A Study of Persistent Fault Analysis

SPACE 2019, Gandhinagar India

Andrea Caforio and Subhadeep Banik

December 5, 2019

LASEC, Ecole Polytechnique Fédérale de Lausanne, Switzerland

Fault Analysis Retrospective i

- First popularized by Boneh et al., exploiting an erroneous CRT computation within RSA [BDL97].
- Shortly thereafter, Biham and Shamir gave a method to leverage the difference between a faulty and correct DES ciphertext, which became known as differential fault analysis [BS97].

Fault Analysis Retrospective ii

- DFA usually requires very few ciphertext pairs but necessitates precise fault injections.
- Many variations have been proposed since, such as algebraic fault analysis and fault rate analysis [CJW10; Wan+13].
- Other attacks like statistical fault analysis and fault sensitivity analysis are statistical in nature [Riv09; Fuh+13].

Fault Duration

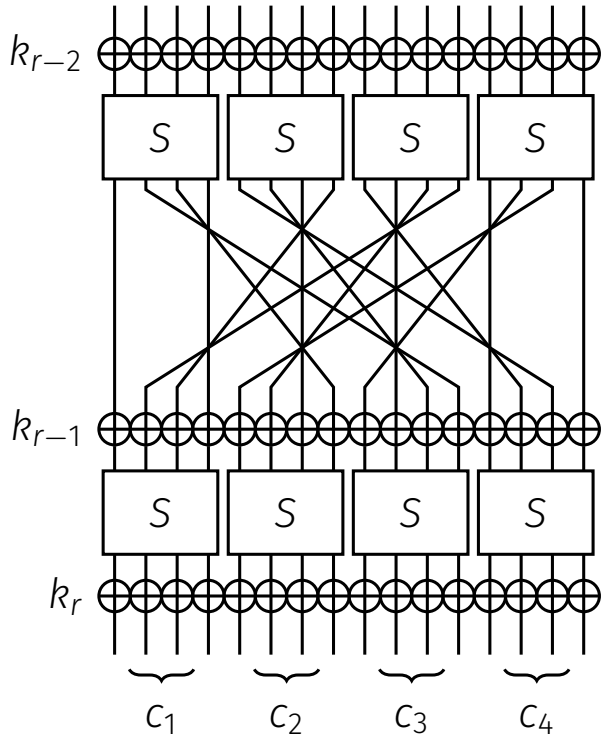
In general, it is possible to group the duration of faults into three duration categories ranging from injections that last a single computation to permanent defects of the underlying hardware.

- Transient
- Permanent
- Persistent

Modus Operandi i

- Persistent fault analysis gained traction at CHES 2018, in a work by Zhang et al. [Zha+18].
- They showed how to exploit a faulty s-box in substitution-permutation networks to recover the last round-key.
- It is a ciphertext-only attack that on average requires ≈ 1500 ciphertexts to be effective against AES.

Modus Operandi ii



Consider a toy SPN whose round-function consists of a key-mixing, permutation, and substitution layer, repeated over r rounds [Jea16].

Modus Operandi iii

- The intermediate state is composed of W words of b bits that are fed to the bijective $b \times b$ s-box $S : \{0, \ldots, 2^b - 1\} \rightarrow \{0, \ldots, 2^b - 1\}$.
- Suppose the adversary managed to overwrite some element $S(j) = u$ for some $0 \leq j < 2^b$ such that $S(j) = v$ with $u \neq v$.

Modus Operandi iv

Let $X_i \in \{0, \ldots, 2^b - 1\}$ be a random variable describing the i-th input word to the last round. It holds that

$$\Pr[S(X_i) = u] = 0.$$

The overall of distribution $S(X_i)$ is skewed away from uniform, namely the element v appears twice as often with probability

$$\Pr[S(X_i) = v] = 2^{1-b}.$$

All the others elements remain at their initial probability value

$$\Pr[S(X_i) = t] = 2^{-b},\ t \in \{0, \ldots, 2^b - 1\} \setminus \{u, v\}.$$

Zhang et al.'s Strategies i

Zhang et al. [Zha+18] propose three different algorithms, in order to recover one word of the last round key that exploit the non-uniform distribution of $S(X_i)$.

Zhang et al.'s Strategies ii

Let $T = \{t_1, \ldots, t_n\}$ be a collection of n collected i-th ciphertext words and denote by

$$\text{Count}(t) = \sum_{i=1}^{n} \mathbb{I}\{t_i = t\}$$

the number of occurrences of word t, where $\mathbb{I}$ designates an indicator function that is one if $t_i = t$ and zero otherwise.

Zhang et al.'s First Strategy

```
1 Counts ← [0, . . . , 0]
2 for t = 0 t < 2^b; t ← t + 1 do
3 |   Counts(t) ← Count(t)
4 end
5 for t = 0 t < 2^b; t ← t + 1 do
6 |   if Counts(t) = 0 then
7 |   |   return t ⊕ u
8 |   end
9 end
```

The first strategy is applicable when the overwritten s-box entry u is known to the adversary. u never occurs in the output of S, so $u \oplus k_i$ can not appear in the collected list of i-th ciphertext words T.

Zhang et al.'s Second Strategy

```
1  Counts ← [0, ..., 0]
2  for t = 0 t < 2^b; t ← t + 1 do
3  |   Counts(t) ← Count(t)
4  end
5  k ← {0, ..., 2^b − 1}
6  for t = 0 t < 2^b; t ← t + 1 do
7  |   if Counts(t) ≠ 0 then
8  |   |   k ← k \ {u ⊕ t}
9  |   end
10 end
11 return k
```

The second strategy also assumes that u is known to the adversary. It proceeds by eliminating impossible key candidates for which $\mathrm{Count}(t) \neq 0$, i.e., keys for which $k_i \oplus t = u$.

Zhang et al.'s Third Strategy

```
1 Counts ← [0, . . . , 0]
2 for t = 0 t < 2^b; t ← t + 1 do
3 |   Counts(t) ← Count(t)
4 end
5 t_max ← arg max{Counts}
6 return t_max ⊕ v
```

The third strategy requires the overwriting element v. Hence, $v \oplus k_i = t$ appears twice as much in the ciphertext output as other values, which identifies k_i.

Complexity

In the presence of a single injection the average number of ciphertexts that are required, until a key-word is exactly recovered, is given by the Coupon Collector's Problem, i.e.,

$$(2^n - 1) \cdot H_{2^n - 1},$$

where n is the word size in bits and H_i is the i-th Harmonic number.

Multiple Faults

In the presence of multiple faults, a key-word cannot be exactly identified anymore since multiple s-box elements do not occur in the output anymore.

# Faults	1	2	3	4	5	6	7	8
Key Candidates	1	2^{16}	$2^{25.36}$	2^{32}	$2^{37.15}$	$2^{41.36}$	$2^{44.92}$	2^{48}

Feistel Schemes

- In general, the standard techniques of persistent fault analysis do not apply to Feistel networks since both the left and right side of the output are masked by previous round function outputs.
- Persistent fault injections can still pose a significant danger, if we loosen the ciphertext-only requirement and allow the re-encryption of plaintexts.

Reverse-Engineering Retrospective i

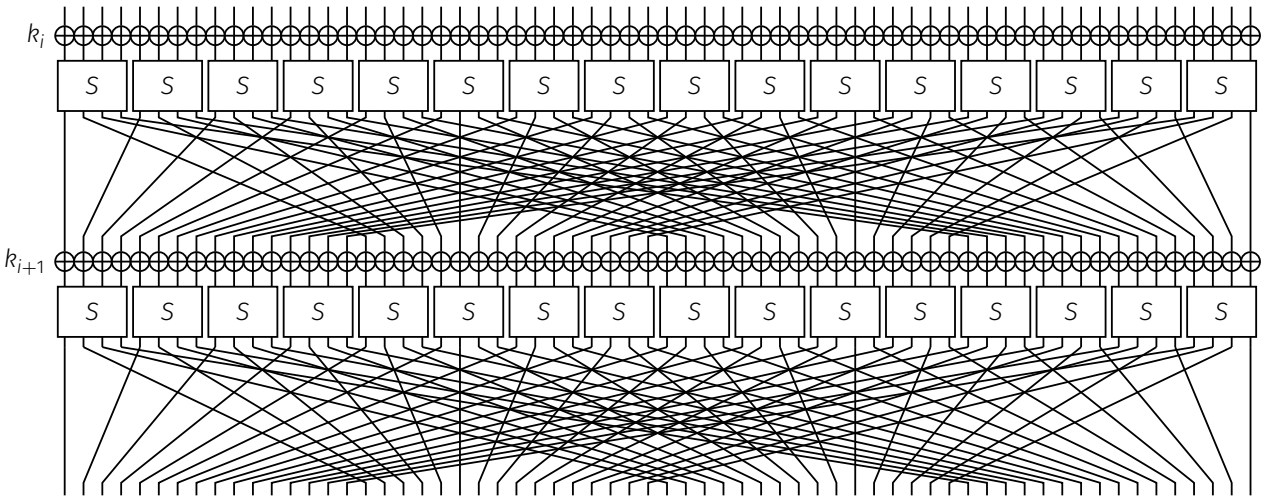
- Reverse-engineering block ciphers is still a relevant craft. The Kuznyechik cipher, selected as the new standard symmetric encryption algorithm for the Russian Federation in 2015, did not fully disclose some of its design rationales and was subsequently reverse-engineered [Shi+14; BPU16].
- In an earlier work, Borghoff et al. [Bor+11] already showed how to cryptanalyse reduced-rounds PRESENT variants whose substitution layer is key-dependent, thus hidden as well. Future work; is it possible to improve this attack using persistent faults?

Reverse-Engineering Retrospective ii

- The idea of leveraging fault injections for reverse-engineering, short FIRE, was introduced by San Pedro et al. [SSG11] in an attempt to use a corrupted last round computation of either AES or DES to recover their hidden s-boxes.
- Other reverse engineering proposals are based on ineffective fault analysis [Cla+15] and integral cryptanalysis [Tie+15].

PRESENT i

PRESENT is an ultra-lightweight block cipher designed by Bogdanov et al. [Bog+07] that operates on 64-bit blocks with a key size of either 80 or 128 bits. It operates over 31 rounds with a substitution layer consisting of a single 4×4 s-box that is applied on all 16 nibbles of the intermediate state and a bit-permutation layer [Jea16].



PRESENT ii

1 **for** $i = 1;\ i \leq 32;\ i \leftarrow i + 1$ **do**
2 $\quad K_i \leftarrow |k_{79}k_{78}\ldots k_{16}|$
3 $\quad |k_{79}k_{78}\ldots k_1k_0| \leftarrow |k_{18}k_{17}\ldots k_{20}k_{19}|$
4 $\quad |k_{79}k_{78}k_{77}k_{76}| \leftarrow S(|k_{79}k_{78}k_{77}k_{76}|)$
5 $\quad |k_{19}k_{18}k_{17}k_{16}k_{75}| \leftarrow |k_{19}k_{18}k_{17}k_{16}k_{75}| \oplus i$
6 **end**

Let $|k_{79}k_{78}\ldots k_1k_0|$ be the individual bits of the master key in big endian notation. In each round the 64 most significant bits yield the current round-key.

PRESENT 16-Round S-Box Recovery i

Due to its simplicity, the PRESENT key-schedule exhibits some peculiarities. For instance, Hernandez et al. [HPA12] showed that there exist keys that expand into very similar round keys.

PRESENT 16-Round S-Box Recovery ii

- During the first 16 rounds no key bit enters the substitution box more than once, the same is true for the latter 16 rounds. Thus no key bit enters the s-box more than twice during the full key-schedule procedure.
- As a consequence, it is possible to compute keys that only access a single s-box element during the first half which in turn leads to the fact that the pattern of the latter half of s-box accesses is entirely determined by this particular s-box value that was accessed during the first half.

PRESENT 16-Round S-Box Recovery iii

Definition (Low-Diffusion Key). *A low-diffusion key $\widetilde{K}$ is a PRESENT master key that, if fed into the key schedule routine, causes only one element of the s-box table to be accessed during the first 16 key-schedule rounds.*

PRESENT 16-Round S-Box Recovery iv

```
1  S(i) ← 0, for 0 ≤ i < 16
2  for i = 0; i < 16; i ← i + 1 do
3  |   k ← PFA(E_{K̃_i})
4  |   for j = 0; j < 16; j ← j + 1 do
5  |   |   S(i) = j
6  |   |   if KeySchedule_S(K̃_i) = k then
7  |   |   |   break
8  |   |   end
9  |   end
10 end
```

Given a faulty device E, for each low-diffusion key $\widetilde{K}_i$, $0 \leq i < 16$ we recover the last round key k through PFA then iterate over all possible values of $S(i) = j$ and compare whether an offline key-schedule calculation is equal to k.

PRESENT Full-Round S-Box Recovery i

Definition (Access Rate). *The access rate of a low-diffusion key, denoted by* $r_{\widetilde{K}_i}(j)$*, is the number of accessed s-box elements by the key* $\widetilde{K}_i$ *during the key-schedule routine when* $S(i) = j$.

PRESENT Full-Round S-Box Recovery ii

i	$\widetilde{K}_i$	$r_{\widetilde{K}_i}(j)$															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0x037bf04d5c0567402460	16	15	13	13	11	11	11	11	15	14	12	13	11	11	11	11
1	0x026ae06f7e012300ace8	16	15	13	13	11	11	11	11	16	15	13	13	11	11	11	11
2	0x0159d009180defc13570	16	15	13	13	11	11	11	11	15	14	12	13	11	11	11	11
3	0x0048c02b3a09ab81bdf8	16	15	13	13	11	11	11	11	15	14	12	13	11	11	11	11
4	0x073fb0c5d41476420640	16	15	13	13	11	11	11	11	15	14	13	14	11	11	11	11
5	0x062ea0e7f61032028ec8	16	15	14	14	11	11	11	11	15	14	12	13	11	11	11	11
6	0x051d9081901cfec31750	16	16	14	14	11	11	11	11	15	14	13	14	11	11	11	11
7	0x040c80a3b218ba839fd8	16	15	14	14	11	11	11	11	15	15	13	14	11	11	11	11
8	0x0bf3715c4c2745446020	16	15	13	13	12	12	12	11	15	14	12	13	12	12	11	11
9	0x0ae2617e6e230104e8a8	16	15	13	13	12	12	11	11	15	14	12	13	12	12	12	11
10	0x09d15118082fcdc57130	16	15	13	13	12	12	11	12	15	14	12	13	12	12	11	11
11	0x08c0413a2a2b8985f9b8	16	15	13	13	12	12	11	11	15	14	12	13	12	12	11	12
12	0x0fb731d4c43654464200	16	15	13	13	12	11	12	12	15	14	12	13	11	11	12	12
13	0x0ea621f6e6321006ca88	16	15	13	13	11	11	12	12	15	14	12	13	12	11	12	12
14	0x0d951190803edcc75310	16	15	13	13	11	12	12	12	15	14	12	13	11	11	12	12
15	0x0c8401b2a23a9887db98	16	15	13	13	11	11	12	12	15	14	12	13	11	12	12	12

PRESENT Full-Round S-Box Recovery iii

Definition (Access Pattern). *Denote by $p_{\widetilde{K}_i}(j)$ the set of accessed s-box entries during the key-schedule for low-diffusion key $\widetilde{K}_i$ with $S(i) = j$.*

PRESENT Full-Round S-Box Recovery iv

1 Choose $i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
2 Overwrite $S(i) = j$ such that $r_{\widetilde{K}_i}(j) = 11$
3 $k, v \leftarrow \mathsf{PFA}(E_{\widetilde{K}_i})$
4 $L \leftarrow \{0, \ldots, 15\} \setminus \{v\}$
5 $S'(l) = 0,\ 0 \leq l \leq 15;\ S'(i) = j$
6 **foreach** $\pi \in \Pi_{15,10}(L)$ **do**
7 $\quad z \leftarrow 0$
8 $\quad$ **foreach** $p \in p_{\widetilde{K}_i}(j)$ **do**
9 $\quad\quad S'(p) \leftarrow \pi(z),\ z \leftarrow z + 1$
10 $\quad$ **end**
11 $\quad$ **if** $\mathtt{KeySchedule}_{S'}(\widetilde{K}_i) = k$ **then**
12 $\quad\quad$ **return** S'
13 $\quad$ **end**
14 **end**

Let the injected fault be of the form $S(i) = j$ such that $r_{\widetilde{K}_i}(j) = 11$. In this case there are only 10 remaining s-box entries in the entire key schedule routine that are accessed for the low-diffusion key $\widetilde{K}_i$.

PRESENT Full-Round S-Box Recovery v

- There are $\frac{15!}{5!} \approx 2^{33.34}$ potential arrangements that have to be checked in the worst case since due to the low-diffusion key and the fault injection only 10 s-box entries need to be assigned from a set of 15 potential values.
- This results in a reduction by a factor of 120 compared to the brute-force approach ($15! \approx 2^{41}$). The remaining 5 elements can then be safely brute-forced. Thus the computational complexity of this method is around $2^{33.34} + 5! \approx 2^{33.34}$ offline key schedule computations.

PRESENT Full-Round S-Box Recovery vi

The result from before can be generalized to multiple injections where some random elements of the s-box are overwritten with the same value such that the last round-key can be exactly retrieved.

	Trials						
	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$
Brute-Force	$2^{43.3}$	$2^{41.7}$	$2^{39.7}$	$2^{37.3}$	$2^{34.8}$	$2^{31.9}$	$2^{28.9}$
Our Result	$2^{37.6}$	$2^{36.2}$	$2^{34.5}$	$2^{32.6}$	$2^{30.5}$	$2^{28.2}$	$2^{25.7}$

PRESENT Permutation Layer Recovery i

On paper, recovering the permutation layer appears to be a harder task due to the sheer amount $64! \approx 2^{296}$ of possibilities.

PRESENT Permutation Layer Recovery ii

Let $C = \{c_1, \ldots, c_n\}$ be a set of n ciphertexts from the faulty device. Denote by $|c(i), c(j), c(k), c(l)|$ the nibble that is created by extracting the bits i, j, k, l from a ciphertext c. Further, denote by $C(i, j, k, l)$ the set of nibbles that is generated by extracting bits i, j, k, l from each ciphertext, i.e.,

$$C(i,j,k,l)=\{|c_1(i),c_1(j),c_1(k),c_1(l)|,\ldots,|c_n(i),c_n(j),c_n(k),c_n(l)|\}.$$

PRESENT Permutation Layer Recovery iii

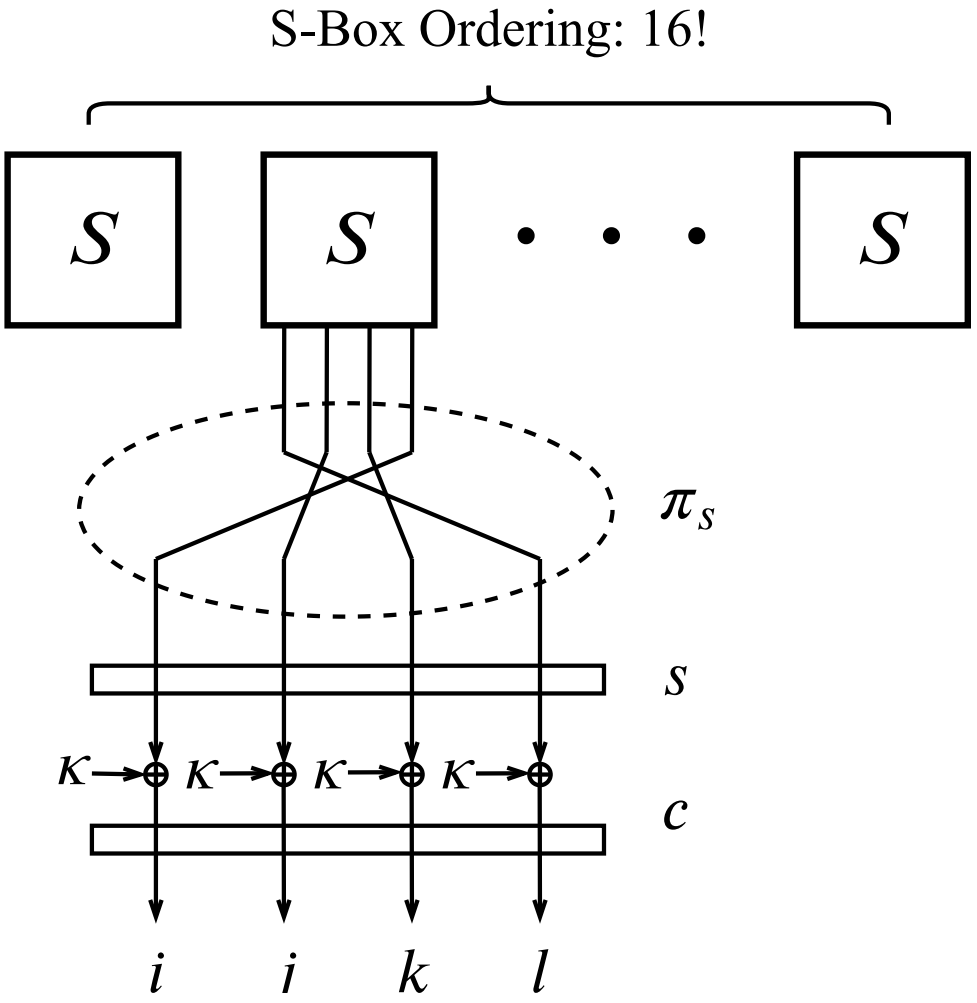
```
1 L ← {0, 1, 2, . . . , 63}
2 C ← {c_1, . . . , c_n}
3 foreach π_0, π_1, π_2, π_3 ∈ Π_4(L) do
4     if |C(π_0, π_1, π_2, π_3)| < 16 then
5         output |π_0, π_1, π_2, π_3|
6         L ← L \ |π_0, π_1, π_2, π_3|
7     end
8 end
```

Suppose an random entry in the S-box is overwritten due to a fault. If each bit in a nibble i, j, k, l stems from the same s-box then we have necessarily $|C(i, j, k, l)| < 16$ due to the persistent fault injection.

PRESENT Permutation Layer Recovery iv

- The expected number of ciphertexts that are required until an incorrect nibble can be excluded is again given by the Coupon Collector's Problem and stands at around 55.
- The remaining complexity is $24^{16} \times 16! \approx 2^{118}$ to recover the entire permutation.

PRESENT Permutation Layer Recovery v



PRESENT Permutation Layer Recovery vi

- Potential permutation fixed points can be recovered through linear cryptanalysis (Matsui's Piling-up Lemma).
- Through further injections we can recover the 4-bit permutation of after each s-box. Note that if element 0 or 15 of the s-box are overwritten we can recover the last round-key.
- The remaining ordering of the s-boxes $16! \approx 2^{45}$ can then be brute-forced.

AES S-Box Recovery

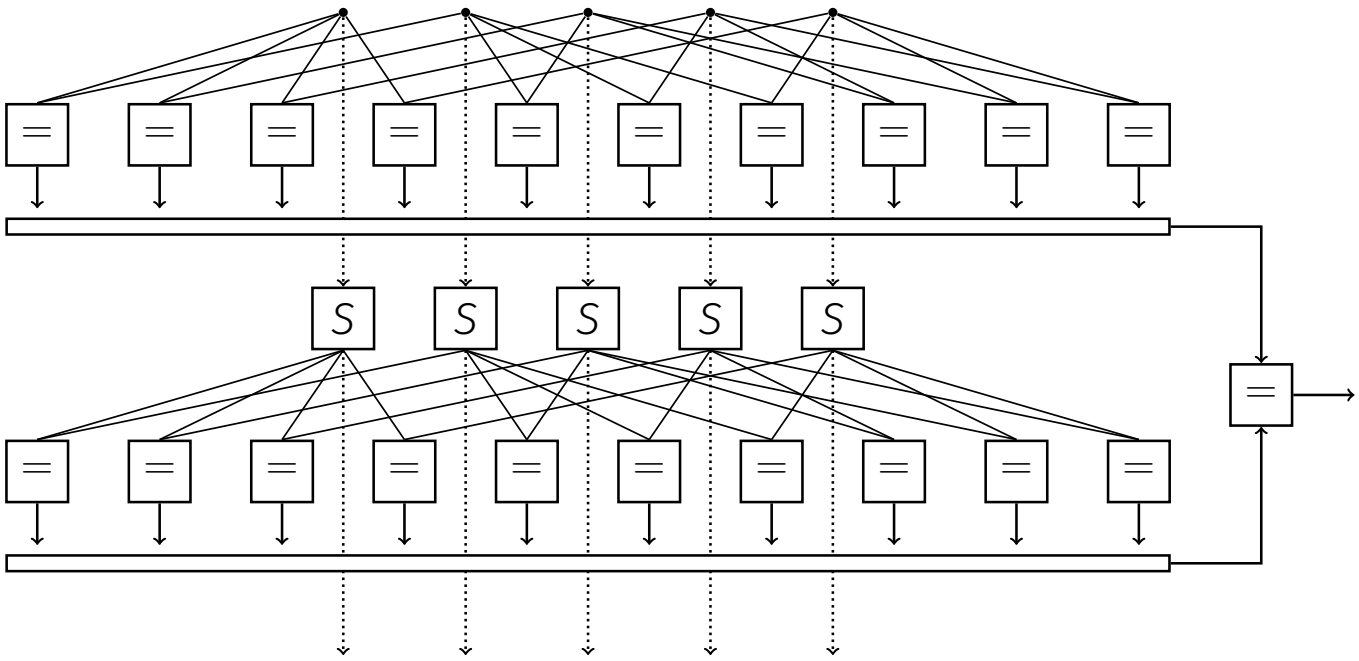
The techniques for PRESENT s-box recovery can be extended to AES in a reduced-round setting. However, only a handful entries can be recovered with each injection.

A Dedicated Countermeasure i

- The situation looks quite dire. Persistent fault injections remain devastating even in the presence of existing fault detection countermeasures.
- **Rule of Thumb.** Merely reacting to individual detected errors is not enough (correct ciphertexts leak information). The fault must be detected as early as possible and the affected device must engage in outputting safe ciphertexts until a subsequent reboot.

A Dedicated Countermeasure ii

We propose a novel, low-overhead and dedicated countermeasure against persistent fault injections in bijective s-boxes.



A Dedicated Countermeasure iii

Lemma. *Denote by S a faulty $n \times n$ s-box with t altered entries $v_1, \ldots, v_t$ such that $S(v_i) = S(u_i)$ for some $u_i \notin \{v_1, v_2, \ldots, v_t\}$. Let $p_{n,m,t}$ be the probability that at least one pair of equal entries is accessed in one round. The value is given by*

$$
\begin{aligned}
p_{n,m,t} = 1 &- \left(\frac{2^n - 2t}{2^n}\right)^m \\
&- \sum_{i=1}^{t} \left(2^i \binom{t}{i} \right. \\
&\times \sum_{k_1}^{m} \sum_{k_2}^{m-k_1} \cdots \sum_{k_i}^{m-\sum_{j=1}^{i-1}} \binom{m}{k_1} \cdots \binom{m - \sum_{j=1}^{i-1} k_j}{k_i} \\
&\times \left. \left(\frac{1}{2^n}\right)^{\sum_{j=1}^{i} k_j} \left(\frac{2^n - 2t}{2^n}\right)^{m - \sum_{j=1}^{i} k_j} \right).
\end{aligned}
$$

A Dedicated Countermeasure iv

Corollary. *Given a r-round block cipher whose substitution box bears t equalized entries. Denote by* $p^r_{n,m,t}$ *the probability that an error is detected. It is given by*

$$p^r_{n,m,t} = 1 - (1 - p_{n,m,t})^r.$$

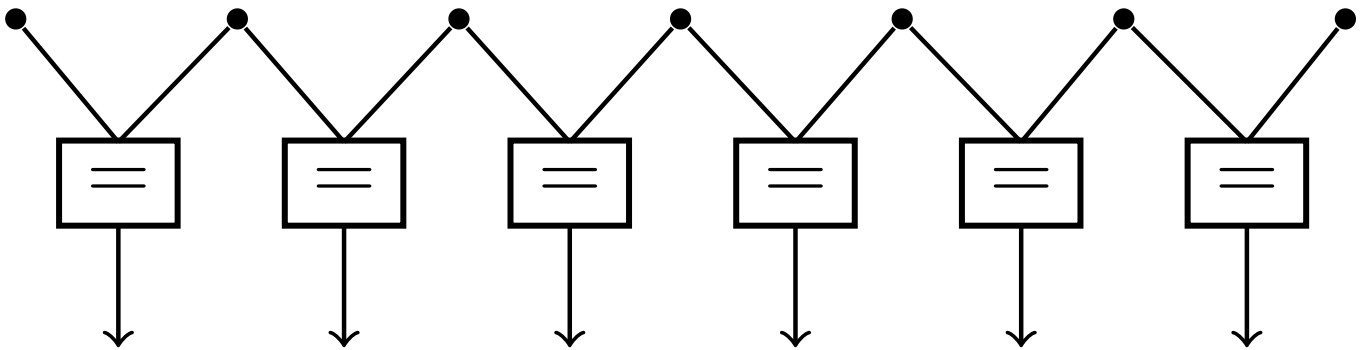
A Dedicated Countermeasure v

For AES, in a single-fault setting, on average $\frac{1}{0.0341} \approx 30$ ciphertexts are required until the fault is detected.

	$p^r_{n,m,t}$					
	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$p^{10}_{8,16,t}$ (AES)	0.0341	0.0671	0.0990	0.1288	0.1597	0.1884
$p^{31}_{4,16,t}$ (PRESENT)	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1

A Dedicated Countermeasure vi

For both AES and PRESENT it is necessary to perform $\binom{16}{2} = 120$ pairwise byte or nibble comparisons. This necessitates a rather large overhead in hardware implementations.



A Dedicated Countermeasure vii

- Only doing 15 comparisons changes the detection probability, which is now lower-bounded by $\frac{p_{n,m,t}}{8}$.
- For AES, this bound is tight, with an overall detection probability over 10 rounds given by

$$1 - (1 - \frac{p_{8,16,t}}{8})^{10} \approx 0.004335.$$

- For PRESENT, this bound is not tight, it can be shown that in a single-fault setting the detection probability for one encryption is roughly 0.97

A Dedicated Countermeasure viii

For AES, it performs well on average but it is especially effective in the presence of more than one persistent fault injection.

Residual Key Entropy Probabilities (AES)

	$[0, 15)$	$[15, 30)$	$[30, 45)$	$[45, 60)$	$[60, 75)$	$[75, 90)$	$[90, 105)$	$[105, 120)$	$[120, 128]$
$t = 1$	0.00195	0.00361	0.00856	0.01699	0.03708	0.07811	0.16763	0.35617	0.32989
$t = 2$	0.00000	0.00000	0.00008	0.00071	0.00343	0.01654	0.07612	0.35053	0.55261
$t = 3$	0.00000	0.00000	0.00000	0.00002	0.00020	0.00261	0.02669	0.26949	0.70104
$t = 4$	0.00000	0.00000	0.00000	0.00000	0.00001	0.00032	0.00860	0.18980	0.80127

A Dedicated Countermeasure ix

For PRESENT, we have a very strong protection already in the single-fault setting.

Residual Key Entropy Probabilities (PRESENT)

	$[56, 57)$	$[57, 58)$	$[58, 59)$	$[60, 61)$	$[61, 62)$	$[62, 63)$	$[63, 64)$	64
$t = 1$	0.00000	0.00000	0.00002	0.00032	0.00067	0.03099	0.00000	0.96798
$t = 2$	0.00000	0.00000	0.00000	0.00001	0.00001	0.00085	0.00000	0.99914

A Dedicated Countermeasure x

The reduced comparison network can be implemented with very low-overhead for both AES and PRESENT.

#	Architecture	Type	Area (GE)	Overhead (in %)	Latency (cycles)	Energy (nJ)	TP_{max} (Gbps)
	AES						
1	Round based	Unprotected	12876		11	0.72	2.371
		Protected	13615	5.7	11	0.78	1.555
2	8-bit Serial	Unprotected	2060		246	3.20	0.086
		Protected	2143	4.0	246	3.23	0.075
	PRESENT						
1	Round based	Unprotected	1316		33	0.19	1.598
		Protected	1713	30.2	33	0.29	1.050
2	4-bit Serial	Unprotected	892		564	2.50	0.052
		Protected	963	7.9	564	2.71	0.046

Bibliography i

Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. “On the Importance of Checking Cryptographic Protocols for Faults”. In: *Advances in Cryptology — EUROCRYPT ’97*. Ed. by Walter Fumy. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 37–51. ISBN: 978-3-540-69053-5.

A. Bogdanov et al. “PRESENT: An Ultra-Lightweight Block Cipher”. In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466. ISBN: 978-3-540-74735-2.

Bibliography ii

Julia Borghoff et al. “Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes”. In: *Fast Software Encryption*. Ed. by Antoine Joux. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 270–289. ISBN: 978-3-642-21702-9.

Alex Biryukov, Léo Perrin, and Aleksei Udovenko. “Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2016, pp. 372–402.

Bibliography iii

Eli Biham and Adi Shamir. “Differential fault analysis of secret key cryptosystems”. In: *Advances in Cryptology — CRYPTO ’97*. Ed. by Burton S. Kaliski. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 513–525. ISBN: 978-3-540-69528-8.

Nicolas T Courtois, Keith Jackson, and David Ware. “Fault-algebraic attacks on inner rounds of DES”. In: *E-Smart’10 Proceedings: The Future of Digital Security Technologies*. Strategies Telecom and Multimedia. 2010.

Bibliography iv

Christophe Clavier et al. “Complete reverse-engineering of AES-like block ciphers by SCARE and FIRE attacks”. In: *Cryptography and Communications - Discrete Structures, Boolean Functions and Sequences* Issue 1 (Mar. 2015).

Thomas Fuhr et al. “Fault attacks on AES with faulty ciphertexts only”. In: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE. 2013, pp. 108–118.

Bibliography v

Julio Cesar Hernandez-Castro, Pedro Peris-Lopez, and Jean-Philippe Aumasson. “On the Key Schedule Strength of PRESENT”. In: *Data Privacy Management and Autonomous Spontaneus Security*. Ed. by Joaquin Garcia-Alfaro et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 253–263. ISBN: 978-3-642-28879-1.

Jérémy Jean. *TikZ for Cryptographers*. `www.iacr.org/authors/tikz/`. 2016.

Bibliography vi

Matthieu Rivain. “Differential fault analysis on DES middle rounds”. In: *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 457–469.

Vasily Shishkin et al. “Low-weight and hi-end: Draft Russian encryption standard”. In: *CTCrypt’14, 05-06 June 2014, Moscow, Russia. Preproceedings* (2014), pp. 183–188.

Manuel San Pedro, Mate Soos, and Sylvain Guilley. “FIRE: fault injection for reverse engineering”. In: *IFIP International Workshop on Information Security Theory and Practices*. Springer. 2011, pp. 280–293.

Bibliography vii

Tyge Tiessen et al. “Security of the AES with a Secret S-Box”. In: *International Workshop on Fast Software Encryption*. Springer. 2015, pp. 175–189.

An Wang et al. “Fault rate analysis: breaking masked AES hardware implementations efficiently”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 60.8 (2013), pp. 517–521.

Fan Zhang et al. “Persistent Fault Analysis on Block Ciphers”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 150–172.