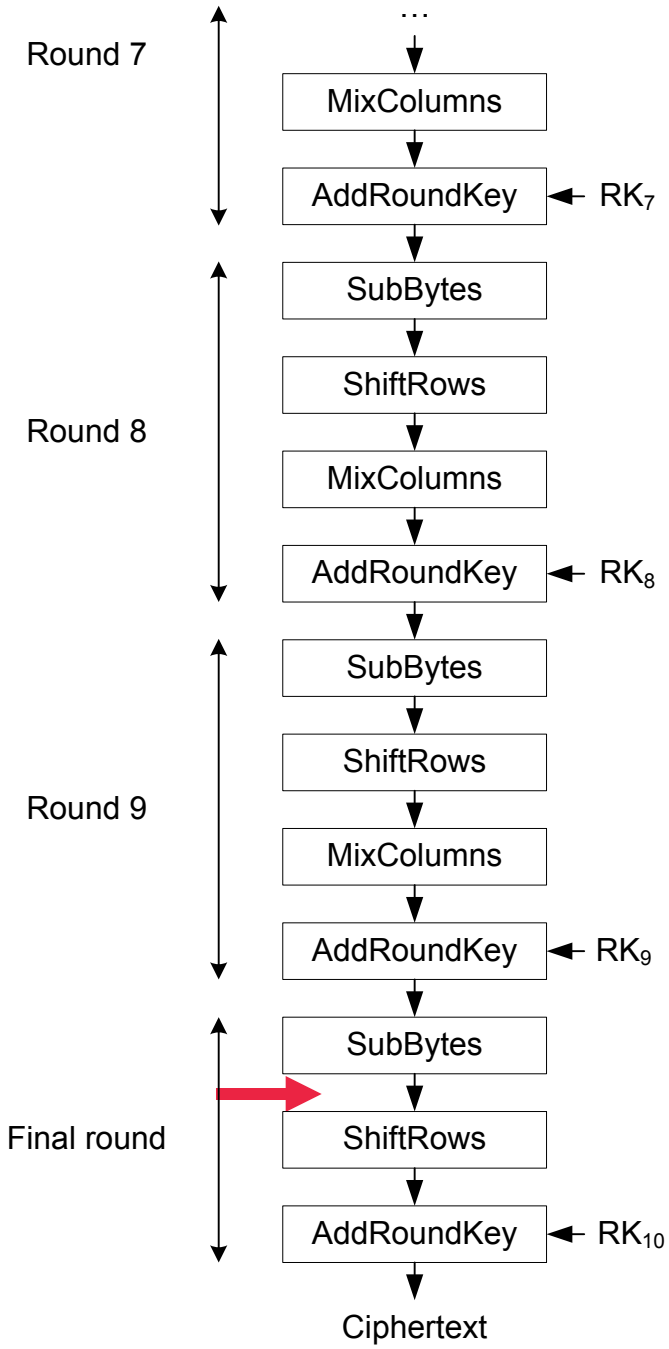
  The attacker again learns nothing as long as the induced fault is unknown



Fault Attacks on AES

Can an attacker learn something about the key by changing the input of Shiftrows?

The same statements hold as for the attack before AddRoundkey



Fault Attacks on AES

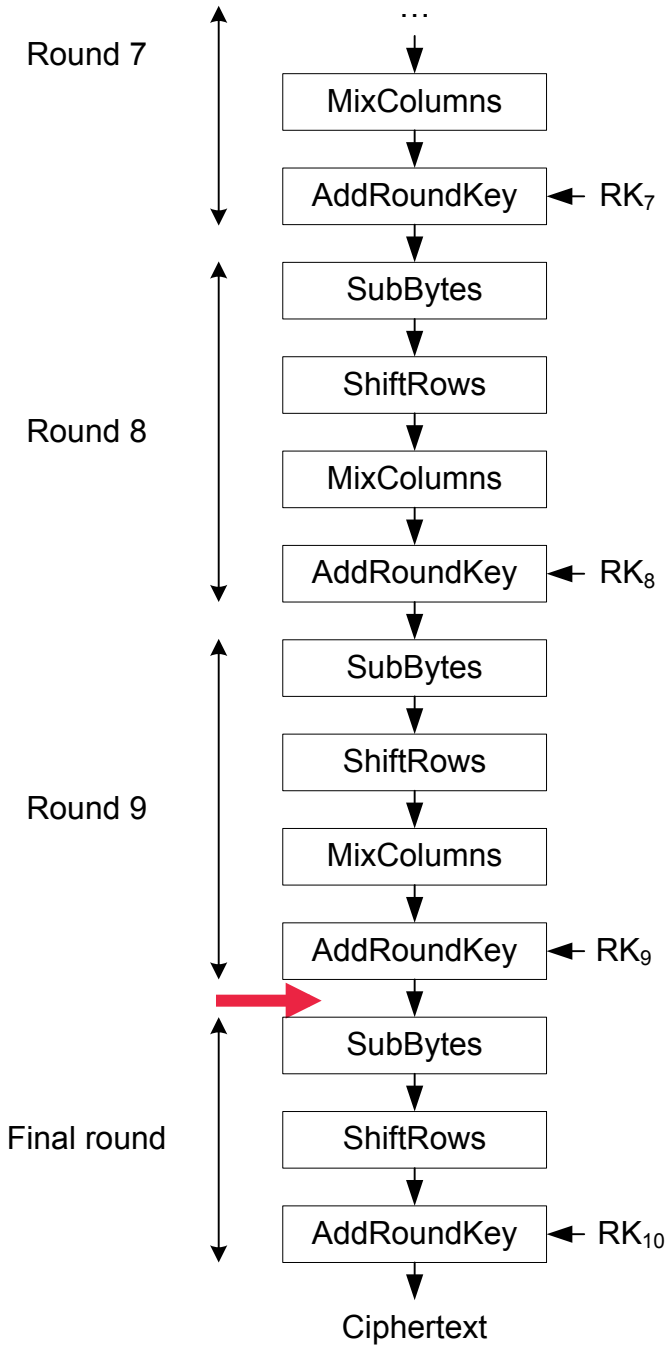
Can an attacker learn something about the key by changing the input of SubBytes?

It depends on the fault model

Fault model 1:

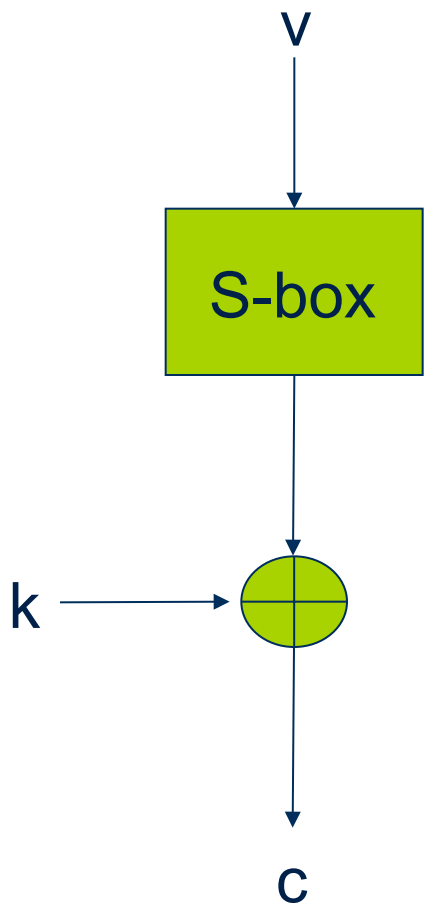
- The attacker can toggle a bit and knows the bit position:

  Yes, an attack is possible



Fault Attack that toggles a bit of an input of an Sbox

- Assume that the attacker is able to toggle the LSB of an Sbox input
- Assume the attacker performs for each plaintext one correct encryption and one faulty encryption
- For each pair of ciphertext and faultytext, the attacker can calculate the Sbox input v based on the ciphertext and the faultytext and he can check for which key there is only a difference in the LSB of the Sbox input



Example

At the attacked byte position the correct ciphertext output is 1a; the faulty output is 99; Based on this the attacker can calculate the Sbox input for the different keys and check whether there is only a difference in the LSB or not

```
                          k= 0   1   2   3   4   5   6   7   8   9
-------------------------------------------------------------------

C= 1a : S^-1(C xor k):

F= 99 : S^-1(F xor k):
```

Example

At the attacked byte position the correct ciphertext output is 1a; the faulty output is 99; Based on this the attacker can calculate the Sbox input for the different keys and check whether there is only a difference in the LSB or not

```
                          k= 0  1  2  3  4  5  6  7  8  9
-------------------------------------------------------
C= 1a : S^-1(C xor k): 43 44 34 8e e9 cb c4 de 39 82
F= 99 : S^-1(F xor k): f9 e2 e8 37 75 1c 6e df ac 96
```

Typically, only few key candidates remain for each pair of ciphertext and faultytext. Two pairs of ciphertext and faultytext are usually sufficient to determine one key byte

Fault Attacks on AES

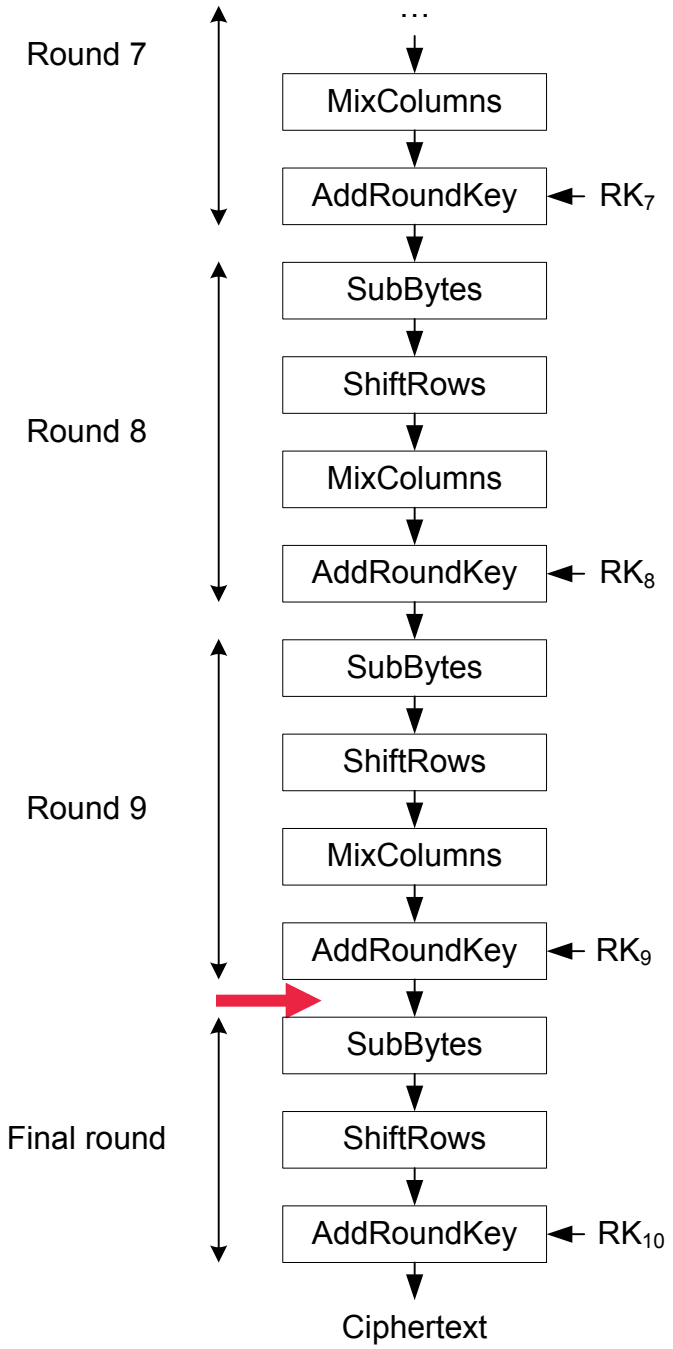
Can an attacker learn something about the key by changing the input of SubBytes?

It depends on the fault model

Fault model 2:

- The attacker can insert a unknown random fault:

  Attack is not possible because the random fault is again unknown

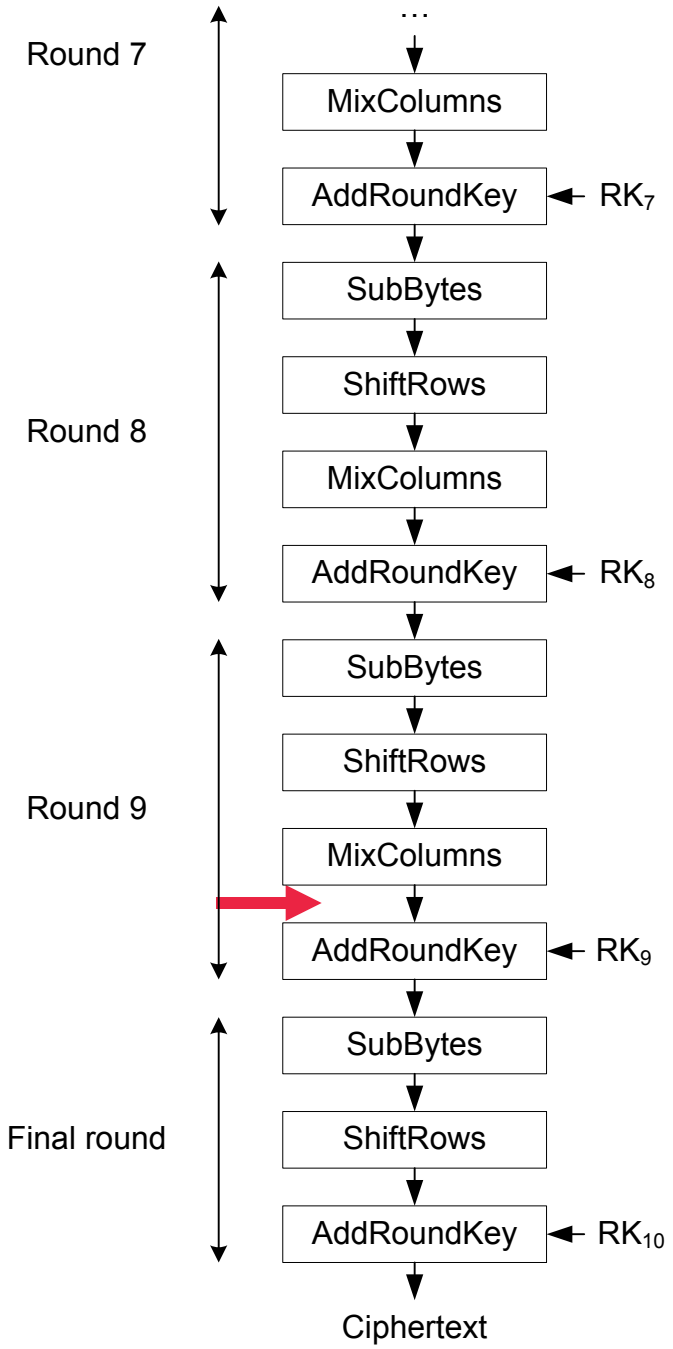


Fault Attacks on AES

Can an attacker learn something about the key by changing the input of AddRoundKey?

The same attacks work as for the Input of SubBytes

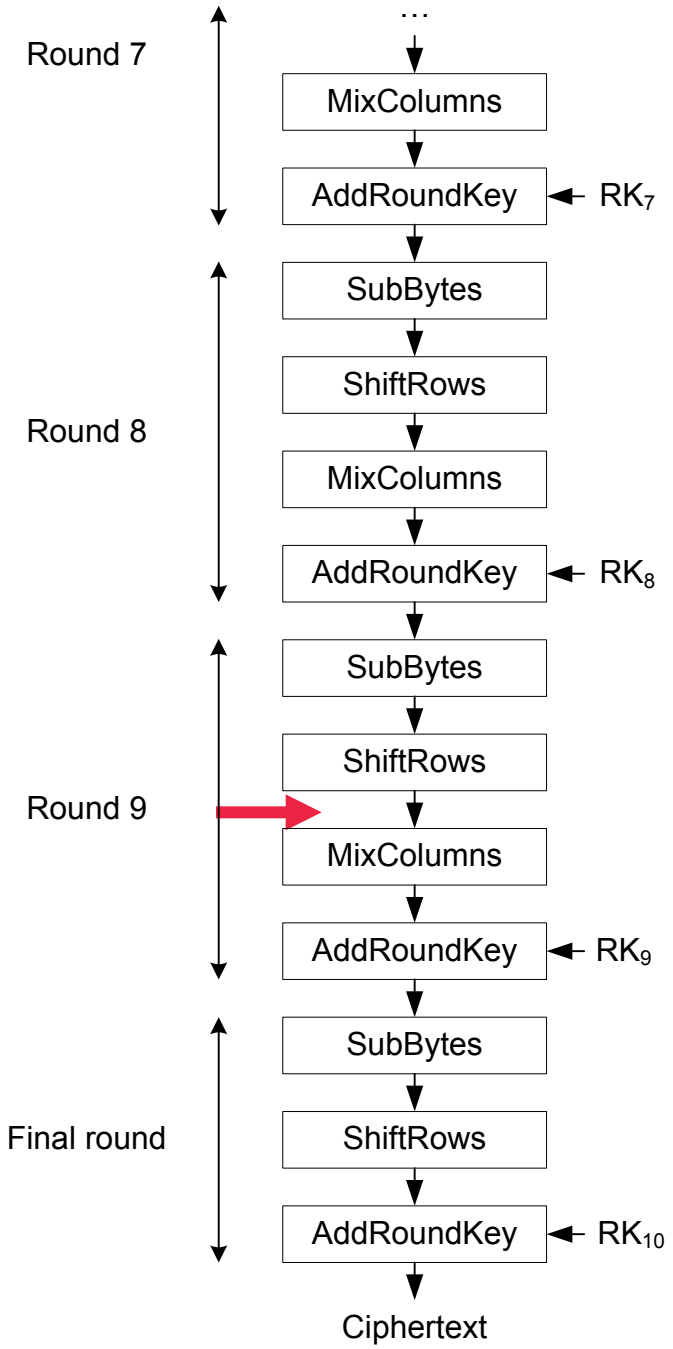
Important observation: Given an induced fault before this operation, it holds that the difference between the faulty encryption and the correct encryption is the same before and after the AddRoundKey operation



Fault Attacks on AES

Can an attacker learn something about the key by changing the input of MixColumns in Round 9?

This already leads to a very powerful attack, which is the basis of Piret's fault attack [PQ03]



Fault Propagation of the MixColumns Operation

- Mixcolumns is a linear operation that takes four bytes as inputs

- Assume the attacker induces a random error on the first input byte
  - We model this error as a bitwise difference Δ: the attack changes s_1 to $s_1 \oplus \Delta$
  - It holds:

    $\text{MixColumns}([s_1 \oplus \Delta, s_2, s_3, s_4]) =$

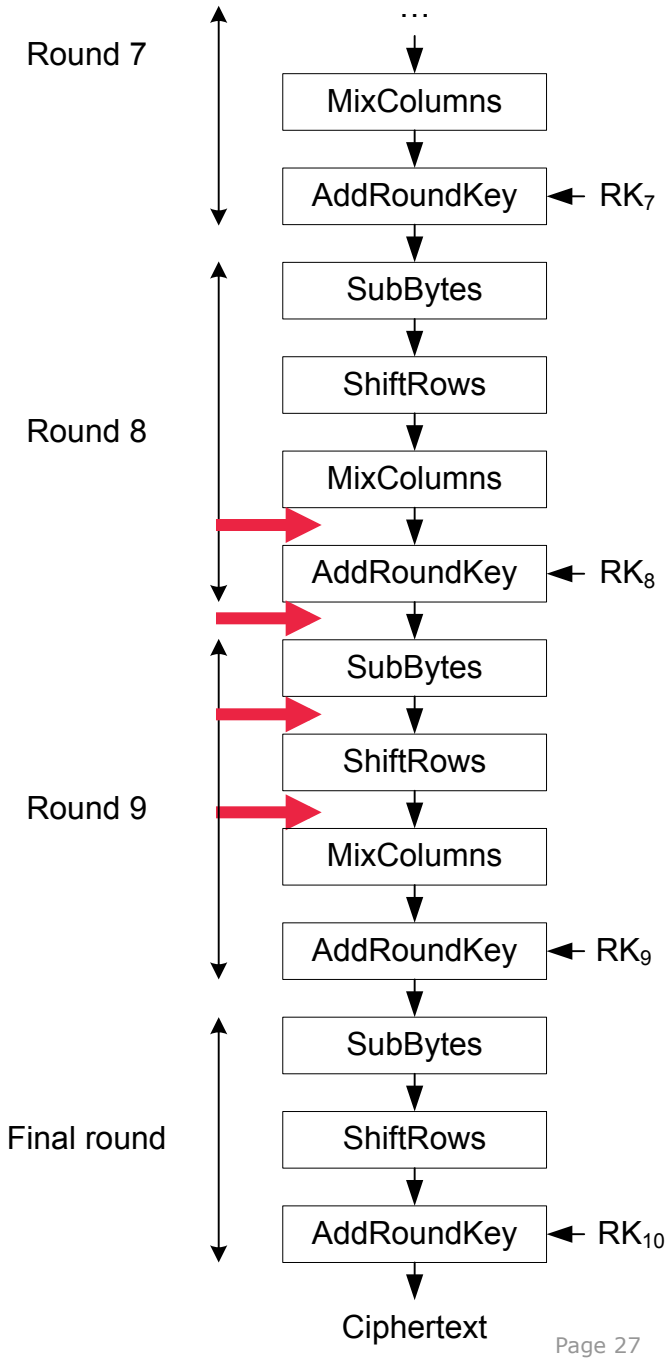
    $\text{MixColumns}([s_1, s_2, s_3, s_4]) \oplus \text{MixColumns}([\Delta, 0, 0, 0])$

- Observe: There are 255 possible values for Δ; hence, there are also 255 possible values for $\text{MixColumns}([\Delta, 0, 0, 0])$

Attack Setting

- Assume the attacker induces a **random unknown error** in one byte of the AES state somewhere in the computation between MixColumns in round 8 and MixColumns in round 9

- Observe:
  - It always holds that only one input byte of MixColumns in round 9 is affected by the attack
  - It always holds that four bytes of the ciphertext are affected by the attack



Preparation for the Attack

- The attacker generates a pair of ciphertext and faultytext by changing one byte of the state between MixColumns in round 8 and round 9
- 4 bytes are different between the faultytext and the ciphertext
- The attacker does not know which byte has been attacked, but based on the observed output difference, the attacker can narrow the possible bytes to 4 bytes of the state → these are the 4 bytes that enter MixColumns in round 9 as one column and then lead to the 4 different output bytes

- The attacker can now determine the 4*255 possible differences that MixColumns can produce for a fault in one input byte:
  - MixColumns([Δ, 0, 0, 0])
  - MixColumns([0, Δ, 0, 0])
  - MixColumns([0, 0, Δ, 0])
  - MixColumns([0, 0, 0, Δ])
- These are 1020 possible differences at the output of MixColumns in round 9

Principle of the of the Attack

- Obtain a pair of faultytext and ciphertext with a four byte difference that is caused by a byte error between Mixcolumns in round 8 and round 9
- Calculate the list of 1020 possible differences at the output of MixColumns in round 9
- Run through all 2^{32} combinations for the four key bytes in round 10 that correspond to the bytes of the ciphertext that are affected by the attack and for the pair of ciphertext and faultytext calculate the output of MixColumns in round 9 for each key combination
- Make a list of those key combinations that lead to one of the 1020 possible output differences of MixColumns
- If more than a unique combination remains, repeat the attack again with a new pair of ciphertext and faultytext and discard all combinations that are not in the list of both pairs

Efficient Implementation of the Attack

- For the first key byte of the affected 4 bytes of the ciphertext do
  - Calculate $Sbox^{-1}()$ for the ciphertext and the faultytext and the 256 possible keys
  - Check whether this key leads to a valid differential at byte position 1
  - Generate a list of potential key candidates by storing
    - the first key byte
    - the MixColumns output differential number (1 .. 1020)
- for each subsequent byte position do
  - Calculate $Sbox^{-1}()$ for the ciphertext and the faultytext and the 256 possible keys
  - Check whether this key leads to a valid differential at this byte position
  - update the list of key candidates: for all existing key candidates that are based on this differential add those bytes that are valid at his position and this differential as next byte

Effectiveness of the Attack

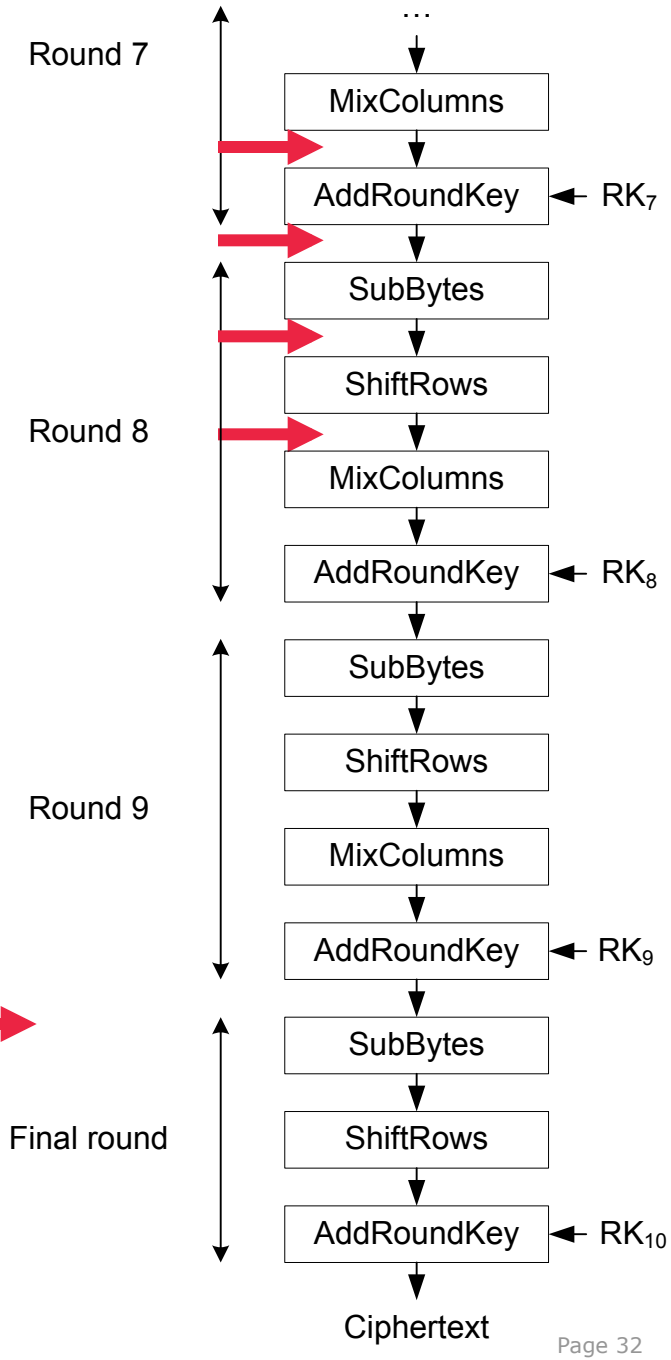
- This attack is already quite powerful
- Two pairs of ciphertext and faultytext are sufficient to reveal four key bytes
- However, this can be extended to a more powerful attack very easily …

Final Attack Setting

- Assume the attacker induces a random unknown error in one byte of the AES state somewhere in the computation between MixColumns in round 7 and MixColumns in round 8

- Observe:
  - It always holds that only one input byte of each MixColumns operation in round 9 is affected by the attack
  - The attack as described before can be applied for each of the four MixColumns operations in round 9

  → only two pairs of ciphertext and faultytext are necessary to reveal the entire key! See [PQ03]



Summary of [PQ03]

- The attack is generic in the sense that it works also for other linear transformations than AES MixColumns
- It is very effective, because it only requires two pairs of ciphertext and faultytext to reveal the entire key
- The fault model is already quite generic. The attack works, if one byte of the state between MixColumns of round 7 and round 8 is affected.
  - it does not matter which byte is attacked
  - it does not matter which bits of this byte are attacked
  - it is not necessary to know the fault Δ that has been induced

Questions

- How is it possible to induce a fault?
- Error detection codes.
- How many bits can you recover by applying a safe-error attack, per induced fault?
- How does the Bellcore attack work?
- What are two countermeasures to secure RSA?
- How does AES work?
- How does a fault attack on AES work, when you toggle the LSB before the final round S-box operation?
- How many plaintext / ciphertext pairs do you need to recover the full key?
- How many plaintext / ciphertext pairs do you need when you insert an error at the end of the 7^{th} round?

References

[97BDL] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. *On the Importance of Checking Cryptographic Protocols for Faults* (Extended Abstract). EUROCRYPT 1997.

[PQ03] Jean-Jacques Quisquater, and Gilles Piret: *A Differential Fault Attack Technique Against SPN Structures, with Application to the AES and KHAZAD*, Workshop on Cryptographic Hardware and Embedded Systems — CHES 2003, LNCS, Springer Verlag

[SMR09] Dhiman Saha and Debdeep Mukhopadhyay and Dipanwita RoyChowdhury: *A Diagonal Fault Attack on the Advanced Encryption Standard,* Cryptology ePrint Archive, Report 2009/581, available online at http://eprint.iacr.org/