Secure Implementation of Cryptographic Algorithms

Chapter 8
Fault Attacks

Wieland Fischer & Berndt Gammel
Infineon Technologies

(Slide set: Courtesy Stefan Mangard, TU Graz)

Content of Chapter 8

Motivation

- The idea of *fault attacks* is to determine the secret key of a device by inducing a fault during the computation of a cryptographic algorithm.
- Typically, the attacker obtains the output of an encryption for some plaintext and he obtains the output of a faulty encryption of some plaintext. A ciphertext and a corresponding faulty ciphertext are called a pair of ciphertext and faulty text.
- It depends on the type of fault and on the position of the fault within the algorithm, how much information about the key can be learned from a pair of ciphertext and faulty text.
- The strongest attacks on AES only require either
  - Two pairs of ciphertext/faulty text-pairs
  - One ciphertext/faulty text pair and the corresponding plaintext

  to reveal the entire key.

Chapter 8

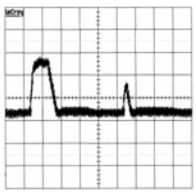
8.1. Basics of Fault Attacks

8.1. Basics of Fault Attacks
Reliability of Digital Circuits

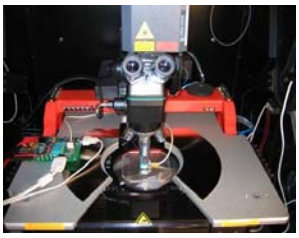
- Digital circuit require certain operating conditions to work properly
  - temperature range,
  - operating frequency,
  - supply voltage,
  - ...
- By changing the operating conditions an attacker can change the state of the circuit.

8.1. Basics of 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.



8.1. Basics of Fault Attacks
Effects of Fault Attacks

- Fault attacks can affect
  - The control flow: instructions are changed, skipped, …
  - Data integrity: data values changed.
- On hardware that includes no countermeasures against fault attacks, it is possible to observe practically all kinds of faults one can think of:
  - Arbitrary changes of the instruction pointer.
  - Arbitrary changes of data in registers or memory.
- Consequently, the following things may happen:
  - PIN and other checks are skipped.
  - Data pointers might be set from uncritical values to secret values (e.g. the key) and I/O functions may dump the secret values out of the chip.
  - …

8.1. Basics of Fault Attacks
Generic Fault Attack on Cryptographic Algorithms

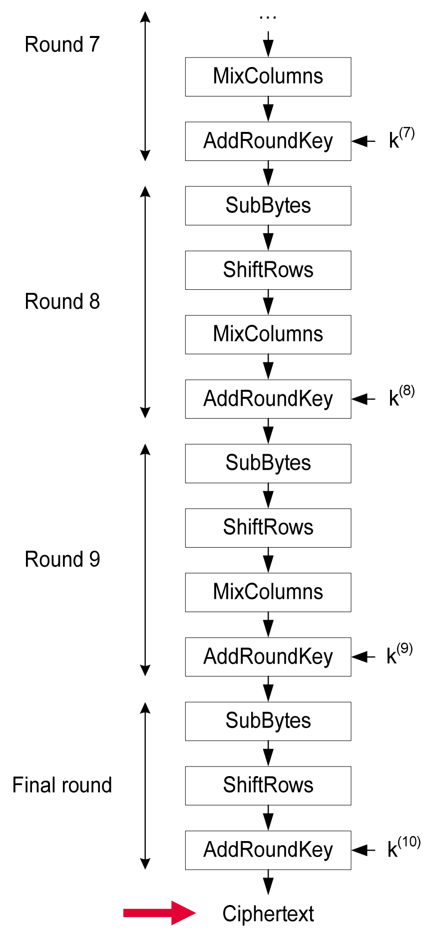
- Assume the attacker can change bits of data values selectively to a defined an known value, say, 0 for the example:
- Trivial attack in this fault model:
  - The attacker performs a reference encryption of some unknown plaintext and an unknown key.
  - The attacker repeats this encryption and attacks 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 simple attack method is also known as "safe error attack". 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.

8.2. Fault Attacks on AES

8.2. Fault Attacks on AES
Fault Attacks on AES

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

NO



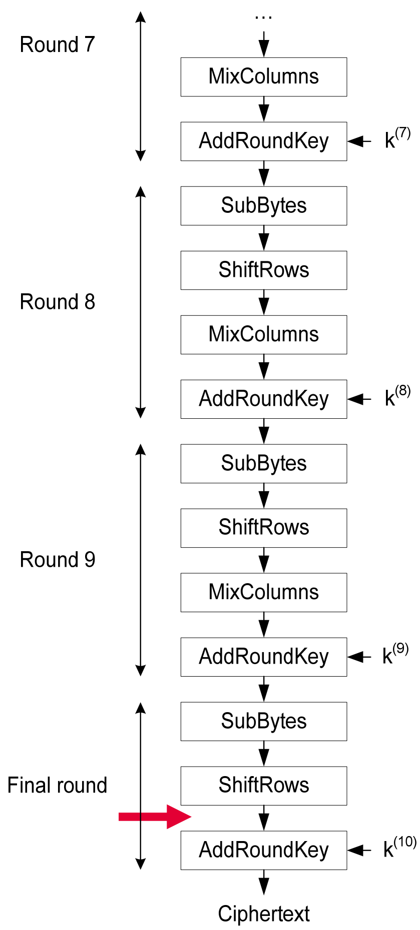
8.2. Fault Attacks on AES
Fault Attacks on AES

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

  It depends on the type of induced fault, also called the *fault model*:

- Fault model 1:
  The attacker can toggle one or multiple bits:

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



8.2. Fault Attacks on AES
Fault Attacks on AES

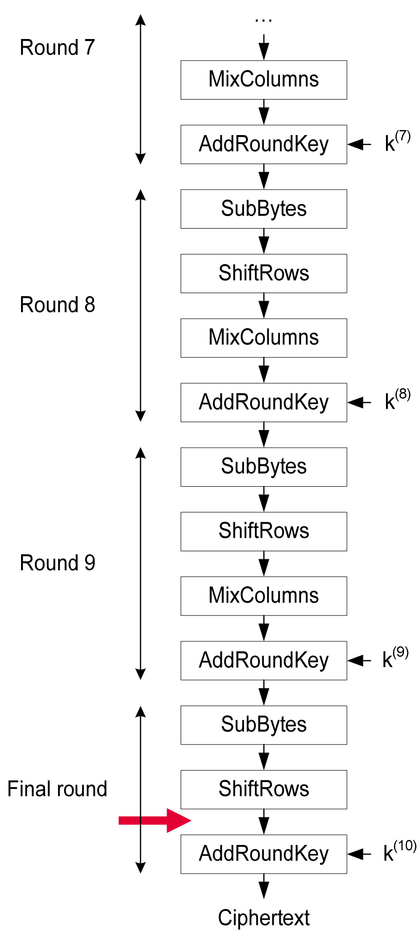
- Can an attacker learn something about the key by changing the input of AddRoundKey?

  It depends on the fault model:

- Fault model 2:
  The attacker can set bits to certain known values:

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

  $c_i = sr_i \oplus k_i^{(10)}$, e.g. with $sr_i := 0$, he gets $c_i = k_i^{(10)}$.



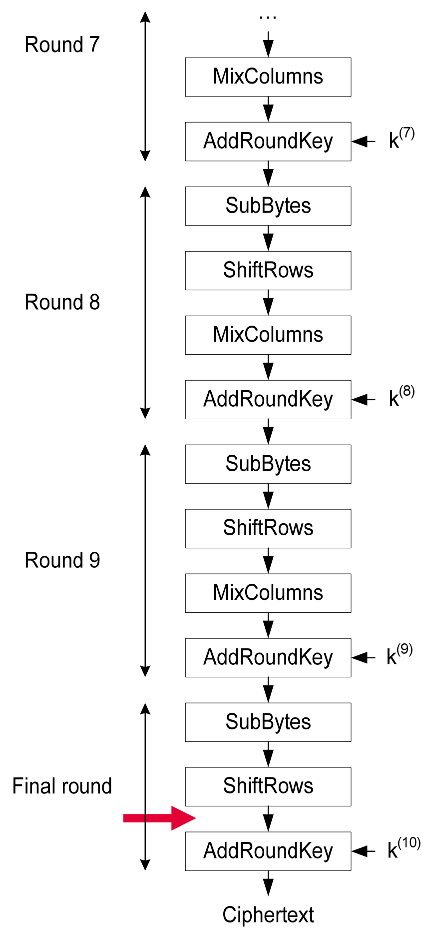
8.2. Fault Attacks on AES
Fault Attacks on AES

- 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:

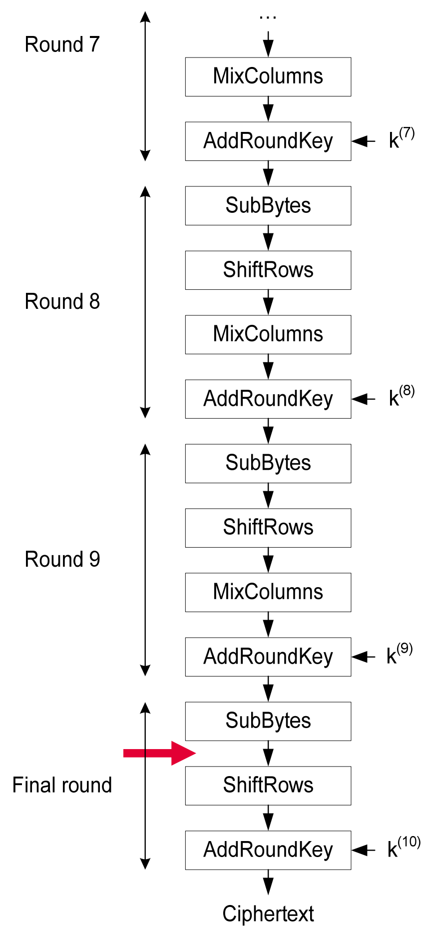
  The attacker again learns nothing as long as the induced fault is unknown.



8.2. Fault Attacks on AES
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.



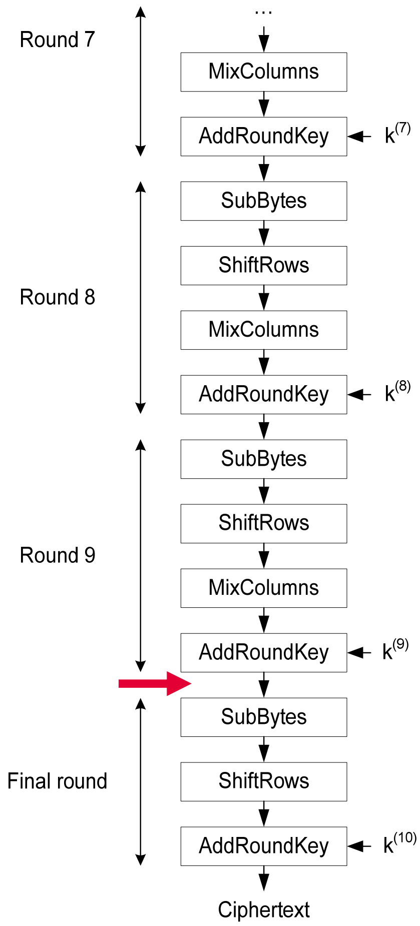
8.2. Fault Attacks on AES
Fault Attacks on AES

- 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, but does not know the value:

  Yes, an attack is possible!



8.2. Fault Attacks on AES
Fault Attack that toggles a bit of an S-box input

- Assume that the attacker is able to toggle the LSB of an S-box input.

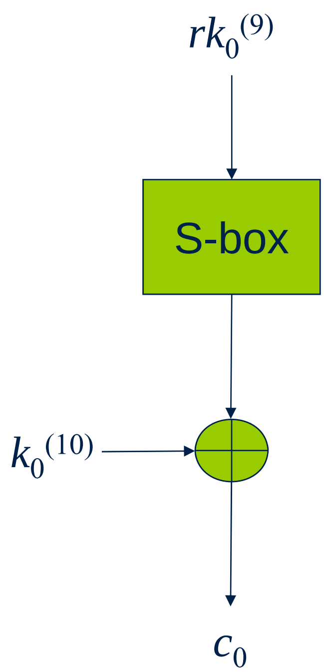
- Assume for each plaintext the attacker performs one correct encryption and one faulty encryption.

- For each pair of ciphertext byte c_i and faulty text byte f_i and a key hypothesis, the attacker can calculate the S-box input $rk_i^{(9)}$ based on the ciphertext and the faulty text, and he can check for which key hypothesis there is only a difference Δ_0 in the LSB of the Sbox input:

$$S^{-1}(c_i \oplus k_i^{(10)\mathrm{hyp}}) = rk_i^{(9)}$$

$$S^{-1}(f_i \oplus k_i^{(10)\mathrm{hyp}}) = rk'^{(9)}_i = rk_i^{(9)} \oplus \Delta_i$$

(we skip ShiftRows for notational simplicity)



8.2. Fault Attacks on AES
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 S-box input for the different keys and check whether there is only a difference in the LSB or not

	k=	00	01	02	03	04	05	06	07	08	09
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

8.2. Fault Attacks on AES
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 S-box input for the different keys and check whether there is only a difference in the LSB or not.

	k=	00	01	02	03	04	05	06	07	08	09
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 faulty text. Two pairs of ciphertext and faulty text are usually sufficient to determine one key byte.

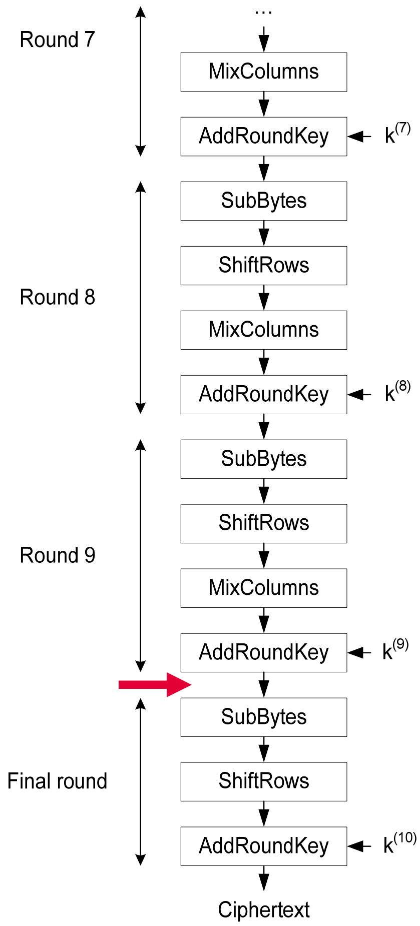
8.2. Fault Attacks on AES
Fault Attacks on AES

- 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 an unknown random (byte) fault:

  Attack is not possible because the random fault is again unknown.



8.2. Fault Attacks on AES
Fault Attacks on AES

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

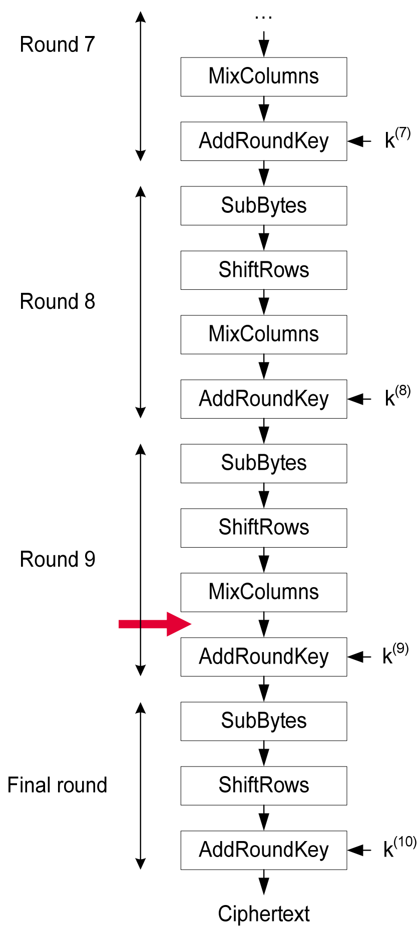
  The same attacks work as for the Input of SubBytes.

- 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.

  $$S^{-1}(c_i \oplus k_i^{(10)\text{hyp}}) \oplus k_i^{(9)} = mc_i^{(9)}$$

  $$S^{-1}(f_i \oplus k_i^{(10)\text{hyp}}) \oplus k_i^{(9)} = mc_i^{(9)} \oplus \Delta_i$$

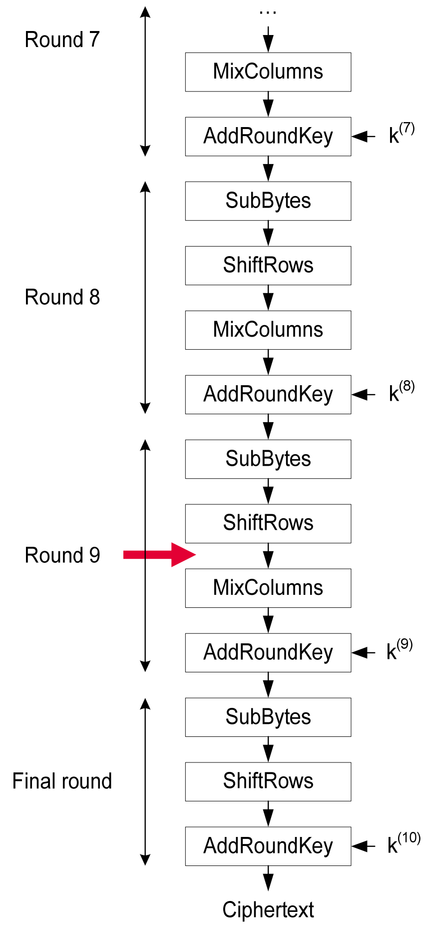
  (we skip ShiftRows for notational simplicity)



8.2. Fault Attacks on AES
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]



8.2. Fault Attacks on AES
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 $sr_0^{(9)}$.
  - We model this error as a bitwise difference Δ: the attack changes

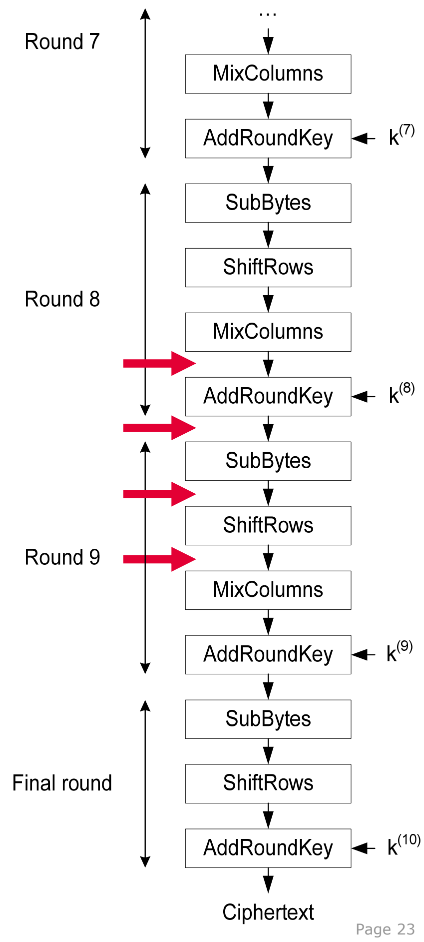
    $sr_0^{(9)}$ to $sr_0^{(9)} \oplus \Delta$
  - It holds:

$$\mathrm{MixColumns}([sr_0^{(9)} \oplus \Delta, sr_1^{(9)}, sr_2^{(9)}, sr_3^{(9)}]) =$$
$$= \mathrm{MixColumns}([sr_0^{(9)}, sr_1^{(9)}, sr_2^{(9)}, sr_3^{(9)}]) \oplus \mathrm{MixColumns}([\Delta, 0, 0, 0]).$$

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

8.2. Fault Attacks on AES
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 (i.e. of $sr^{(9)}$) in round 9 is affected by the attack.
  - It always holds that four bytes of the ciphertext are affected by the attack.



8.2. Fault Attacks on AES
Preparation for the Attack

- The attacker generates a pair of ciphertext and faulty text 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 $4 \cdot 255$ possible differences that MixColumns can produce for a fault in one input byte:

$$\text{MixColumns}([\Delta, 0, 0, 0])$$

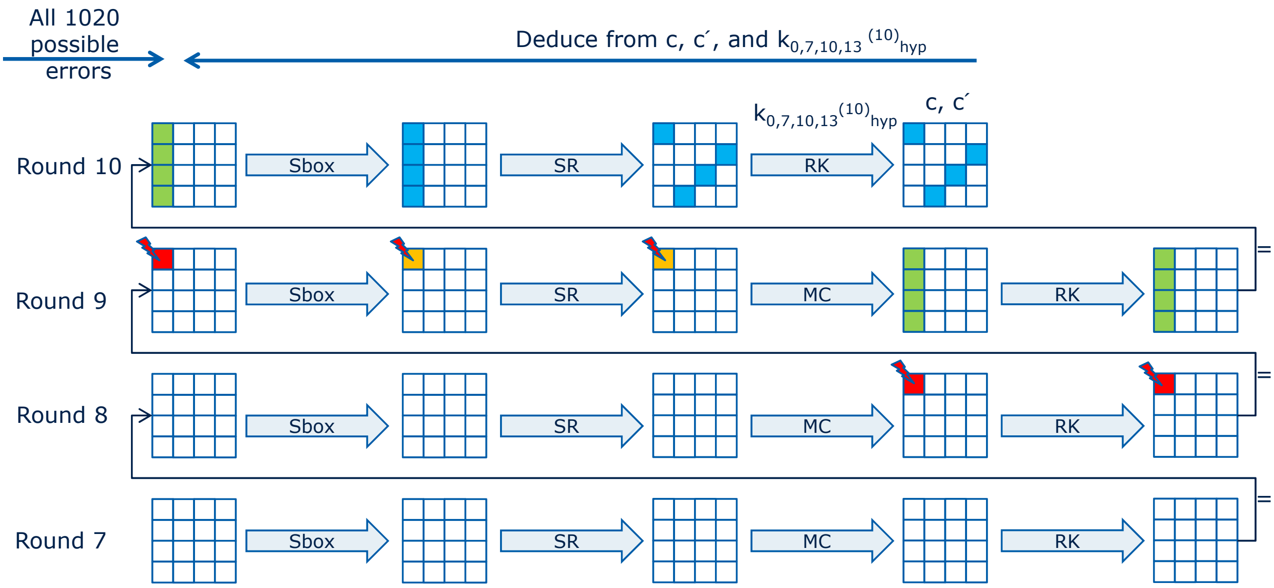
$$\text{MixColumns}([0, \Delta, 0, 0])$$

$$\text{MixColumns}([0, 0, \Delta, 0])$$

$$\text{MixColumns}([0, 0, 0, \Delta])$$

- These are 1020 possible differences at the output of MixColumns in round 9

8.2. Fault Attacks on AES
Principle of the of the Attack



8.2. Fault Attacks on AES
Principle of the of the Attack

- Obtain a pair of faulty text 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} values 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 value.
- Make a list of those key values that lead to one of the 1020 possible output differences of MixColumns.
- If more than a unique value remains, repeat the attack again with a new pair of ciphertext and faultytext and discard all values that are not in the list of both pairs.

8.2. Fault Attacks on AES
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 faulty text and the 256 possible keys.
  - (Check whether this key leads to a valid differential at this byte position.)
  - 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 faulty text 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.

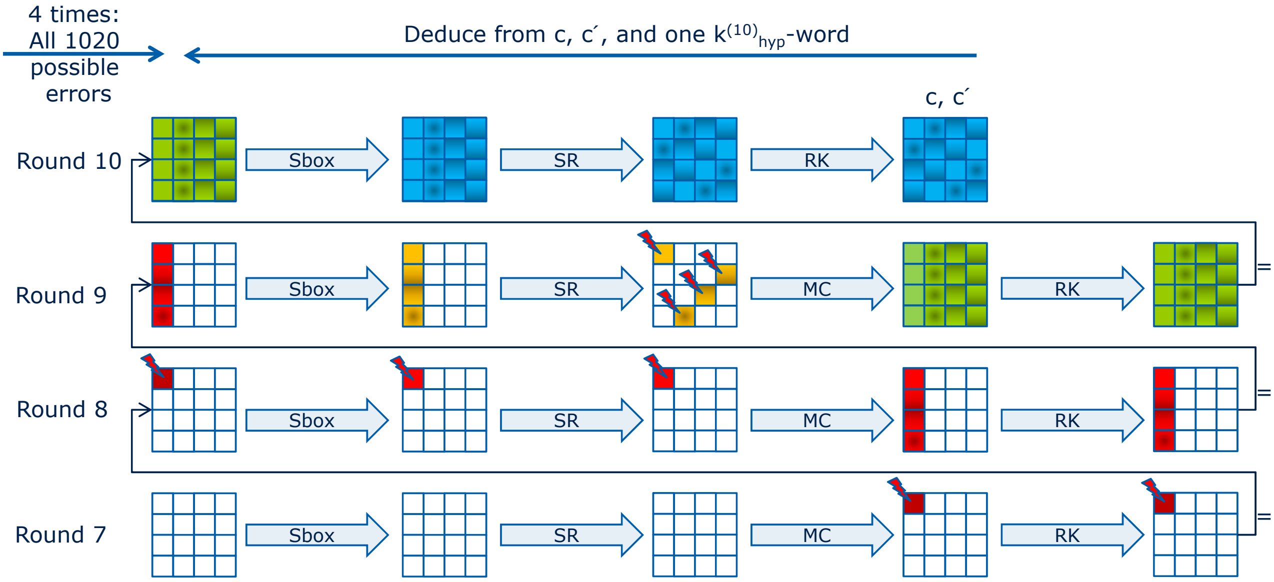
8.2. Fault Attacks on AES
Effectiveness of the Attack

- This attack is already quite powerful.

- Two pairs of ciphertext and faulty text are sufficient to reveal four key bytes.

- However, this can be extended to a more powerful attack very easily …

8.2. Fault Attacks on AES Parallelization

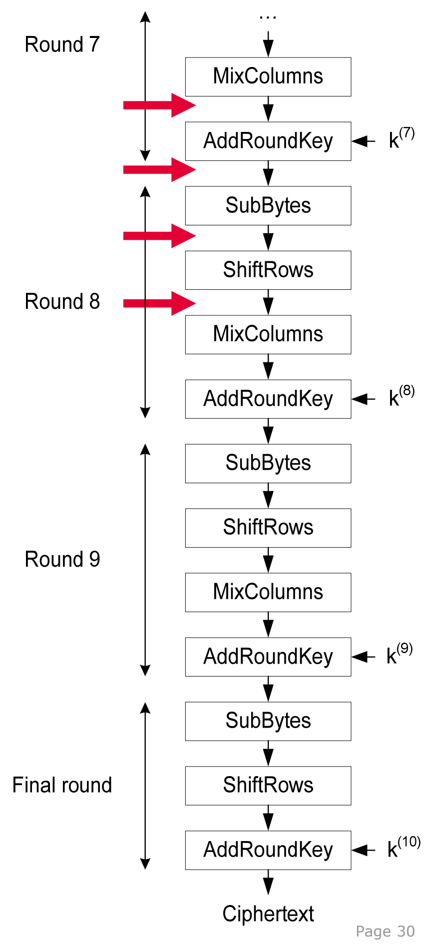


8.2. Fault Attacks on AES Final Attack Setting

- 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 faulty text are necessary to reveal the entire key! See [PQ03]



8.2. Fault Attacks on AES
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.

8.2. Fault Attacks on AES
Extension of the Attack of [PQ03]

- In 2009, Saha et al. [SMR09] extended the attack of Piret.

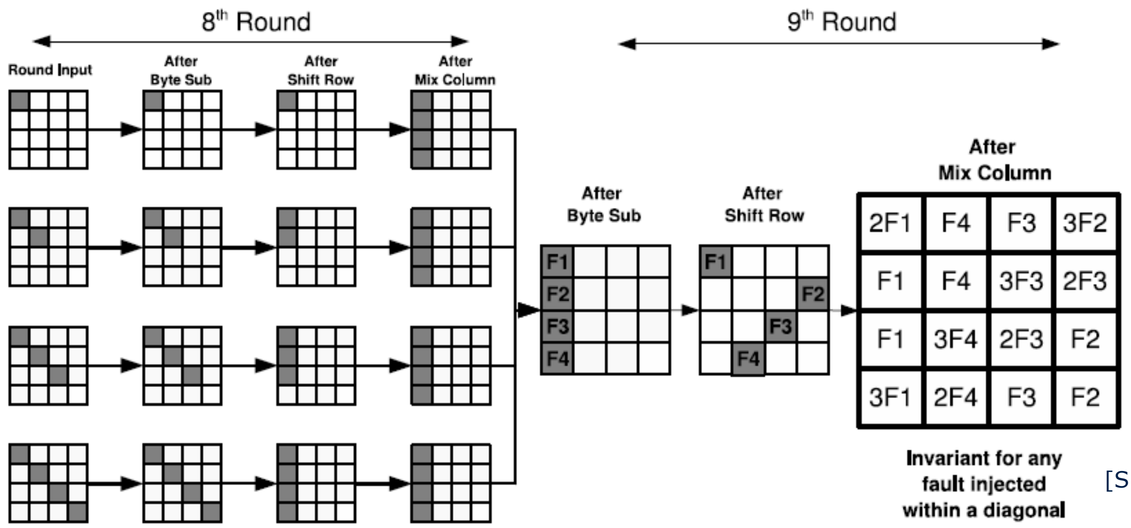
- In the extended attack, Saha et al. the attacker can modify up to **12 bytes** between MixColumns of round 7 and MixColumns of round 8 and the attack still reveals the entire AES key based on few pairs of ciphertext and faulty text.

  → Even if the attacker is very imprecise when inducing the fault, the key can still be revealed.

8.2. Fault Attacks on AES
The Goal

- Like in the original attack of Piret, the goal is to insert a one byte fault in each column of the AES state before MixColumns in round 9.

8.2. Fault Attacks on AES
The Attack of [SMR09]



[SMR09]

- The dark fields show bytes that are changed in the fault attack.
- $F_1, \ldots, F_4$ denote the XOR differences between the correct reference encryption and the faulty encryption.
- Remember that $2 \equiv x$ and $3 \equiv (x+1)$.

8.2. Fault Attacks on AES
Exploiting the Invariant

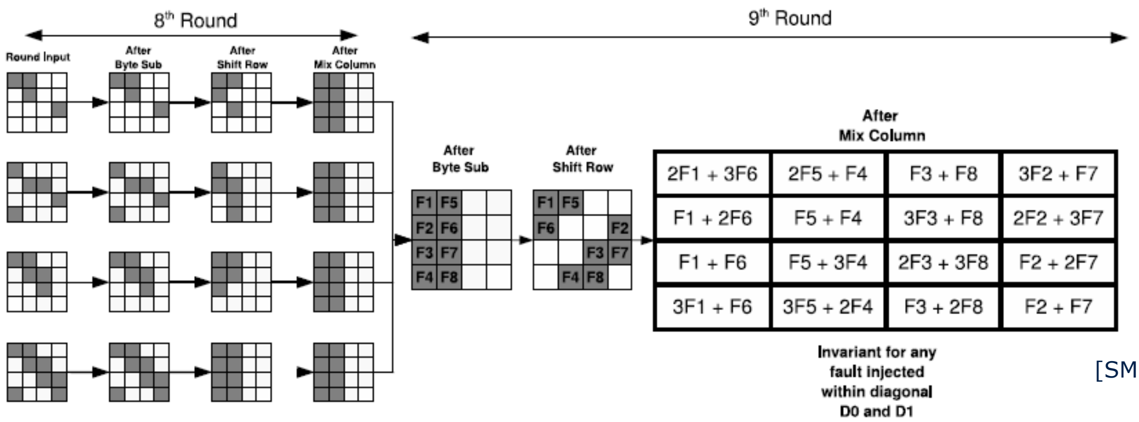
- Assume an attack on the first column and denote the elements as follows:
  - $a_0 = 2 \cdot F_1; a_1 = F_1; a_2 = F_1; a_3 = 3 \cdot F_1;$
- It then holds that
  - $a_0 = 2 \cdot a_1;\ a_1 = a_2;\ a_3 = 3 \cdot a_1$
- In order to exploit these equations:
  - The attacker calculates the MixColumns output errors of round 9 for all possible keys and for all pairs of ciphertext and faultytext.
  - The attacker then checks for which keys the equations above hold.
  - Note: as each equation requires only two byte elements the complexity of the algorithm is 2^{16} only.

After Mix Column

2F1	F4	F3	3F2
F1	F4	3F3	2F3
F1	3F4	2F3	F2
3F1	2F4	F3	F2

Invariant for any fault injected within a diagonal

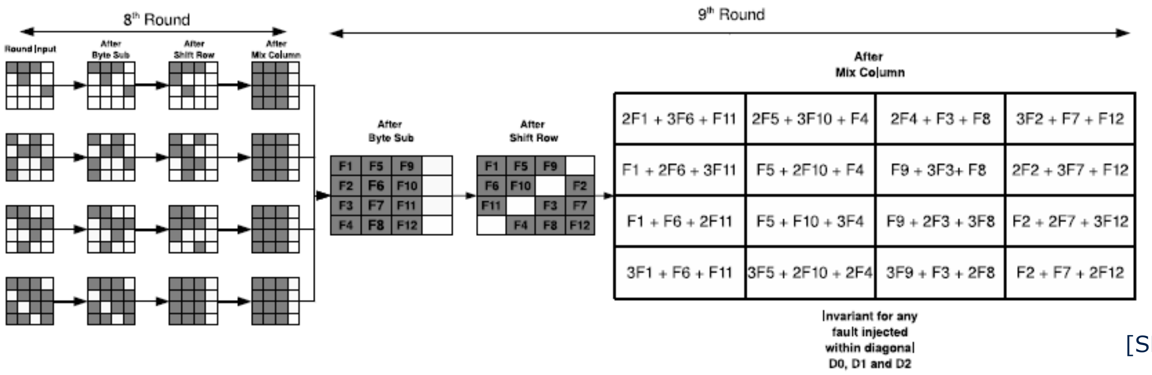
8.2. Fault Attacks on AES
The Attack of [SMR09]



[SMR09]

- In the first column, one has:
  - $a_0 = 2F_1 \oplus 3F_6;\ a_1 = F_1 \oplus 2F_6;\ a_2 = F_1 \oplus F_6;\ a_3 = 3F_1 \oplus F_6$.
- The attacker needs to check for all key candidates whether it holds that
  - $a_1 \oplus a_3 = a_0$ and $2a_1 \oplus 3a_3 = 7a_2.$ $[2 \equiv x,\ 3 \equiv x+1,\ 7 \equiv x^2+x+1]$
  - Since there are two relation with each 3 variables, the attacker has to make a hypothesis on 3 key bytes. So there are 2^{24} hypotheses for this column.

8.2. Fault Attacks on AES
The Attack of [SMR09]



[SMR09]

- In the first column one has:
  - $a_0 = 2F_1 \oplus 3F_6 \oplus F_{11}$; $a_1 = F_1 \oplus 2F_6 \oplus 3F_{11}$; $a_2 = F_1 \oplus F_6 + 2F_{11}$; $a_3 = 3F_1 + F_6 + F_{11}$.

- The attacker needs to check for all key candidates whether it holds that
  - $11a_0 \oplus 13a_1 = 9a_2 \oplus 11a_3$.
  - Since there is only one relation with 4 variables, the attacker has to make a hypothesis on all 4 key bytes. So there are 2^{32} hypotheses for this column.

8.2. Fault Attacks on AES
Summary of [SMR09]

- The other columns can be attacked in the same way as the first column.

- The complexity for all the attacks is never higher than 2^{32} and hence very practical.

- Even in case 12 bytes of the state are changed, according to [SMR09] 4 pairs of ciphertext and faulty text are sufficient to reveal the entire key.

8.2. Fault Attacks on AES
Countermeasures

- Countermeasures against fault attacks always require redundancy.
- This redundancy can either be implemented in software or hardware.
- Typical countermeasures in software:
  - Encrypt twice and compare the two results.
  - Save intermediate state after round x; after the calculation of the ciphertext, calculate backward again until round x and check whether the calculated result matches the saved intermediate state.
- Since every attack follows a – more or less restrictive – fault model, the designer may use the fact for a countermeasure that an attack is restricted to this fault model.

Chapter 8

8.3. Fault Attacks on RSA

8.3. Fault Attacks on RSA
The Bellcore Attack I

- Recall RSA with CRT: [97BDL]

$$
\begin{aligned}
S_p &:= m^{dp} \bmod p && [= m^{(d \bmod (p\text{-}1))} \bmod p]\\
S_q &:= m^{dq} \bmod q && [= m^{(d \bmod (q\text{-}1))} \bmod q]\\
S &:= S_q + q \cdot [(S_p\text{-}S_q) \cdot q_{\text{inv}} \bmod p]
\end{aligned}
$$

  - 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
    $x = \gcd(S'\text{-}S, N)$.
  - If $x > 1$ then $x = p$ or $x = q$.

- Explanation: Assume S_p' is disturbed. Then

$$
\begin{aligned}
S' - S &\equiv S_q + q \cdot [(S_p' - S_q) \cdot q_{\text{inv}} \bmod p] - S_q - q \cdot [(S_p\text{-}S_q) \cdot q_{\text{inv}} \bmod p]\\
&\equiv q \cdot [(S_p'\text{-}S_p) \cdot q_{\text{inv}} \bmod p] \bmod N\\
&= q \cdot y.
\end{aligned}
$$

  and therefore

$$\gcd(S' - S, N) = \gcd(q{\cdot}y, q{\cdot}p) = q.$$

8.3. Fault Attacks on RSA
The Bellcore Attack II

- 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.

- One obvious countermeasure:
  - 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…

8.3. Fault Attacks on RSA Countermeasure Against Bellcore I

Shamir's Countermeasure: [97Sh]

S_{pt} := $m^d \bmod p{\cdot}t$ [$= m^{(d \bmod (p\text{-}1)(t\text{-}1))} \bmod p{\cdot}t$]
S_{qt} := $m^d \bmod q{\cdot}t$ [$= m^{(d \bmod (q\text{-}1)(t\text{-}1))} \bmod q{\cdot}t$]
($S_{pt} \bmod t = S_{qt} \bmod t$)? → No: error
S_p := $S_{pt} \bmod p$
S_q := $S_{qt} \bmod q$
S := $S_q + q \cdot [(S_p\text{-}S_q) \cdot q_{\text{inv}} \bmod p]$

here, t is for example some 32 bit prime.

- Error detection probability: $\approx 1 - (1/t)$
- Advantage: Little overhead, since e.g. $p{\cdot}t$ is just 32 bits longer than p.
- Disadvantage:
  - needs d instead of (d_p, d_q), which usually is not provided by the protocol.
  - details: many security holes.

So, what to do, in order to work with (d_p, d_q) instead of d?

8.3. Fault Attacks on RSA Countermeasure Against Bellcore II

Shamir's Countermeasure, adapted: [01JPY]

Shamir's Countermeasure:
S_{pt} := $m^d \bmod pt$ [$= m^{(d \bmod (p\text{-}1)(t\text{-}1))} \bmod pt$]
S_{qt} := $m^d \bmod qt$ [$= m^{(d \bmod (q\text{-}1)(t\text{-}1))} \bmod qt$]
($S_{pt} \bmod t = S_{qt} \bmod t$)? → No: error
S_p := $S_{pt} \bmod p$
S_q := $S_{qt} \bmod q$
S := $S_q + q\, [(S_p\text{-}S_q) q_{\text{inv}} \bmod p]$

- S_{pt} := $m^{dp} \bmod p{\cdot}t$
  S_{qt} := $m^{dq} \bmod q{\cdot}t$
  s_{pt} := $m^{(dp \bmod (t\text{-}1))} \bmod t$
  s_{qt} := $m^{(dq \bmod (t\text{-}1))} \bmod t$
  ($S_{pt} \bmod t = s_{pt}$) and ($S_{qt} \bmod t = s_{qt}$)? → No: error
  S_p := $S_{pt} \bmod p$
  S_q := $S_{qt} \bmod q$
  S := $S_q + q \cdot [(S_p\text{-}S_q) \cdot q_{\text{inv}} \bmod p]$

- Advantage: works without d.
- Disadvantage: many holes if implemented and tested in reality:

Which security holes contains Shamir's countermeasure?

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore III

Shamir's Countermeasure, adapted:

$S_{pt} := m^{dp} \bmod pt$
$S_{qt} := m^{dq} \bmod qt$
$s_{pt} := m^{(dp \bmod (t-1))} \bmod t$
$s_{qt} := m^{(dq \bmod (t-1))} \bmod t$
$(S_{pt} \bmod t = s_{pt})$ and $(S_{qt} \bmod t = s_{qt})? \rightarrow$ No: error
$S_p := S_{pt} \bmod p$
$S_q := S_{qt} \bmod q$
$S := S_q + q[(S_p - S_q)q_{inv} \bmod p]$

In depth analysis: [02ABFHS]

1 $p' := p \cdot t$
2 $d_{pt} := d \bmod (p-1)\cdot(t-1)$
3 $q' := q \cdot t$
4 $d_{qt} := d \bmod (q-1)\cdot(t-1)$
5 $S_{pt} := m^{dpt} \bmod p'$
6 $S_{qt} := m^{dqt} \bmod q'$
7 $S_p := S_{pt} \bmod p$
8 $S_q := S_{qt} \bmod q$
9 $S := S_q + q \cdot [(S_p - S_q) \cdot q_{inv} \bmod p]$
10 $(S_{pt} \bmod t = S_{qt} \bmod t)? \rightarrow$ No: error

- Has the following attack points, generating fatal errors:
  - In 1: Disturb p during $p \cdot t$, such that $p' = x \cdot t$. Same with q in 3.
  - In 2: Disturb $(p-1)$. Same in 4.
  - In 7: Disturb reduction, hence the value of S_p. Same in 8.
  - In 9: Disturb computation of $[\ldots]$.
  - In 9: Disturb q_{inv}.

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore IV

New Proposal: [02ABFHS]

$p' := p \cdot t$
$d'_p := d_p + r_1 \cdot (p-1)$
$S'_p := m^{d'p} \bmod p'$
if not$(p' \bmod p = 0) \rightarrow$ error
if not$(d'_p \bmod (p-1) = d_p) \rightarrow$ error
$q' := q \cdot t$
$d'_q := d_q + r_2 \cdot (q-1)$
$S'_q := m^{d'q} \bmod p'$
if not$(q' \bmod q = 0) \rightarrow$ error
if not$(d'_q \bmod (q-1) = d_q) \rightarrow$ error
$S_p := S'_p \bmod p$
$S_q := S'_q \bmod q$
$S := S_q + q \cdot [(S_p - S_q) \cdot q_{inv} \bmod p]$

if not$(S - S'_q \bmod p = 0) \rightarrow$ error
if not$(S - S'_q \bmod q = 0) \rightarrow$ error
$S_{pt} := S'_p \bmod t$
$d_{pt} := d'_p \bmod t-1$
$S_{qt} := S'_q \bmod t$
$d_{qt} := d'_q \bmod t-1$
if not$(S_{qt}^{dpt} - S_{pt}^{dqt} \bmod t = 0) \rightarrow$ error

Shamir's Countermeasure, adapted:

$S_{pt} := m^{dp} \bmod pt$
$S_{qt} := m^{dq} \bmod qt$
$s_{pt} := m^{(dp \bmod (t-1))} \bmod t$
$s_{qt} := m^{(dq \bmod (t-1))} \bmod t$
$(S_{pt} \bmod t = s_{pt})$ and $(S_{qt} \bmod t = s_{qt})? \rightarrow$ No: error
$S_p := S_{pt} \bmod p$
$S_q := S_{qt} \bmod q$
$S := S_q + q[(S_p - S_q)q_{inv} \bmod p]$

- But: What happens, if m mod t=0? Then the last check might not be able to detect an error in an exponentiation!

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore

- Are there completely other ways of countermeasures?

- Yes: Self infecting computation.

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore

- Self infecting Computation as alternative:

- One conceptual problem of the former countermeasures:
  - Error detection bases on decisions which usually depend on 1 bit. An error introduced at the right place/time might overcome the countermeasure.

- Infective Computation introduced in [01YKLM]:
  - If one exponentiation contains an error, destroy the other exponentiation, in order to eliminate the useful information in the erroneous signature.
  - Does not contain any decisions any more but returns a signature in any case.

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore V

<u>Yen, Kim, Lee, Moon</u>: [01YKLM]

$\mathrm{CRT}(S_p, S_q) := S_q + q[(S_p - S_q)q_{\mathrm{inv}} \bmod p]$

$\delta := p - q$

Choose small random r and define:

$d_r := d - r$

$e_r := (d_r)^{-1} \bmod \varphi(N)$, such that e_r is small

1. $k_p := m \operatorname{div} p$, $k_q := m \operatorname{div} q$
   [→ $m = k_p \cdot p + (m \bmod p)$, → $m = k_q \cdot q + (m \bmod q)$]
2. $S_p := m^{dr} \bmod p$
   $m' := [(S_p^{er} \bmod p) + k_p \cdot \delta] \bmod q$
   [→ $m' = m \bmod q = [(m \bmod p) + k_p \cdot p] \bmod q$]
   $S_q := (m')^{dr} \bmod q$
3. $m'' := (S_q^{er} \bmod q) + k_q \cdot q$
   [→ $m'' = m = (m \bmod q) + k_q \cdot q$]
   $S := \mathrm{CRT}(S_p, S_q) \cdot (m'')^r \bmod N$
   [→ $S = (m^{d-r} \bmod N) \cdot (m^r \bmod N) = m^d \bmod N$]

- Unfortunately, this measure can be broken. ([Blömer, Seifert, May] & [Yen, Kim, Moon, FDTC 2004] by disturbing k_q.)

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore VI

<u>Blömer, Otto, Seifert</u>: [03BOS]

Choose more or less small random t_1, t_2 and compute:

$p' := t_1 \cdot p$ and $q' := t_2 \cdot q$

$d_1 := d \bmod \varphi(p')$ and $d_2 := d \bmod \varphi(q')$

$e_1 := (d_1)^{-1} \bmod \varphi(t_1)$ and $e_2 := (d_2)^{-1} \bmod \varphi(t_2)$

1. $S_{p'} := m^{d1} \bmod p'$ and $S_{q'} := m^{d2} \bmod q'$
   $S' := \mathrm{CRT}(S_{p'}, S_{q'})$
2. $c_1 := (m - S'^{e1} + 1) \bmod t_1$ and $c_2 := (m - S'^{e2} + 1) \bmod t_2$
   [$c_1 = c_2 = 1$ if error free]
3. $S := (S')^{c1 \cdot c2} \bmod N$

- This method was also broken [04Wa]:

8.3. Fault Attacks on RSA Countermeasure Against Bellcore

Wagner, "Cryptanalysis of a provably secure CRT-RSA-Algorithm":

- If one attacks the first exponentiation (in this case c_2=1), and if one can guess c_1, then one can recover q by computing
  gcd(S^e - m^{c1} , N),
  since in this case:

  $S' \neq m^d \bmod p$ and $S' = m^d \bmod q$

  $S \neq m^{d \cdot c1} \bmod p$ and $S = m^{d \cdot c1} \bmod q$

  $S^e \neq m^{c1} \bmod p$ and $S^e = m^{c1} \bmod q$.

- Realization, by changing the lowest byte of the temporary message m for only the first exponentiation. Then this message might be written as
  $m' = m\text{-}b$, with $-256 < b < 256$
  hence
  $$c_1 = m - S'^{e1} + 1 \bmod t_1 = m - (m\text{-}b) + 1 \bmod t_1 = b + 1.$$

8.3. Fault Attacks on RSA Countermeasure Against Bellcore VII

Joye, Ciet: [05CJ]

Choose random r_1,r_2 coprime.

$p' := r_1 p$ and $q' := r_2 q$

$i' := (q')^{-1} \bmod p'$

$N := pq$

1. $S_{p'} := m^{dp} \bmod (p')$ and $S_{q'} := m^{dq} \bmod (q')$

   $s_1 := m^{dp \bmod \varphi(r1)} \bmod r_1$ and $s_2 := m^{dq \bmod \varphi(r2)} \bmod r_2$

   $S' := S_{q'} + [(S_{p'} - S_{q'})\, i' \bmod p']\, q'$

2. $c_1 := (S - s_1 + 1) \bmod r_1$

   $c_2 := (S - s_2 + 1) \bmod r_2$

   $g := [(r_3 c_1 + (2^l - r_3) c_2) \text{ div } 2^l]$,
   for some random $0 < r_3 < 2^l$

3. $S := S'^g \bmod N$

- Also this method was broken by [08BCG] :
  This is done like above by guessing the value g.

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore VIII

Giraud: [05Gi]

Uses Montgomery ladder technique to secure only the two partial exponentiations.

Recall Montgomery Ladder for the computation of $m^d \bmod N$:
with $d = (d_{n-1}\, d_{n-2} \ldots d_1\, d_0)_2$. Write $D_i := (d_{n-1}\, d_{n-2} \ldots d_{n-i})_2$.

```
(A0,A1) := (1,m);
for i:=n-1 to 0 by -1 do
      (A0, A1) := (A0 · Adi, Adi · A1);
      // now (A0, A1) = (m^(Dn-i), m^(Dn-i+1))
end;
return A0;
```

Countermeasure:
check whether $A_1 = A_0 \cdot m \bmod N$
if not → error.

8.3. Fault Attacks on RSA
Countermeasure Against Bellcore IX

Kim, Ha, Moon, Yen, Kim: (HPCC 2005)

Uses a square-and-always-multiply technique to secure only the two partial exponentiations.

Square-and-always-multiply for the computation of $m^d \bmod N$:
with $d = (d_{n-1}\, d_{n-2} \ldots d_1\, d_0)_2$. Write $D_i := (d_{n-1}\, d_{n-2} \ldots d_{n-i})_2$.

```
choose random r and compute (r^-1 mod N)
A := m · r;
for i:=n-1 to 0 by -1 do
      A := A^2 mod N        // now A = m^(2Di-1) · r^2
      if di=0 then
            A := A · (r^-1 mod N) mod N
      else if di=1 then
            A := A · (m · r^-1 mod N) mod N
            // now A = m^(Di) · r
      end;
end;
return A · (r^-1 mod N);
```

8.3. Fault Attacks on RSA (Incomplete) History of Countermeasures

Author	Where	When	Broken?
Shamir	Eurocrypt	1997	Aumüller, CHES 2002
Yen et al	ICISC	2001	Yen, FDTC 2004
Aumüller et al	CHES	2002	Yen, ICISC 2002
Blömer et al	CCS	2003	Wagner, CCS 2004
Kim et al	HPCC	2005	-
Joye, Ciet	FDTC	2005	Berzati, FDTC 2008
Giraud	FDTC	2005	-
Fumaroli	FDTC	2006	Kim, FDTC 2007
Vigilant	FDTC	2008	
…			

8.3. Fault Attacks on RSA Conclusion

- Fault Attacks on RSA
  - … are devastating if RSA is implemented with CRT. Only one faulty output may be enough to recover the secret.
  - … on other implementations are also possible, but usually are not as efficient.
- Detection of fault attacks always use some kind of redundancy by
  - … computing multiple times or verifying the result.
  - … inserting redundancy like computing modulo bigger/enriched modules.
  - … using algorithm intrinsic redundancy (Montgomery ladder).
- Detection can be done by
  - … checking the redundancy and returning error/alarm-messages.
  - … using redundancy to destroy further the erroneous result in order to obfuscate the error-information.

References

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

[97Sh] Adi Shamir. *Method and Apparatus for protecting public key schemes from timing and fault attacks*. US Patent Office, November 1999. US Patent Number 5 991 415, also presented at EUROCRYPT 1997.

[99KJJ] Paul C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. CRYPTO 1999.

[00Sch] Werner Schindler. *A Timing Attack against RSA with the Chinese Remainder Theorem*. CHES 2000.

[01JPY] Marc Joye, Pascal Pailler, and Sung-Ming Yen. *Secure evaluation of modular functions*. In Hwang R.J and C.K. Wu, editors, International Workshop on Cryptology and Network Security, pages 227-229, 2001.

References

[01WT] Colin D. Walter and Susan Thompson. *Distinguishing Exponent Digits by Observing Modular Subtractions*. In David Naccache, editor, CT-RSA, volume 2020 of Lecture Notes in Computer Science, pages 192-207. Springer, 2001.

[01YKLM] Sung-Ming Yen, Seungjoo Kim, Seongan Lim, and Sang-Jae Moon. *RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis*. In Kwangjo Kim, editor, Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings,volume 2288 of Lecture Notes in Computer Science, pages 397-413. Springer-Verlag, 2001.

[02ABFHS] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. *Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures*. In CHES 2002.

[02BLW] Bert den Boer, Kerstin Lemke, and Guntram Wicke. *A DPA Attack against the Modular Reduction within a CRT Implementation of RSA*. CHES 2002.

References

[02No] Roman Novak. *SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation*. PKC2002.

[02YMJ] Sung-Ming Yen, Sang-Jae Moon, and JaeCheol Ha. *Hardware Fault Attack on RSA with CRT Revisited*. In Pil Joong Lee and Chae Hoon Lim, editors, ICISC, volume 2587 of Lecture Notes in Computer Science, pages 374-388. Springer, 2002.

[03BOS] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. *A new CRT-RSA algorithm secure against bellcore attacks*. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washingtion, DC, USA, October 27-30, 2003, pages 311-320. ACM, 2003.

[04Wa] David Wagner. *Cryptanalysis of a provably secure CRT-RSA algorithm*. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washingtion, DC, USA, October 25-29, 2004, pages 92-97. ACM, 2004.

References

[05CJ] Mathieu Ciet and Marc Joye. *Practical Fault Countermeasures for Chinese Remaindering Based RSA*. FDTC 2005.

[05Gi] Christophe Giraud. *Fault Resistant RSA Implementation*. FDTC 2005

[05KHMY] ChangKyun Kim, JaeCheol Ha, Sang-Jae Moon, Sung-Ming Yen, and Sung-Hyun Kim. *A CRT-Based RSA Countermeasure Against Physical Cryptanalysis*. In Laurence Tianruo Yang, Omer F. Rana, Beniamino Di Martino, and Jack Dongarra, editors, HPCC, volume 3726 of Lecture Notes in Computer Science, pages 549-554. Springer, 2005.

[06FV] Guillaume Fumaroli and David Vigilant. *Blinded Fault Resistant Exponentiation*. FDTC 2006.

[08BCG] Alexandre Berzati, Cecile Canovas, and Louis Goubin. *In(security) Against Fault Injection Attacks for CRT-RSA Implementations*. FDTC 2008.

[08Vi] David Vigilant. *RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks*. CHES 2008.

References (AES)

[BS97] Eli Biham and Adi Shamir: *Differential Fault Analysis of Secret Key Cryptosystems*, Advances in Cryptology – CRYPTO 1997, LNCS, Springer Verlag.

[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.

[R09] Matthieu Rivain: *Differential Fault Analysis on DES Middle Rounds*, Workshop on Cryptographic Hardware and Embedded Systems — CHES 2009, 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/.