New Results in Differential Cryptanalysis

Christina Boura

Université de Versailles Saint-Quentin-en-Yvelines

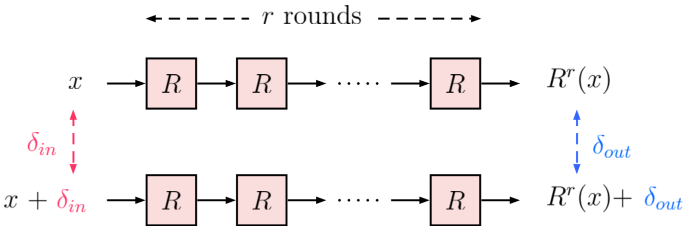
(based on joint-work with Nicolas David, Patrick Derbez, Rachelle Heim and María Naya-Plasencia)

Journées C2

October 19, 2023

Differential cryptanalysis

- Cryptanalysis technique introduced by Biham and Shamir in **1990**.
- Based on the existence of a high-probability **differential** $(\delta_{in}, \delta_{out})$.



- If the probability of $(\delta_{in}, \delta_{out})$ is (much) higher than 2^{-n}, where n is the block size, then we have a differential distinguisher.

Differential attacks

- A differential distinguisher can be used to mount a key recovery attack.
- This technique broke many of the cryptosystems of the 70s-80s, e.g. `DES`, `FEAL`, `Snefru`, `Khafre`, `REDOC-II`, `LOKI`, etc.
- New cryptosystems should come with arguments of resistance by design against this technique.

Propagation of differences

- Linear operation L

$$\delta_{out} = L(x + \delta_{in}) + L(x) = L(\delta_{in}) \quad \Longrightarrow \quad \delta_{in} \xrightarrow{L} \delta_{out} \text{ with probability 1}$$

Propagation of differences

- Linear operation L

$$\delta_{out} = L(x+\delta_{in}) + L(x) = L(\delta_{in}) \implies \delta_{in} \xrightarrow{L} \delta_{out} \text{ with probability } 1$$

- S-box $S : \mathbb{F}_2^m \to \mathbb{F}_2^m$

$$\mathtt{DDT}[\delta_{in}][\delta_{out}] = \#\{x \in \mathbb{F}_2^m : S(x) + S(x+\delta_{in}) = \delta_{out}\}.$$

	0	1	2	3	4	5	6	7
0	8	.	.	.	.	.	.	.
1	.	2	2	.	2	.	.	2
2	.	2	2	.	2	.	.	2
3	.	.	.	.	.	4	4	.
4	.	.	.	4	.	.	4	.
5	.	2	2	.	2	.	.	2
6	.	2	2	.	2	.	.	2
7	.	.	.	4	.	4	.	.

Probability of a differential transition:

$$\Pr(\delta_{in} \xrightarrow{S} \delta_{out}) = \frac{\mathtt{DDT}[\delta_{in}][\delta_{out}]}{2^m}$$

For example, $\Pr(3 \xrightarrow{S} 6) = 2^{-1}$.

Ensure resistance against differential cryptanalysis

Goal

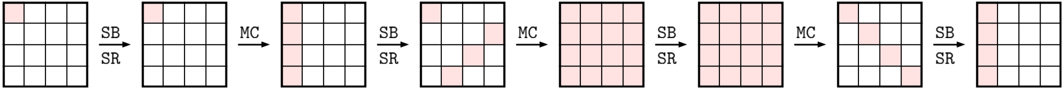
Guarantee that there is **no differential** of probability $> 2^{-n}$ after r rounds.

Ensure resistance against differential cryptanalysis

Goal

Guarantee that there is **no differential** of probability $> 2^{-n}$ after r rounds.

Work with differential characteristics instead : $\delta_{in} = \beta_0 \to \beta_1 \to \cdots \to \beta_r = \delta_{out}$.

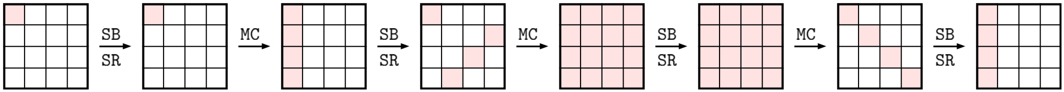


Ensure resistance against differential cryptanalysis

Easier Goal

Guarantee that there is **no differential characteristic** of probability $> 2^{-n}$ after r rounds.

Work with differential characteristics instead : $\delta_{in} = \beta_0 \to \beta_1 \to \cdots \to \beta_r = \delta_{out}$.

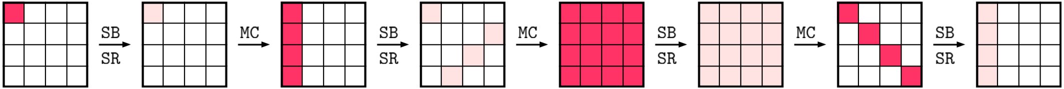


Ensure resistance against differential cryptanalysis

Easier Goal

Guarantee that there is **no differential characteristic** of probability $> 2^{-n}$ after r rounds.

Work with differential characteristics instead : $\delta_{in} = \beta_0 \to \beta_1 \to \cdots \to \beta_r = \delta_{out}$.



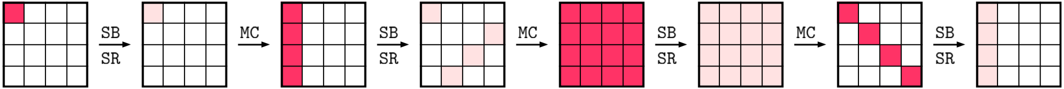
- The smallest the number of active S-boxes the highest the probability can be.

Ensure resistance against differential cryptanalysis

Easier Goal

Guarantee that there is **no differential characteristic** of probability $> 2^{-n}$ after r rounds.

Work with differential characteristics instead : $\delta_{in} = \beta_0 \to \beta_1 \to \cdots \to \beta_r = \delta_{out}$.



- The smallest the number of active S-boxes the highest the probability can be.
- **Strategy:** Guarantee that all r-round differential characteristics have a high number of active S-boxes.

Bound on the probability of a diff. characteristic

- Let $p_{max} = \max_{\delta_{in}, \delta_{out}} \Pr(\delta_{in} \xrightarrow{S} \delta_{out})$.
- Suppose there are at least n_a active S-boxes in each differential characteristic after r rounds.

Then, each differential characteristic has a probability of at most $p_{max}^{n_a}$.

Example on the `AES`: There are at least 25 active S-boxes in each 4-round differential characteristic.

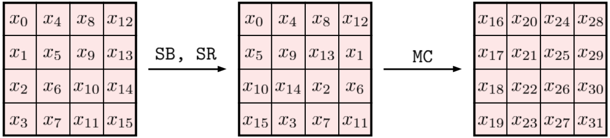
4-round differential characteristics of `AES`

Each such characteristic has a probability at most $2^{-6 \cdot 25} = 2^{-150}$,

as $p_{max} = 2^{-6}$ for the `AES` S-box.

Use MILP for the search

In 2011, Mouha et al. proposed to use a Mixed Integer Linear Programming (MILP) solver to find the minimum number of active S-boxes for the AES.



- Write the propagation constraints by using linear inequalities.

Objectif: Minimize $\sum_{i=0}^{16r-1} x_i$.

Search for distinguishers: state of the art

Today, a lot of the research on differential-like distinguishers is **generic-solver-based** (MILP, SAT, CP).

- Very useful for ciphers with weaker components (e.g. non-MDS linear layers).
- Facilitates the search for related-key characteristics.
- Possible for ciphers with a relatively-small state.

Still open problems remain:

- Weakly-aligned designs cannot be treated this way and need dedicated algorithms and tools.
- The clustering effect is difficult to evaluate.
- The Markov independency assumption does not always apply.

The key recovery problem

Main question

Once a differential distinguisher is discovered, how to use it for key recovery?

- Very technical, tedious and error-prone procedure.
- Not clear how to mount optimal attacks.

If this step is not fully understood, designers can take bad choices for their algorithm.

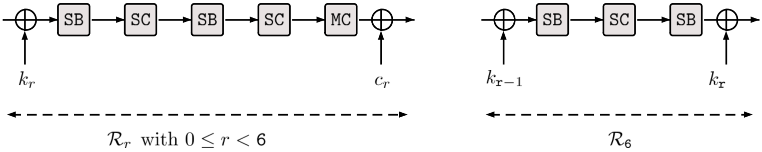
The case of the SPEEDY block cipher

The case of SPEEDY

The SPEEDY family of block ciphers was designed by Leander, Moos, Moradi and Rasoolzadeh and published at CHES 2021.

Target: ultra-low latency.

Main variant: SPEEDY-7-192



- Block size: $n = 192$ bits
- Key size: $k = 192$ bits
- 6-bit S-box, specially designed to ensure low latency.
- Linear layer of branch number 8.

Resistance of SPEEDY to differential cryptanalysis

The designers of SPEEDY presented security arguments on the resistance of the cipher to differential attacks:

- The probability of any differential characteristic over **6 rounds** is $\leq 2^{-192}$.
- Not possible to add more than one key recovery round to any differential distinguisher.

Resistance of SPEEDY to differential cryptanalysis

The designers of SPEEDY presented security arguments on the resistance of the cipher to differential attacks:

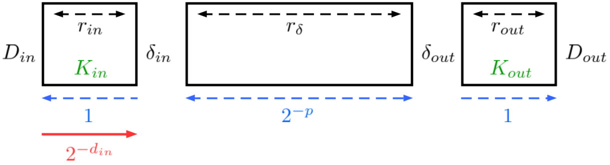
- The probability of any differential characteristic over **6 rounds** is $\leq 2^{-192}$.
- Not possible to add more than one key recovery round to any differential distinguisher. False

Joint work with N. David, R. Heim and M. Naya-Plasencia (EUROCRYPT 2023)

Full break of SPEEDY-7-192 with a differential attack.

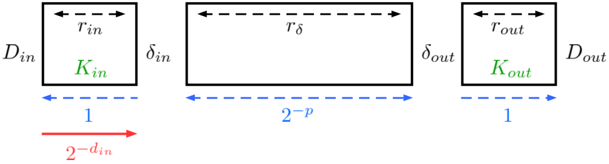
Key recovery problem

Overview of the key recovery procedure

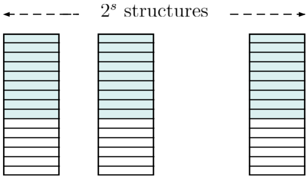


First step: Construct $2^{p+d_{in}}$ plaintext pairs (with $d_{in} = \log_2(D_{in})$).

Overview of the key recovery procedure



First step: Construct $2^{p+d_{in}}$ plaintext pairs (with $d_{in} = \log_2(D_{in})$).



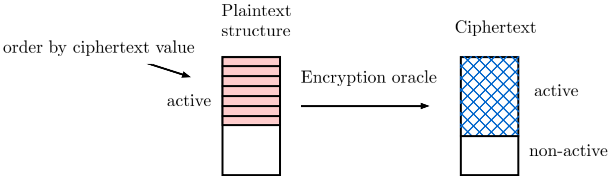
- Use 2^s plaintext structures of size $2^{d_{in}}$
  $\Longrightarrow 2^{2d_{in}-1}$ pairs from a structure.
- As $2^{s+2d_{in}-1} = 2^{p+d_{in}} \Longrightarrow s = p - d_{in} + 1$ structures.

Data complexity: 2^{p+1}, **Memory complexity:** $2^{d_{in}}$

Not all pairs are useful

Idea: Discard pairs that will **not follow** the differential.

- Keep only those plaintext pairs for which the difference of the corresponding output pairs belongs to D_{out}.
- Order the list of structures with respect to the values of the non-active bits in the ciphertext.



Number of pairs for the attack

$$N = 2^{p+d_{in}-(n-d_{out})}.$$

Goal of the key recovery

Goal

Determine the pairs for which there exists an associated key that leads to the differential.

A candidate is a triplet (P, P', k), i.e. a pair (P, P') and a (partial) key k that encrypts/decrypts the pair to the differential.

What is the complexity of this procedure?

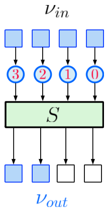
- Upper bound: $\min(2^{\kappa}, N \cdot 2^{|K_{in} \cup K_{out}|})$,
  where κ is the bit-size of the secret key.
- Lower bound: $N + N \cdot 2^{|K_{in} \cup K_{out}| - d_{in} - d_{out}}$,
  where $N \cdot 2^{|K_{in} \cup K_{out}| - d_{in} - d_{out}}$ is the number of expected candidates.

Efficient key recovery

A key recovery is efficient, if its complexity is as close as possible to the lower bound.

Solving an active S-box S in the key recovery rounds

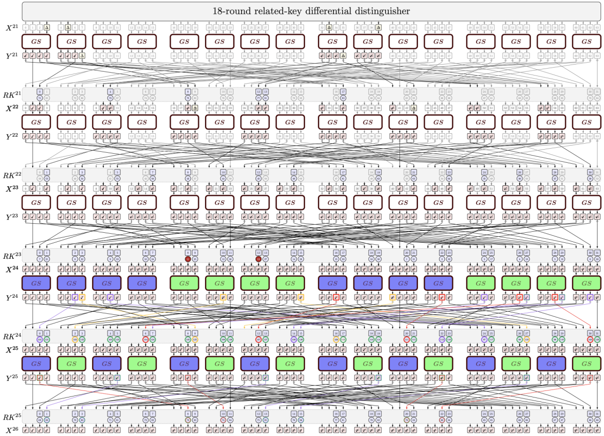
For a given pair, determine whether this pair can respect the differential constraints, and, if yes, under which conditions on the key.



A solution to S is any tuple $(x, x', S(x), S(x'))$ such that $x + x' = \nu_{in}$ and $S(x) + S(x') = \nu_{out}$.

Objectif: Reduce the earliest possible the number of pairs while maximizing the number of fixed key bits in $K_{in} \cup K_{out}$.

Why is this difficult?



Potentially too many active S-boxes and key guesses.

An algorithm for efficient key recovery

Automating the key recovery

- The key recovery for the attack on SPEEDY was very tedious and complex.
- Same issue for other differential attacks (e.g. GIFT-64, RECTANGLE).

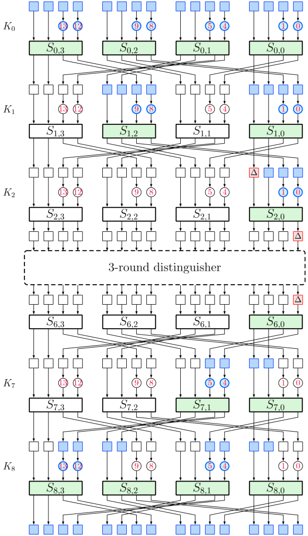
Research goal

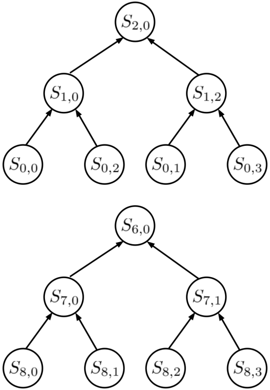
Propose an efficient algorithm together with an automated tool for this procedure.

- Hard to treat this problem for all kind of block cipher designs.
- A first target: SPN ciphers with a bit-permutation layer and an (almost) linear key schedule.

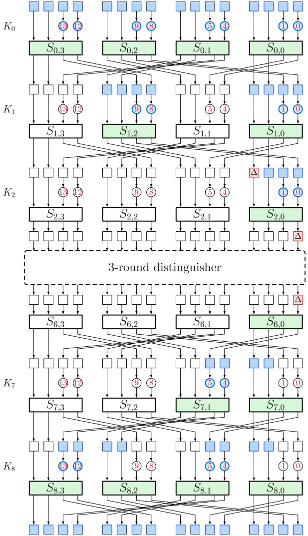
Joint work with David, Derbez, Heim and Naya-Plasencia (under submission).

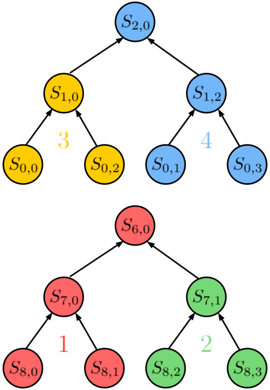
Modeling the key recovery as a graph





Modeling the key recovery as a graph





Order is important!

Algorithm - high level description

First step: Add the key recovery rounds, detect the active S-boxes and build the graph.

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that allows to enumerate all the possible values that the S-boxes of X can take under the differential constraints imposed by the distinguisher.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions
- online time complexity

A strategy can be further refined with extra information: e.g. memory, offline time.

Compare two strategies

Objectif: Build an efficient strategy for the whole graph.

- Based on basic strategies, i.e. strategies for a single S-box.

Output of the tool

An efficient order to combine all basic subgraphs, aiming to minimize the complexity of the resulting strategy.

Compare two strategies $\mathscr{S}_X^1$ and $\mathscr{S}_X^2$ for the same subgraph X

1. Choose the one with the best time complexity.
2. If same time complexity, choose the one with the best memory complexity.

Merging two strategies

Let $\mathscr{S}_X$ and $\mathscr{S}_Y$ two strategies for the graphs X and Y respectively.

- The number of solutions of $\mathscr{S}(X \cup Y)$ **only depends** on $X \cup Y$:

Number of solutions of $\mathscr{S}_{X \cup Y}$

$Sol(X \cup Y) = Sol(X) + Sol(Y) - \#$ bit-relations between the nodes of X and Y

Time and memory associated to $\mathscr{S}_{X \cup Y}$

- $T(\mathscr{S}_{X \cup Y}) \approx \max(T(\mathscr{S}_X), T(\mathscr{S}_Y), Sol(\mathscr{S}_{X \cup Y}))$
- $M(\mathscr{S}_{X \cup Y}) \approx \max(M(\mathscr{S}_X), M(\mathscr{S}_Y), \min(Sol(\mathscr{S}_X), Sol(\mathscr{S}_Y)))$

A dynamic programming approach

- The online time complexity of $\mathscr{S}_{X \cup Y}$ **only depends** on the time complexities of $\mathscr{S}_X$ and $\mathscr{S}_Y$.
- An optimal strategy for $X \cup Y$ **can always** be obtained by merging two optimal strategies for X and Y.
- Use a bottom-up approach, merging first the strategies with the smallest time complexity to reach a graph strategy with a minimal time complexity.

Dynamic programming approach

Ensure that, for any subgraph X, we only keep one optimal strategy to enumerate it.

Pre-sieving

Idea behind the pre-sieving

Reduce the number of pairs as quickly as possible to only keep the $N' \leq N$ pairs that satisfy the differential constraints.

How: Use the differential constraints of the S-boxes of the external rounds.

Advantage

The key recovery is performed on less pairs.

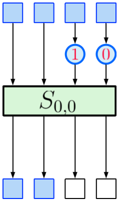
Pre-sieving in practice

Offline step: Per active S-box, build a sieving list L with the solutions to the S-box:

- Bits **without** key addition: store the pair.
- Bits **with** key addition: store the difference.

Online step: For each pair and each S-box, check whether the pair is consistent with the sieving list.

Filter: $\dfrac{|L|}{2^s}$, where s is the size of the tuples in L.



$$(\mathtt{x}_3, \mathtt{x}'_3, \mathtt{x}_2, \mathtt{x}'_2, \mathtt{x}_1 \oplus \mathtt{x}'_1, \mathtt{x}_0 \oplus \mathtt{x}'_0)$$

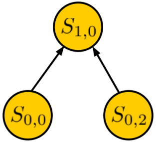
Filter: $\dfrac{36}{2^6} = 2^{-0.83}$.

After this step: $N' = 2^{-5.63} N$.

Precomputing partial solutions

Idea

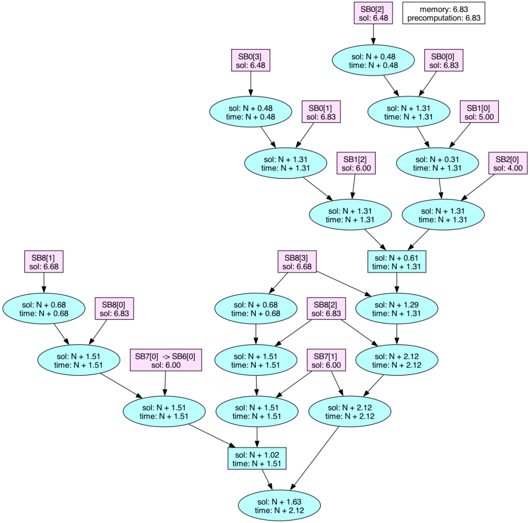
Precompute the partial solutions to some subgraph.



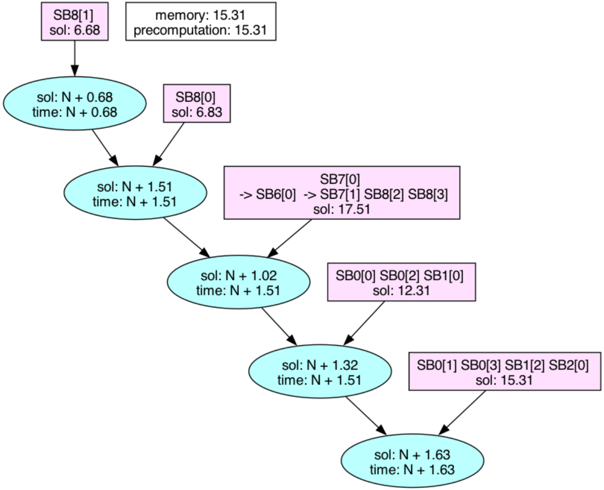
- Impact on the memory complexity and the offline time of the attack.
- The optimal key recovery strategy depends on how much memory and offline time are allowed.

Applications

Application to the toy cipher (1)



Application to the toy cipher (2)



Application to RECTANGLE

RECTANGLE is a block cipher designed by Zhang, Bao, Lin, Rijmen, Yang and Verbauwhede in 2015.

- The designers proposed a differential attack on **18 rounds** of RECTANGLE-80 and RECTANGLE-128.
- Broll et al. improved the time complexity of this attack with advanced techniques.

The tool found an optimal attack on **19 rounds** of RECTANGLE-128 without any extra effort.

Application to other ciphers

Start from an existing distinguisher that led to the best key recovery attack against the target cipher.

- `PRESENT-80`: Extended by two rounds the previous best differential attack.
- `GIFT-64` and `SPEEDY-7-192`: Best key recovery strategy without additional techniques.

Extensions and improvements

- Handle ciphers with more complex linear layers.
- Handle ciphers with non-linear key schedules.
- Incorporate tree-based key recovery techniques by exploiting the structure of the involved S-boxes.

The best distinguisher **does not always** lead to the best key recovery!

Ultimate goal

Combine the tool with a distinguisher-search algorithm to find the best possible attacks.

Other open problems

- Prove optimality.
- Apply a similar approach to other attacks.

Other open problems

- Prove optimality.
- Apply a similar approach to other attacks.

Thanks for your attention!