Fault Analysis of DPA-Resistant Algorithms

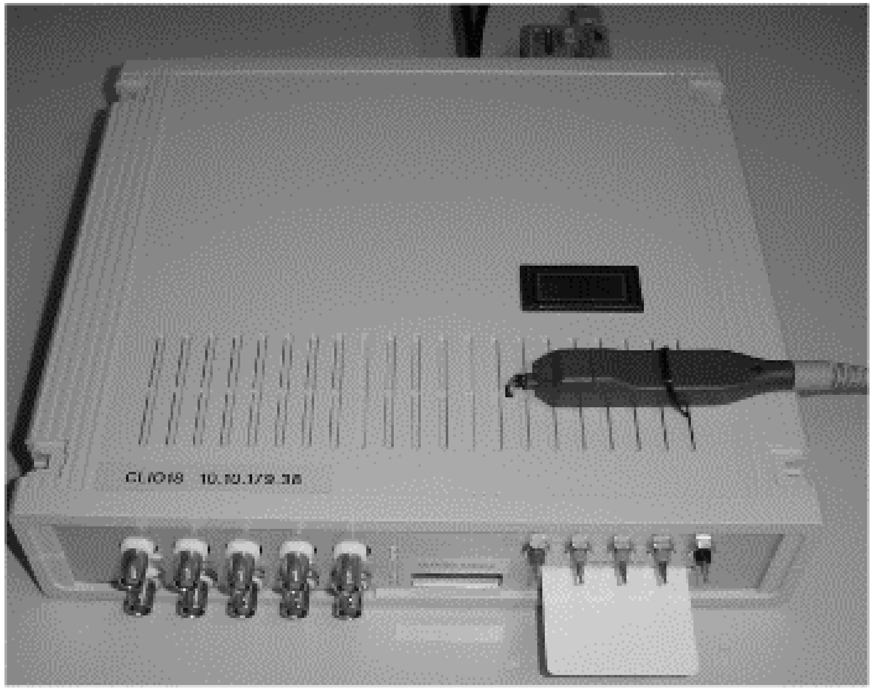
Frederic Amiel,
Christophe Clavier
& Michael Tunstall

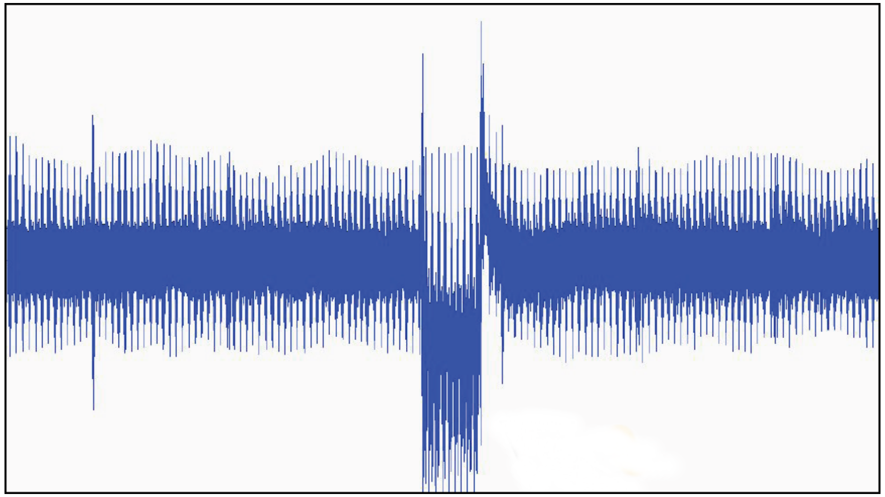
Collision Fault Analysis

- In an attack was proposed against a hardware AES [Blomer and Seifert, 2003].
- If one bit of the first XOR is set to zero (using a fault) and the ciphertext compared a normal execution.
  - If the ciphertexts are the same then the bit was zero.
  - If the ciphertexts are different then the bit was one.
- Can find the Key in 128 faults … but requires a high degree of precision.

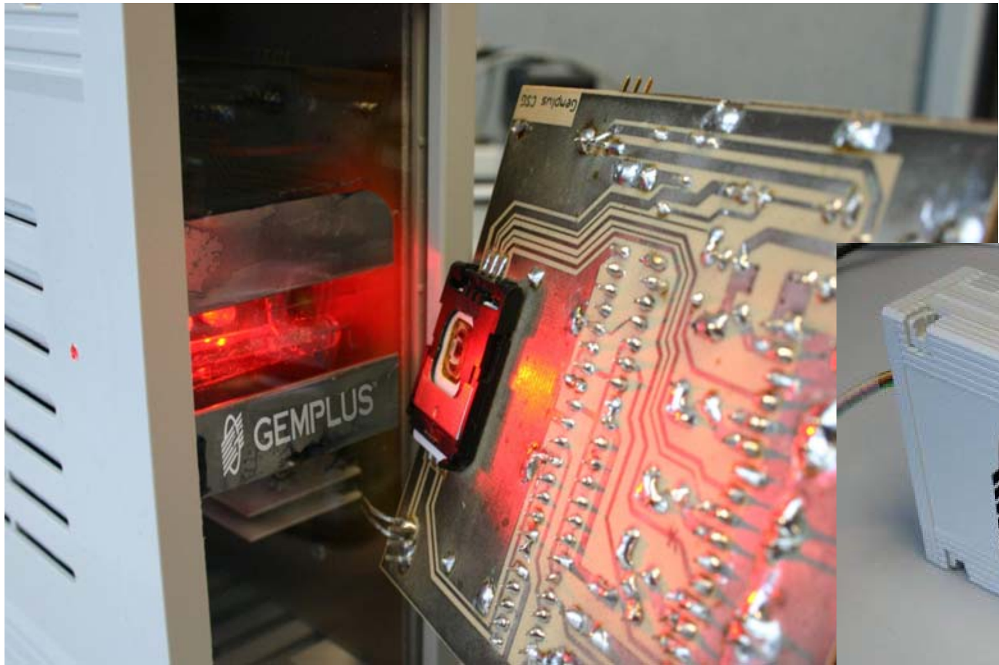
- We attempted a bytewise version of this on an 8-bit microcontroller.
  - Setting a byte to zero and searching for a message block that would produce this ciphertext.

Fault Injection Equipment: CLIO Glitch Injector

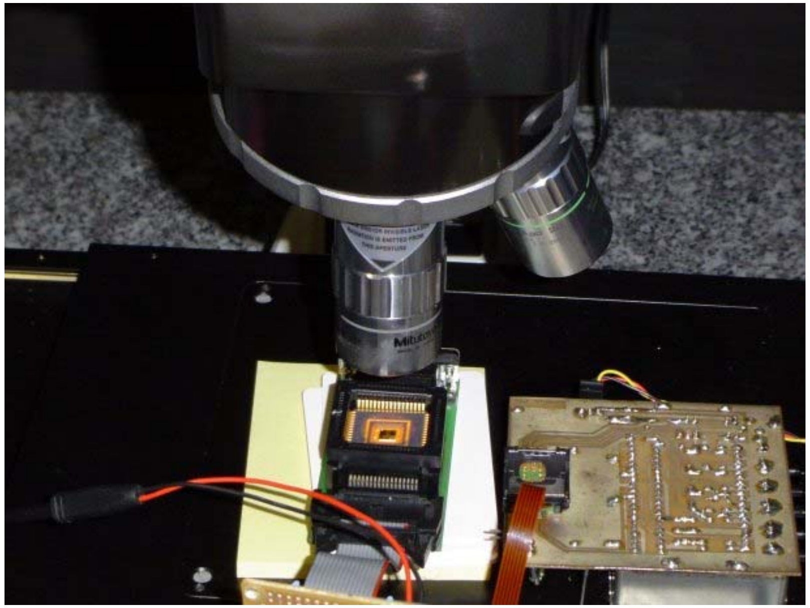


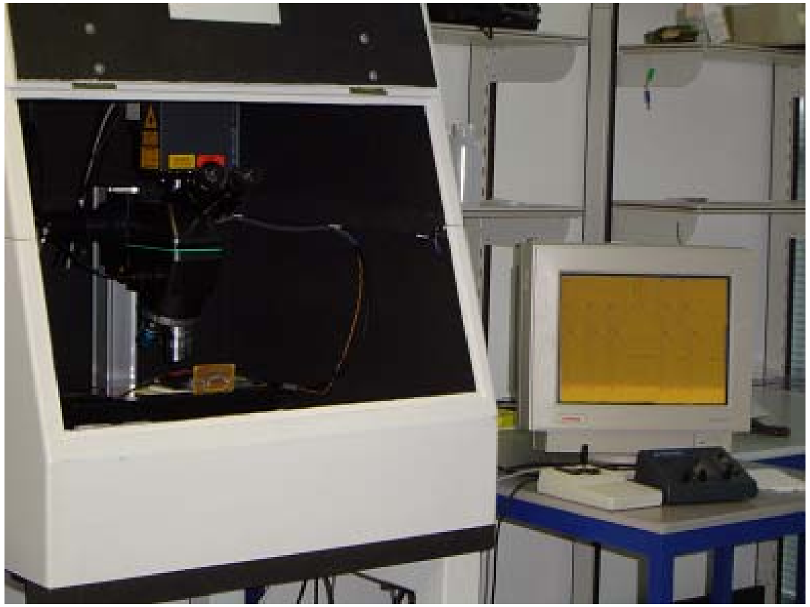


Fault Injection Equipment: Flash



Fault Injection Equipment: Laser





Collision Fault Analysis

- All of the attack methods were unsuccessful in producing the desired effect.

- Previously, have been able to use faults to break open loops.
- Allows the key loading loop to be broken to gradually reduce the key [Biham and Shamir, 1997]

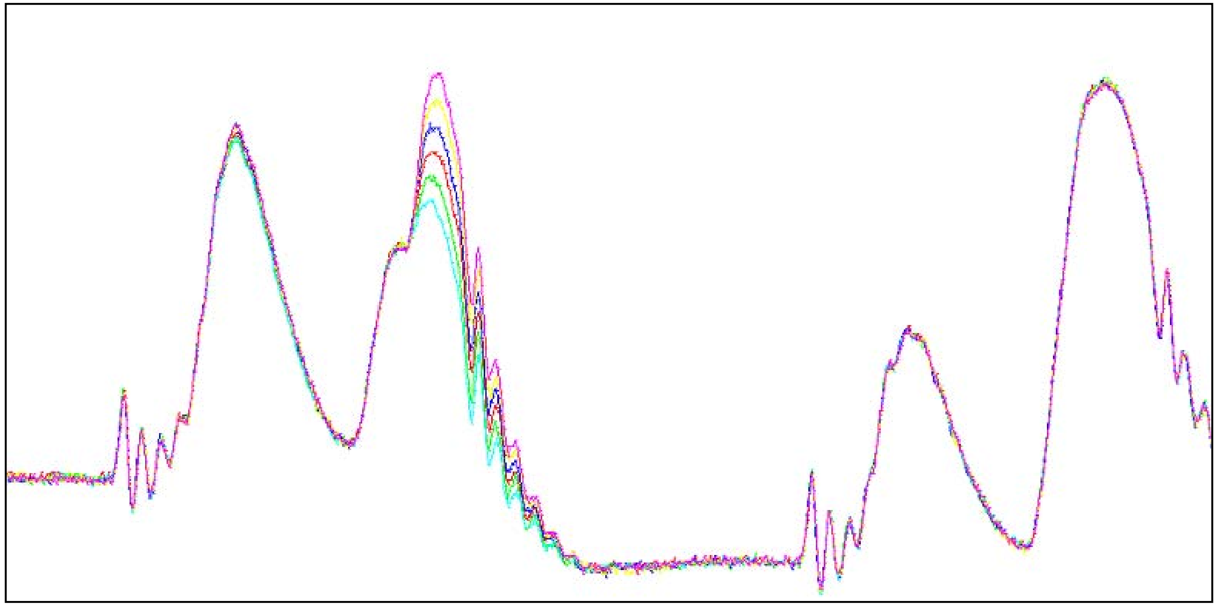
Input	AES Key																	Output
$M \rightarrow$	$K_0 =$	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	$\rightarrow C_0$
$M \rightarrow$	$K_1 =$	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	00	$\rightarrow C_1$
$M \rightarrow$	$K_2 =$	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	00	00	$\rightarrow C_2$
$M \rightarrow$	$K_3 =$	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	00	00	00	$\rightarrow C_3$
$\vdots$	$\vdots$																	$\vdots$
$M \rightarrow$	$K_{14} =$	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	$\rightarrow C_{14}$
$M \rightarrow$	$K_{15} =$	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	$\rightarrow C_{15}$

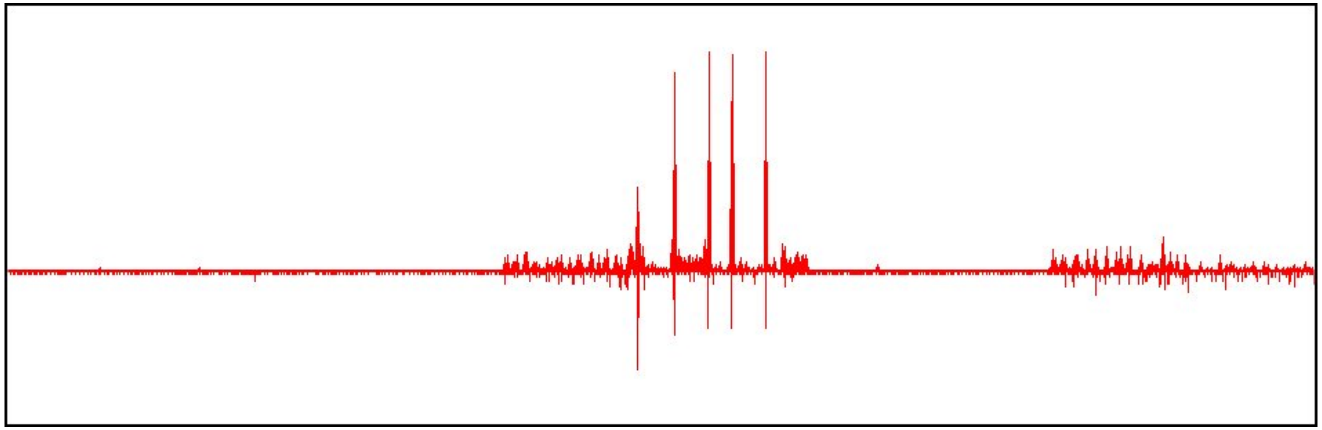
Collision Fault Analysis

- Were able to produce this with a glitch.
- Scanning the whole loop (127 faulty ciphertexts), we found the keys:

```
00000000000000000000000000000000
FE000000000000000000000000000000
FEDC0000000000000000000000000000
FEDCBA00000000000000000000000000
FEDCBA98000000000000000000000000
                ...
FEDCBA98765432100123456789000000
FEDCBA98765432100123456789ABCDEF
FEDCBA98765432100123456789ABCD00
FEDCBA98765432100123456789ABCDCD
FEDCBA98765432100123456789AB0000
FEDCBA98765432100123456789AB00EF
FEDCBA98765432100123456789AB00AB
FEDCBA98765432100123456789ABEF00
FEDCBA98765432100123456789AB5300
```

DPA-Resistant Algorithms





- Statistical analysis of small differences in power consumption to predict intermediate states.

DPA-Resistant Algorithms

- All data is masked by XORing each byte with a random *R.*
- S-boxes therefore need to be constructed in RAM.

Algorithm 4: Randomising S-Box Values

Input: $S = (s_0, s_1, s_2, \ldots, s_n)_x$ containing the s-box, $\mathbf{R}$ a random $\in [0, n]$, and r a random $\in [0, x)$.

Output: $RS = (rs_0, rs_1, rs_2, \ldots, rs_n)_x$ containing the randomised s-box.

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

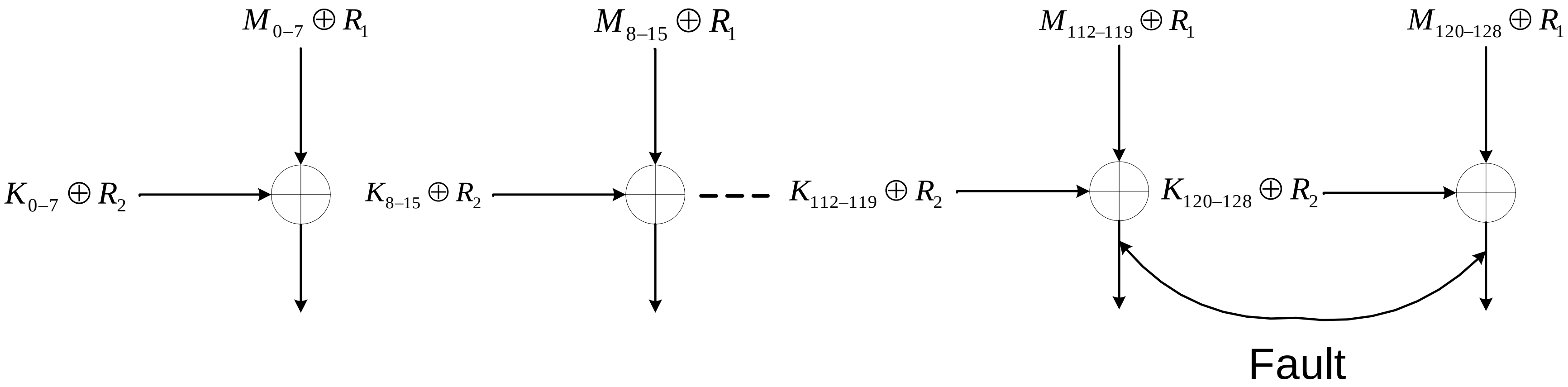
  $rs_i \leftarrow s_{(i \oplus \mathbf{R})} \oplus r$

end

return RS

CFA on a DPA-Resistant Algorithm

- Reconsider the first XOR with the key.
- Two bytes are changed, although each byte of the message and key is masked with a Random:



- Fault on two bytes to a fixed values (to zero for example).
- Gives (in memory):

$M \oplus K \oplus R$	$M \oplus K \oplus R$	$\ldots$	$M \oplus K \oplus R$	$M \oplus K \oplus R$	0	0

- In algorithm:

$M \oplus K$	$M \oplus K$	$\ldots$	$M \oplus K$	$M \oplus K$	R	R

CFA on a DPA-Resistant Algorithm

- In our case with one fault we can break the for loop, two bytes too early.

```
For (i=0; i<16; i++)
{
   acAESwork[i] = acAESdata[i] ^ acAESkey[i];
}
```

- Assuming key and data are already masked. acAESwork will be its non-initialised state (zero if we are lucky).
- By collision we can find $K_{112-119} \oplus R_1 \oplus R_2$ and $K_{120-128} \oplus R_1 \oplus R_2$
- We therefore know $K_{112-119} \oplus K_{120-128}$

CFA on a DPA-Resistant Algorithm

- DPA countermeasures include a random order.
- Assuming key and data are already masked, acAESwork will be its non-initialised state (zero if we are lucky).
- Gives $\binom{16}{2} = 120$ different combinations.
- A key-dependent dictionary of 2^{23} entries can be constructed for this (350 Mb) i.e. dictionary needs to be constructed with device under attack – one week using a smart card).
- Can use a fragment of the dictionary, and acquire more data.

CFA on a DPA-Resistant Algorithm

- This gives pairs of masked values (with different randoms) at different indexes.
- For example:

–	–	–	$K_{112-119} \oplus R$	$K_{120-128} \oplus R$
–	$K_{96-103} \oplus R'$	–	$K_{112-119} \oplus R'$	–

- To convert the mask:

$$
\begin{aligned}
M &= (K_{112-119} \oplus R) \oplus (K_{112-119} \oplus R') \\
&= R \oplus R'
\end{aligned}
$$

$$K_{96-103} \oplus R' \oplus M = K_{96-103} \oplus R$$

- With enough samples the whole key (masked with *R*) can be found, leaving an exhaustive search of 2^8 different keys.

CFA on a DPA-Resistant Algorithm

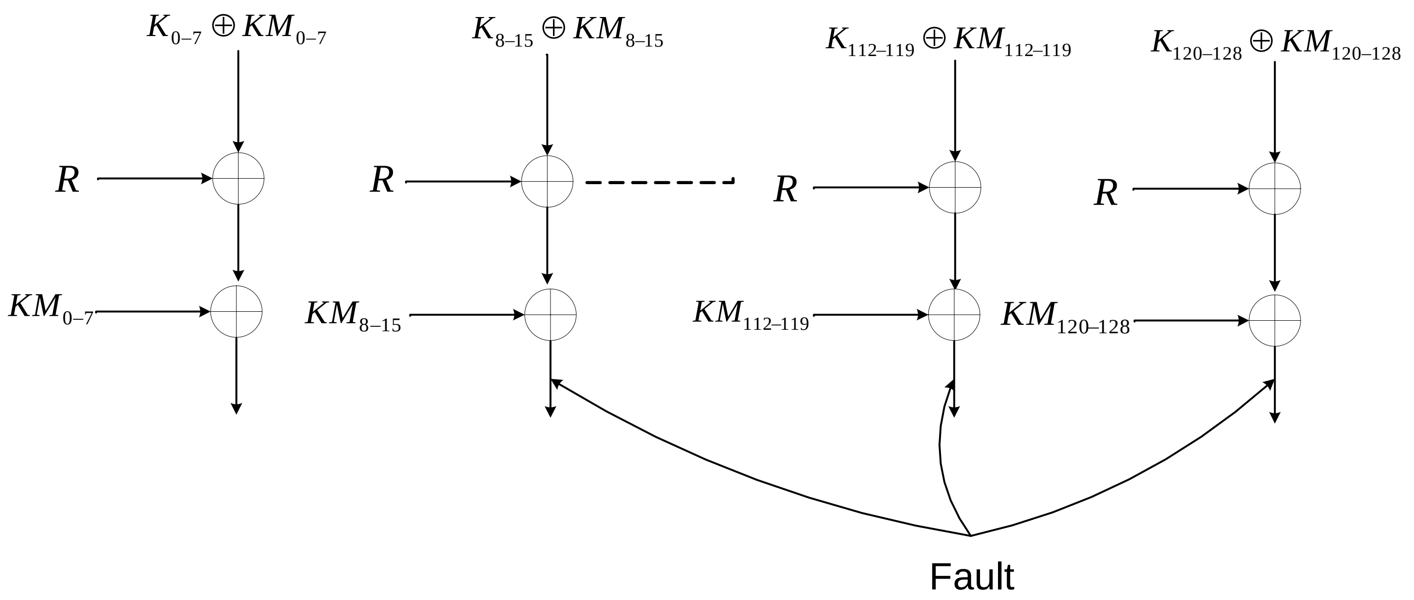
- Scanning the loop found 71 pairs of key bytes, e.g.

```
F11A----------------------------
00------00----------------------
8E--------21--------------------
00--------88--------------------
5B----------C7------------------
24------------A9----------------
00------------88----------------
--B4----------D2----------------
--06------------9F--------------
--B6--------------------6B------
----7F6E------------------------
----B2--D4----------------------
```

- Giving 31 different keys I.e. no XOR difference.
- Leading to a search of approx. 2^{12} different keys,

CFA on key masking

- Keys are often stored in non-volatile memory XORed with a (unchanging) random of the same bit length as the key.
- This random needs to be replaced with *R* before the DPA-resistant algorthm can be called.
- A different attack can be applied to this mechanism.



CFA on key masking

- Fault on two byte too a fixed value (zero for example).
- Gives (in memory):

$K \oplus R$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- In algorithm:

K	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

- A dictionary of the 2^{16} combinations of *K* and *R* can be created.

- As before, the random order means that a random byte will be transfered correctly.

- Key-independent dictionary size of 2^{20} (40 MB).

CFA on key masking

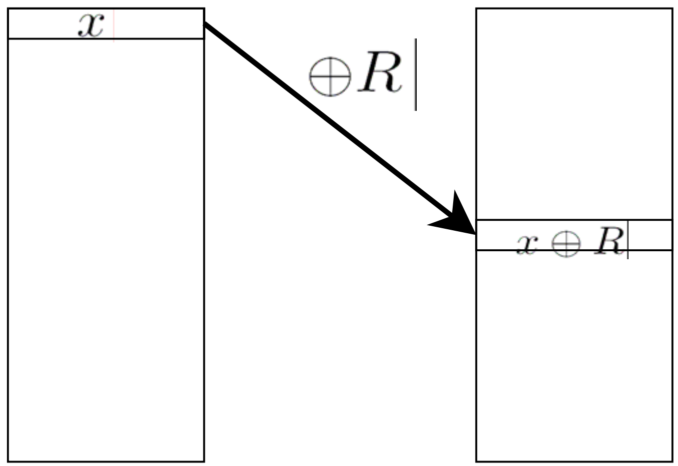
- Attacking this loop produced 60 collisions e.g.

Ciphertext	**Index**	**Key Byte**
F81E9C53601A9D27BF14A439CFB89329	13	CC
9589F701F254450A95B9ACE3F56CC525	8	77
D5B7691596141F967B8933B3EC19D80E	5	44
FA88725F36EED9A99DA1BC318861F1CA	5	44
0CA8BF1D394DA73B5DB36C03C6F19540	16	FF
7ED1484607BBCF135F90B460DADA1FCD	4	33
A1EDC486CAD6C32EA16DE3CFDD309201	4	33
0CA8BF1D394DA73B5DB36C03C6F19540	16	FF
B2C5E49D5B5AE03478A06D7212151870	16	FF
96FA183C668222C6094A5D5D2791F489	1	FA

- Found the key instantly as there was no contradictory information.

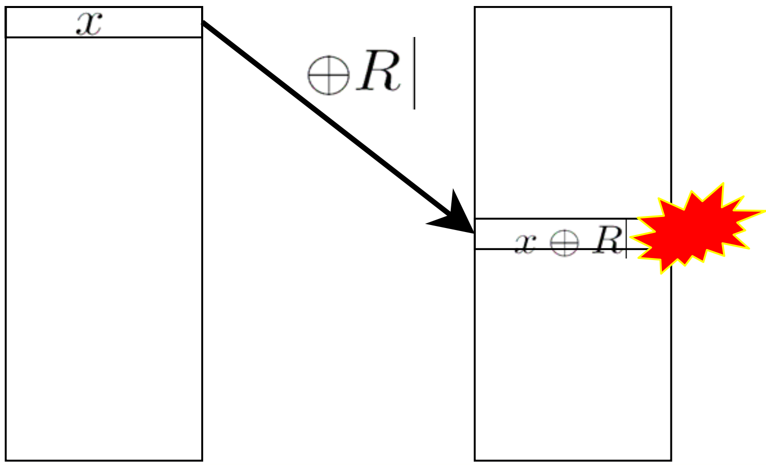
Fault Injection during S-Box Construction

- DPA resistant DES constructs S-Boxes in RAM.



Fault Injection during S-Box Construction

- The elements written can be changed.
- Then we can say:
  - Element used if ciphertext is corrupt.
  - Otherwise element is not used.
- We can then construct hypothesis' on the first subkey.
  - e.g. If the first element of the first S-Box is corrupt and produces a corrupt ciphertext, then the first six bits of the key could be.



$$K_{0-5} = 0 \oplus M_{0-5}$$

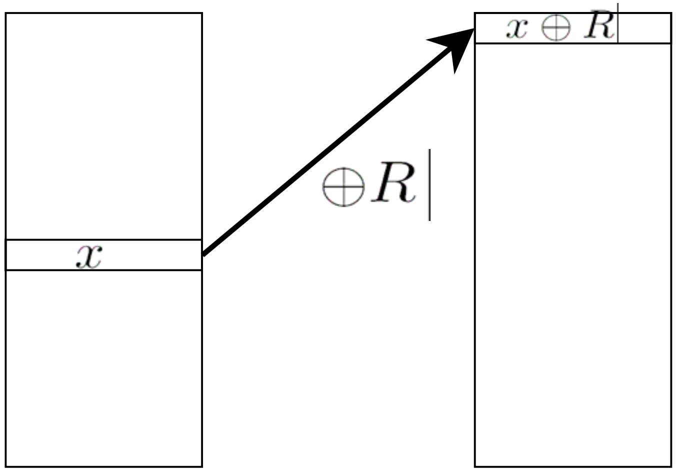
Fault Injection during S-Box Construction

- Changing S-Box values one by one a list of hypotheses can be constructed on the first subkey (for example).
- Repeating with a different message leads to a different list of hypotheses.
- The actual subkey is in the intersection of the two lists

- Attack tools for target chip.
  - Using duty cycle bug to change RAM writing. Found while trying to implement Differential Fault Analysis on the target chip.
  - Tools designed to exploit this bug found a DES key in approx. 45 minutes.

Countermeasures

- **Construct S-boxes in a random order:**



Fault Injection during S-Box Construction

- In the case where S-Box creation is randomised, can use Differential Fault Analysis [Biham and Shamir, 1997] to attack DES.

- If a random S-Box values is changed the probability that this value is used in the fifteenth round is $\left(\frac{63}{64}\right)^{15}\frac{1}{64} = 0.0123.$

- In the DES anti-DPA two values are changed (compressed S-box), leading to probability $2\left(\frac{63}{64}\right)^{15}\frac{1}{64}\left(\frac{63}{64}\right)^{16} = 0.0192.$

- False positives do occur but merely add noise.

Simulated Example

$P\text{-}Perm^{-1}(R15)$ Ciphertext

```
e837cd41 -> 6fb23ead0534752b
```

Correct Ciphertext

```
e837fd41 -> 6eb63ead55b0753d
e837cd31 -> 7b3276ac2024302b
e837cda1 -> 79a776ec2124743b
e837ad41 -> 6eb21ead51307429
eb37cd41 -> 6fb32ead0534fd2e
e937cd41 -> 6fb22ead05347d2a
e837cd31 -> 7b3276ac2024302b
9837cd41 -> 6fa82eec45616422
e8a7cd41 -> 67f2babc0530312b
d837cd41 -> 6fa83eec4521752b
```

Faulty Ciphertexts

Results

- Implementation with randomised S-Boxes takes 8 minutes.
- Implementation with all anti-DPA countermeasures takes 20 minutes.
  - Random Delays.
  - Random Order.

Countermeasures

- S-box construction requires a checksum (at least 16 bits), as if more than one S-box element is changed, an x-bit checksum will be correct with probability 1/x.

- Repeating the initial functions, (only the rounds of an algorithm need to be repeated to prevent DFA).

- Initialise “work” areas of memory with random values, each byte needs to be different. Using an LFSR may be risky if the algorithm is known – just adds complexity.