Related-key differential analysis of the AES

Margot Funk [1]

Joint work with Christina Boura [1] and Patrick Derbez [2]

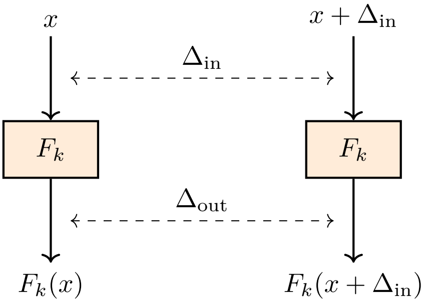
[1]Paris-Saclay University - Versailles University

[2]University of Rennes 1

October 2023

Differential cryptanalysis

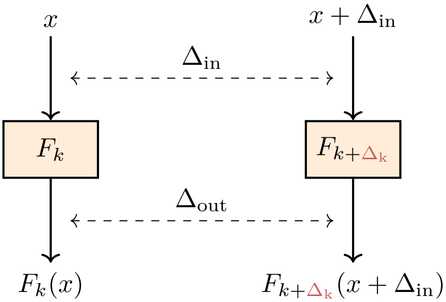
- Exploits a high probability differential distinguisher



single-key differential

Differential cryptanalysis

- Exploits a high probability differential distinguisher

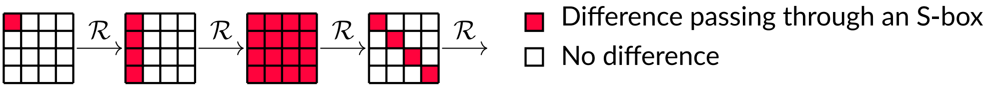


related-key differential

AES differential trails

active S-boxes, max DP of the AES S-box = 2^{-6}

$\hookrightarrow$ bound on the differential probability



4-round truncated differential trail of AES with 25 active S-boxes: $\boldsymbol{p \leq 2^{-25 \times 6}}$

Single-key model VS Related-key model

- **Single-key**: **simple** and **powerful** security proofs.
- **Related-key**: much **weaker**.
  Related-key attacks on the full AES-192 and AES-256, Biryukov et al., 2009

Modeling the AES truncated trails

Use of **generic solvers** (Wu and Wang, 2009 and Mouha et al., 2011)

Problem of finding differential trails $\longrightarrow$ Model $\longrightarrow$ Solver

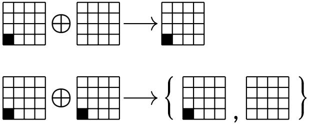
Model

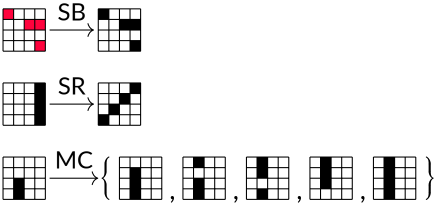
- **Variables:** byte of the truncated trail $\leftrightarrow$ `var` $\in \{0, 1\} \subset \mathbb{Z}$.
- **Objective function:** minimize the sum of variables that pass through an S-box.
- **Set of constraints** *(ex: linear inequalities)*

Modeling the AES truncated trails

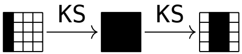
Basic propagation rules ...

XOR of two bytes





... do not necessarily lead to valid truncated trails.

Ex: 

is not instantiable.

Modeling the AES truncated trails

Gérault et al. (2018, 2020), Rouquette et al. (2022)

- Use a Constraint Programming (CP) solver.
- Few seconds or minutes for most of the instances.
- Outperforms previous works:
  - Branch & bound (Biryukov et al., 2010): several weeks for AES–192,
  - Dynamic programming for AES–128 (Fouque et al., 2013).

Dynamic programming for differential bounds on AES

Dynamic programming for differential bounds

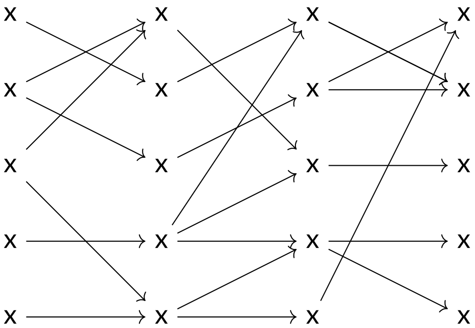
Fouque et al., CRYPTO 2013

- Generic tool based on dynamic programming.
- Complexity easy to understand.
- Application for AES–128: 30 minutes, 60 GB.

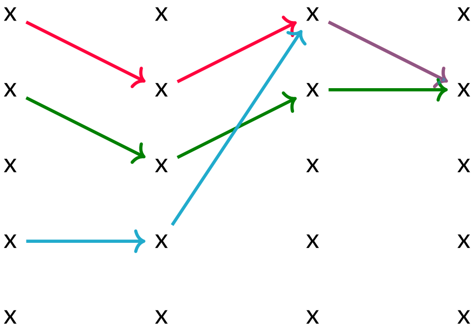
Our work

- Extend the work of Fouque et al. (2013) for all versions of AES.
- Running time comparable to that of the CP approach of Gérault et al. (2018, 2020).

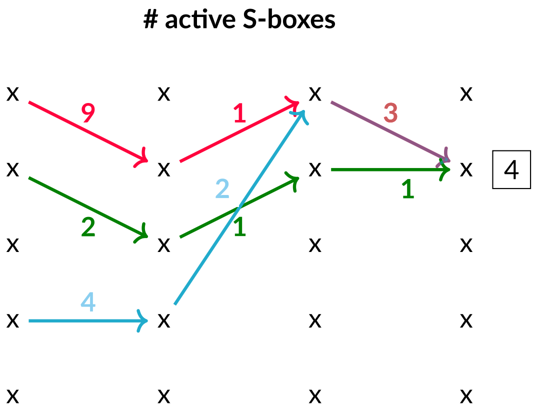
Principle of the dynamic programming algorithm of [FJP13]



Principle of the dynamic programming algorithm of [FJP13]



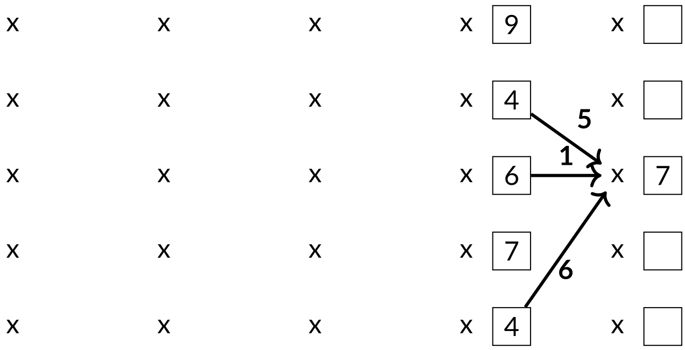
Principle of the dynamic programming algorithm of [FJP13]



Principle of the dynamic programming algorithm of [FJP13]

x	x	x	x	9
x	x	x	x	4
x	x	x	x	6
x	x	x	x	7
x	x	x	x	4

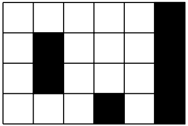
Principle of the dynamic programming algorithm of [FJP13]



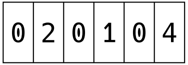
Adapting the dynamic programming algorithm of [FJP13]

- Reduce the memory complexity

Truncated difference



Compressed difference



Truncated differences

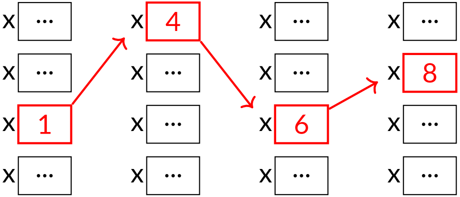
- AES-128: 2^{32}
- AES-192: 2^{40} ✗
- AES-256: 2^{48} ✗

Compressed differences

- AES-128: $2^{18.58}$
- AES-192: $2^{23.22}$
- AES-256: $2^{27.86}$

Adapting the dynamic programming algorithm of [FJP13]

- Integrate constraints over several rounds in a second step.

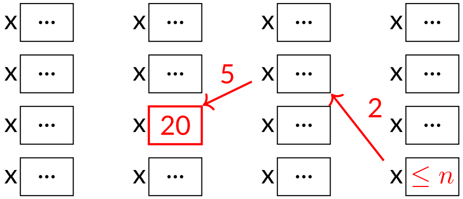


min

$\hookrightarrow$ bound not necessarily tight!

Adapting the dynamic programming algorithm of [FJP13]

- Integrate constraints over several rounds in a second step.



1. Search for a **compressed trail** with n active S-boxes.
   - depth-first search approach in the backward direction
   - check some linear relations
2. Turn it, if possible, into a **truncated trail**.

Complexity and running time

- For the dynamic programming phase:

	Time complexity	Memory (Bytes)
AES-128	$r \times 2^{22.89}$	$(9r - 9) \times 2^{18.58}$
AES-192	$r \times 2^{27.53}$	$(3r - 3) \times 2^{23.22}$
AES-256	$r \times 2^{32.18}$	$(3r - 4) \times 2^{27.86}$

Complexity and running time

Algorithm	Rounds	Min nb of active S-boxes	# trails	Real time (1) (User time)	Time [RGMS22] (2)
AES-128	4	12	1	1s (1s)	31s
	5	17	81	40s (5m6s)	2h24m24s
AES-192	6	10	3	1s (8s)	17s
	7	14	2	1s (9s)	46s
	8	18	4	1m35s (12m37s)	1m23s
	9	24	6	4d5h (20d4h)	30m
AES-256	11	20	4	42s (4m30s)	5m30s
	12	20	4	42s (4m16s)	4m37s
	13	24	4	52s (5m24s)	7m
	14	24	4	50s (5m5s)	9m17s

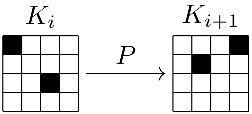
(1) 8-core Ryzen 3700X processor, 3.6 GHz, 32 GB of RAM

(2) 1-core Intel Xeon E5-2630 v4, 3.10 Ghz with 10 cores under a Linux Debian 10 (Buster), 16 GB of RAM (default JVM configuration)

Alternative permutation-based key schedules for AES

Related works

Permutation-based key schedule for AES–128



: efficient in software and hardware.

- Khoo et al., ToSC 2017
- Derbez et al., SAC 2018

Double MILP model

Goal: find a permutation ensuring b **active S-boxes**.

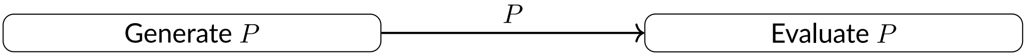
Generate P

Evaluate P

Ensure that P is a permutation.

Double MILP model

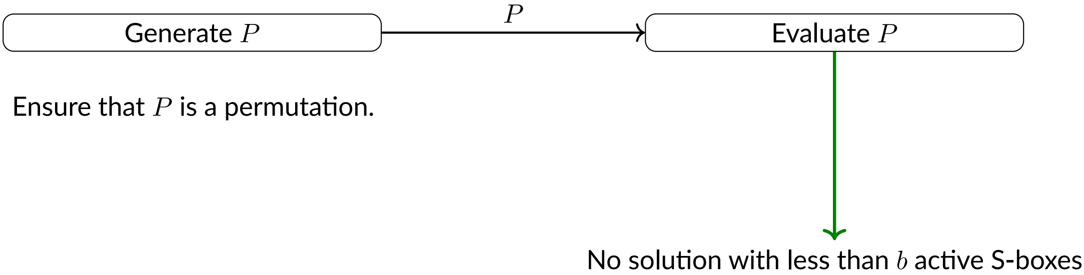
Goal: find a permutation ensuring b **active S-boxes**.



Ensure that P is a permutation.

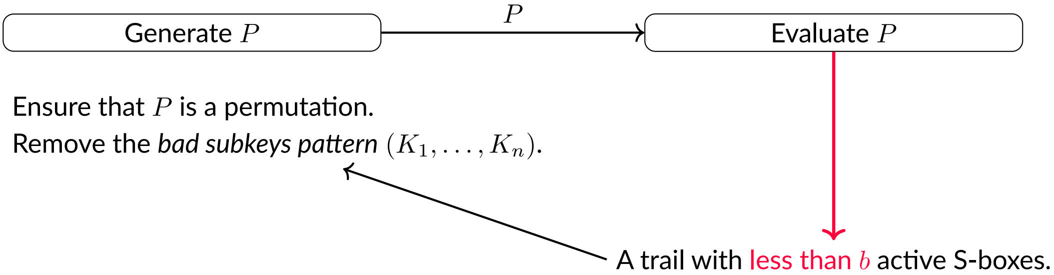
Double MILP model

Goal: find a permutation ensuring b **active S-boxes**.



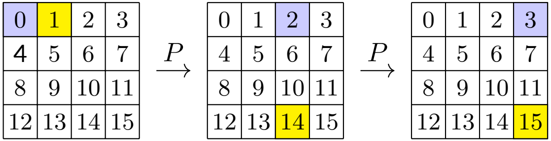
Double MILP model

Goal: find a permutation ensuring b **active S-boxes**.



Removing a bad subkeys pattern

- 1rst idea: forbide the exact trail.



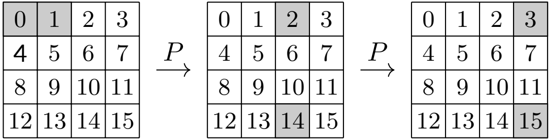
At most 3 of these equalities should be true.

$P(0) = 2 \qquad P(1) = 14$

$P(2) = 3 \qquad P(14) = 15$

Removing a bad subkeys pattern

- 2nd idea: forbide the subkeys pattern.



At most 3 of these equalities should be true.

$$P(0) = 2 \qquad P(1) = 14$$
$$P(1) = 2 \qquad P(0) = 14$$
$$P(2) = 3 \qquad P(14) = 15$$
$$P(14) = 3 \qquad P(2) = 15$$

Results

Rounds	3	4	5	6	7	8	9	10
AES-128	5	12	17					
Khoo et al.	5	10	14	19	23			
P_{128}	5	10	14	20	22			
AES-192	1	4	5	10	14	18	24	29
P_{192}	1	5	10	13	17	22	25	28
AES-256	1	3	3	5	5	10	15	16
P_{256}	1	2	5	10	14	16	22	26

Conclusion and perspectives

The **key schedule** is one of the **less understood components** in block ciphers.

Perspectives

- Clarify the security goals.
- Search for key schedules that are not permutations of bytes.