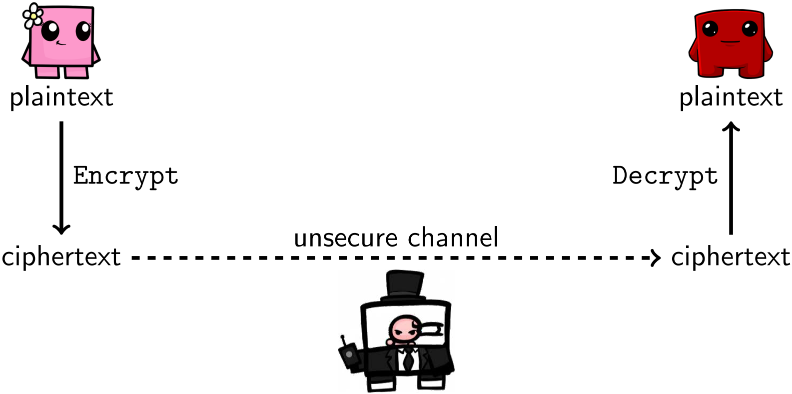
Making (near) Optimal Choices for the Design of Block Ciphers

Baptiste Lambin

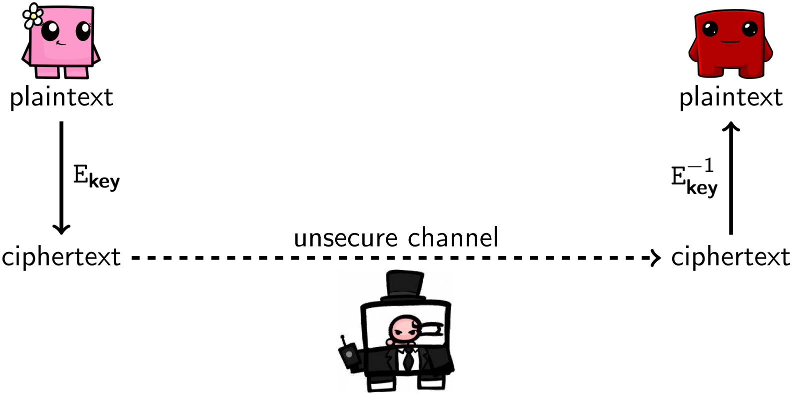
Horst Görtz Institute for IT Security, Ruhr University Bochum

27/01/2020

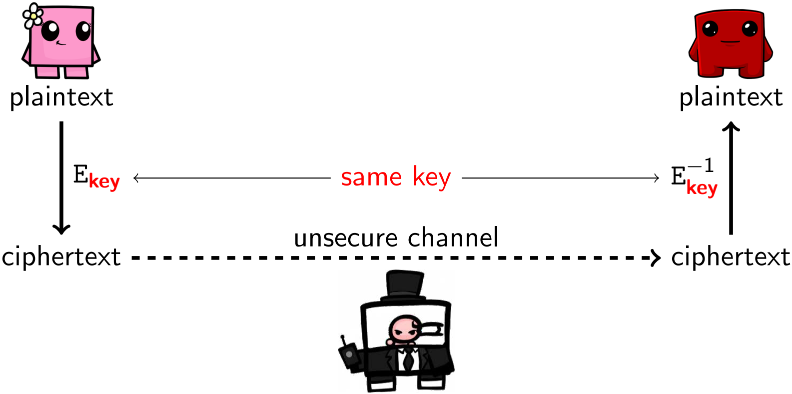
Cryptography and Encryption



Symmetric Encryption



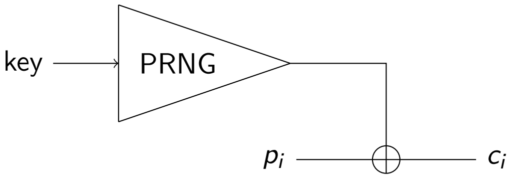
Symmetric Encryption



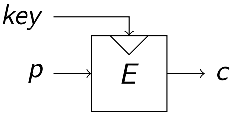
Block and Stream Ciphers

Two main ways to build symmetric encryption :

- Stream Ciphers :



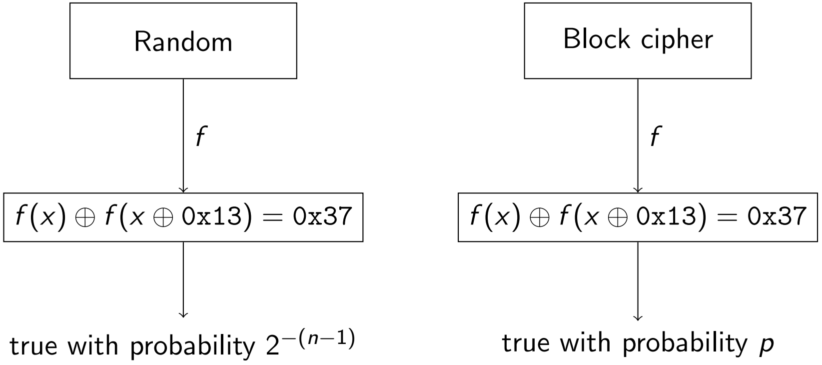
- Block ciphers :



E_{key} is a permutation of fixed size (n bits)

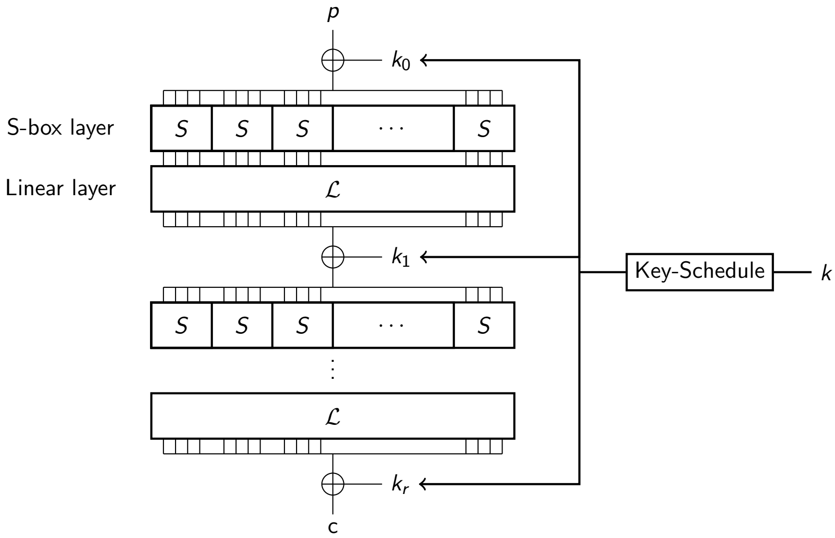
Distinguishers

$\Rightarrow$ Behavior of the block cipher that a random function does not have.

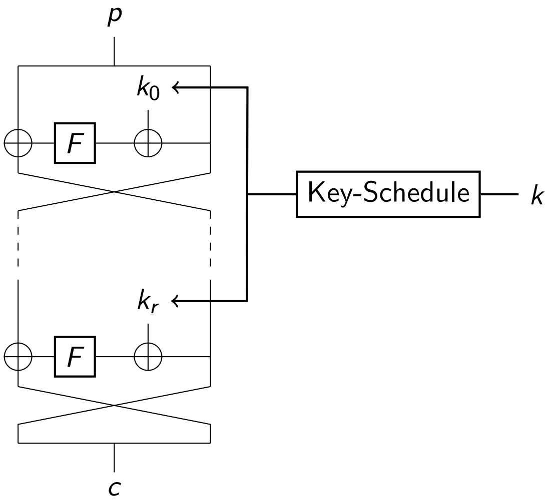


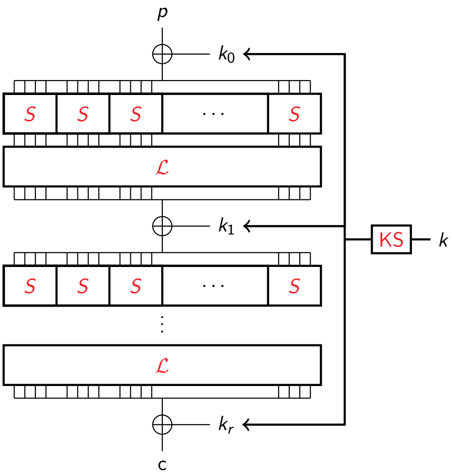
$p \gg 2^{-(n-1)} \Rightarrow$ we have a distinguisher

Substitution-Permutation Networks

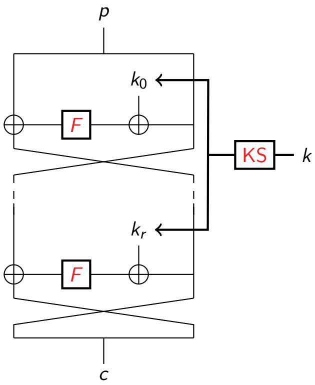


Feistel Networks





p
k_0
S S S · · · S
$\mathcal{L}$
k_1
KS
k
S S S · · · S
$\mathcal{L}$
k_r
c



p
k_0
F
KS
k
k_r
F
c

Finding Optimal Components

Naïve algorithm : exhaustive search

Finding Optimal Components

Naïve algorithm : exhaustive search

Pros :

- (Relatively) easy to implement
- Optimality is easy to prove

Finding Optimal Components

Naïve algorithm : exhaustive search

Pros :

- (Relatively) easy to implement
- Optimality is easy to prove

Cons (non-exclusive) :

- The search space can be very large
  e.g. From 2^{52} up to 2^{75} in the first part of this presentation
- Testing one candidate can be expensive
  e.g. In the second part of this presentation, "only" 2^{44} candidates but testing each of them is expensive

Tools for Optimization

- (Mixed) Integer Linear Programming (and some other variants)
- Constraint Programming
- Metaheuristics (near optimality)
- SAT (somewhat)
- Dedicated algorithms

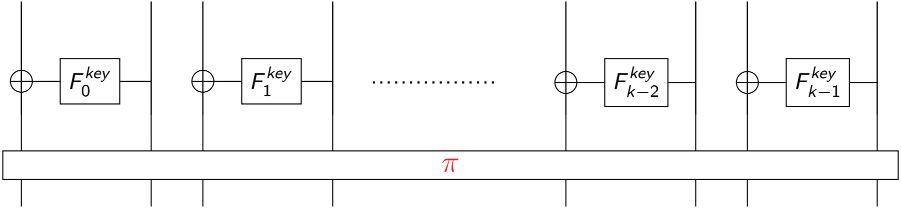
Tools for Optimization

- (Mixed) Integer Linear Programming (and some other variants)
- Constraint Programming
- Metaheuristics (near optimality)
- SAT (somewhat)
- Dedicated algorithms

In this talk :

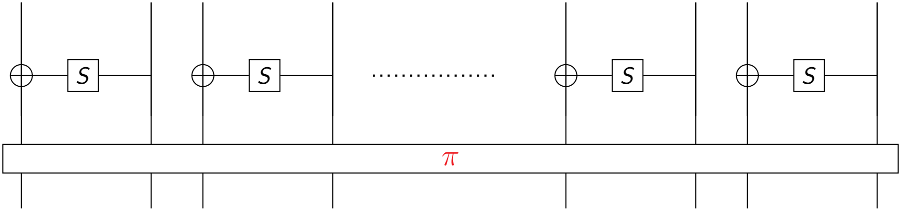
- Part 1 : Dedicated algorithm ($\sim$ Branch-and-Bound) + efficient testing for the small cases
- Part 2 : Metaheuristics + Constraint Programming

Generalized Feistel Network



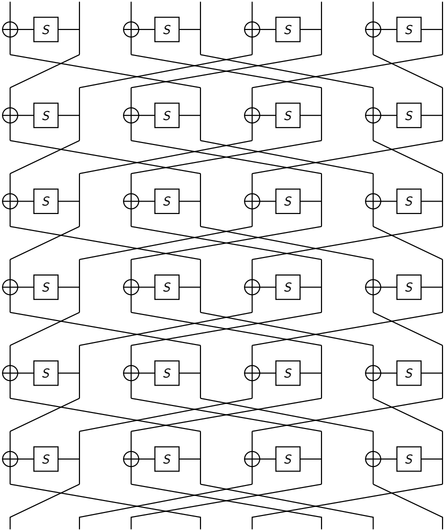
- State composed of $2k$ blocks
- k Feistels in parallel followed by a permutation π
- Easier to design but slower diffusion

Generalized Feistel Network

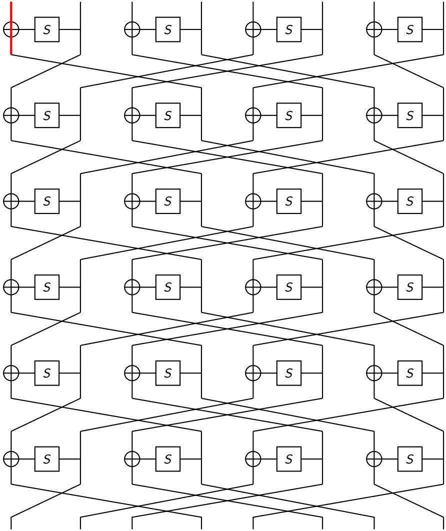


- State composed of $2k$ blocks
- k Feistels in parallel followed by a permutation π
- Easier to design but slower diffusion
- In this work, the key and the definition of the F-functions don't matter

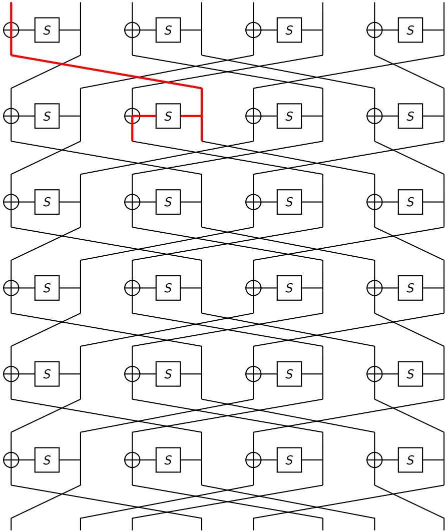
Diffusion Round



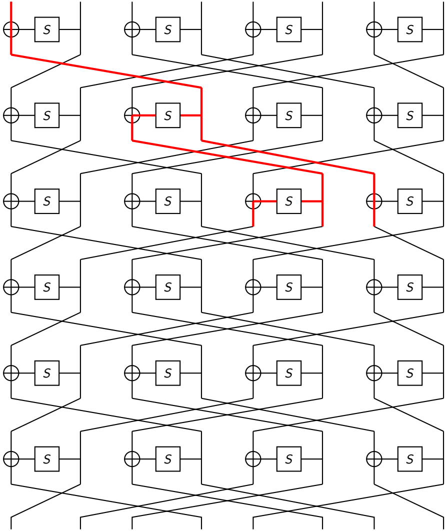
Diffusion Round



Diffusion Round

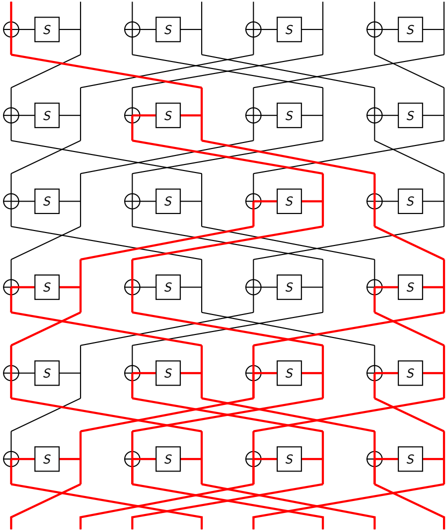


Diffusion Round

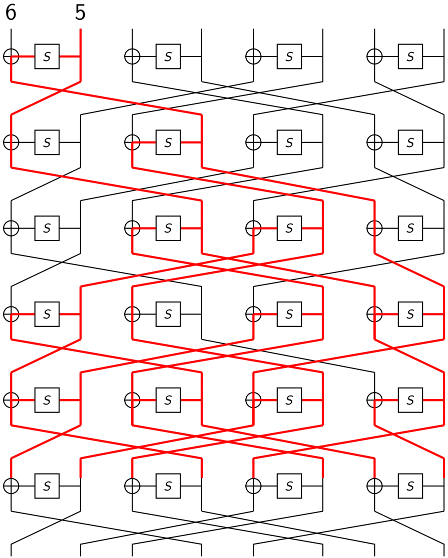


Diffusion Round

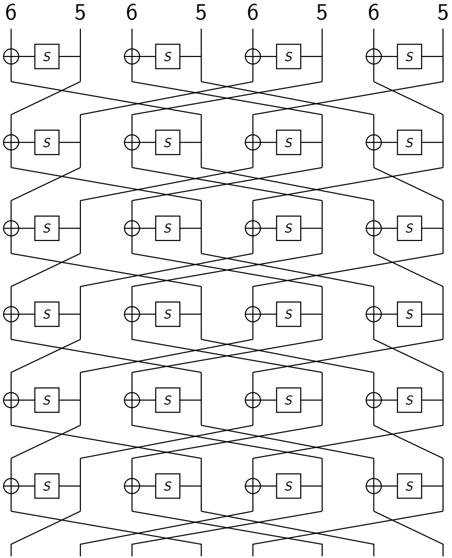
6



Diffusion Round

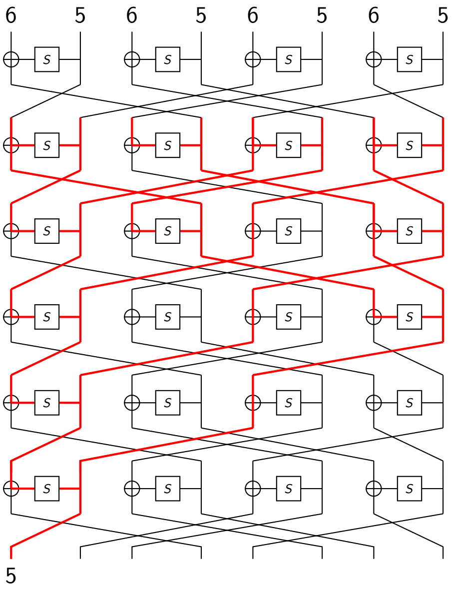


Diffusion Round



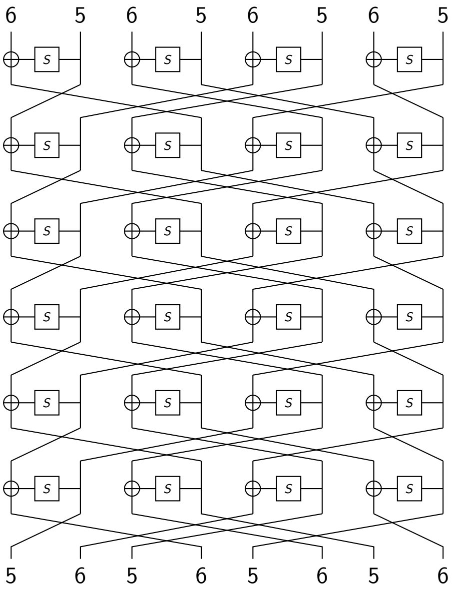
- Depends only on π
- Tied to impossible differential and integral attacks
- For encryption...

Diffusion Round



- Depends only on π
- Tied to impossible differential and integral attacks
- For encryption and decryption

Diffusion Round



- Depends only on π
- Tied to impossible differential and integral attacks
- For encryption and decryption
- $\mathrm{DR}(\pi) = 6$ here

Previous Work

- Suzaki and Minematsu at FSE'10
  - Lower bound on $\mathrm{DR}(\pi)$ depending only on k
  - Exhaustive search for $2k \leq 16$
  - Observed that all optimal permutations in these cases are *even-odd*
  - Generic construction with $\mathrm{DR}(\pi) = 2\log_2 k$ (not optimal in general)

Previous Work

- Suzaki and Minematsu at FSE'10
  - Lower bound on $\mathrm{DR}(\pi)$ depending only on k
  - Exhaustive search for $2k \leq 16$
  - Observed that all optimal permutations in these cases are *even-odd*
  - Generic construction with $\mathrm{DR}(\pi) = 2\log_2 k$ (not optimal in general)
- Cauchois *et al.* at FSE'19
  - Equivalence relation for *even-odd* permutations
  - Optimal *even-odd* permutations for $18 \leq 2k \leq 26$
  - Good candidate for $2k = 32$ (already known from FSE'10) and $2k = 64, 128$

Previous Work

- Suzaki and Minematsu at FSE'10
  - Lower bound on $\mathrm{DR}(\pi)$ depending only on k
  - Exhaustive search for $2k \leq 16$
  - Observed that all optimal permutations in these cases are *even-odd*
  - Generic construction with $\mathrm{DR}(\pi) = 2\log_2 k$ (not optimal in general)
- Cauchois *et al.* at FSE'19
  - Equivalence relation for *even-odd* permutations
  - Optimal *even-odd* permutations for $18 \leq 2k \leq 26$
  - Good candidate for $2k = 32$ (already known from FSE'10) and $2k = 64, 128$

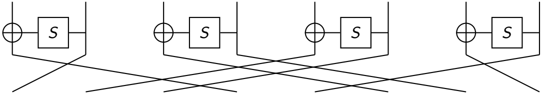
Open problem : is the permutation on 32 blocks optimal ?

Diffusion round of 10 but lower bound at 9 rounds.

This Work

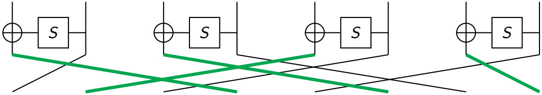
- We solve this 10-year-old problem
- New characterization for the diffusion round
  $\Rightarrow$ Efficient algorithm to search for an optimal permutation
- Results for $28 \leq 2k \leq 42$
- Security evaluation for all permutations found

Even-odd Permutations



$$\pi = (3, 0, 5, 6, 1, 2, 7, 4)$$

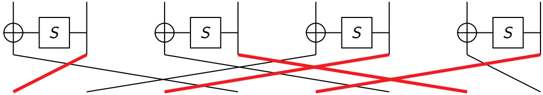
Even-odd Permutations



$$\pi = (3, 0, 5, 6, 1, 2, 7, 4)$$

$$p = (1, 2, 0, 3) \qquad \pi(2i) = 2p(i) + 1$$

Even-odd Permutations

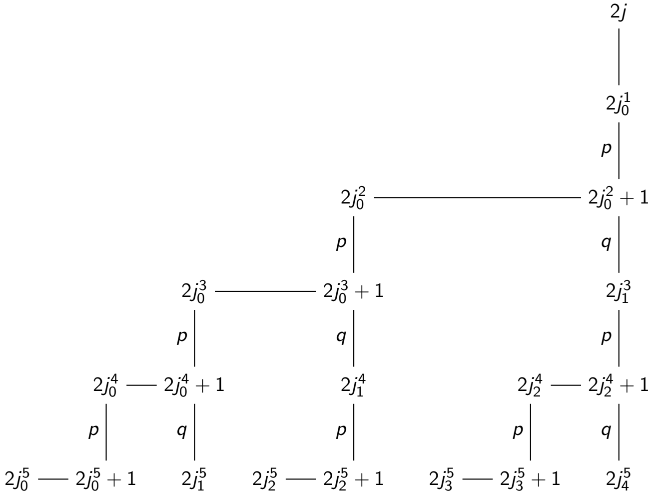


$$\pi = (3, 0, 5, 6, 1, 2, 7, 4)$$

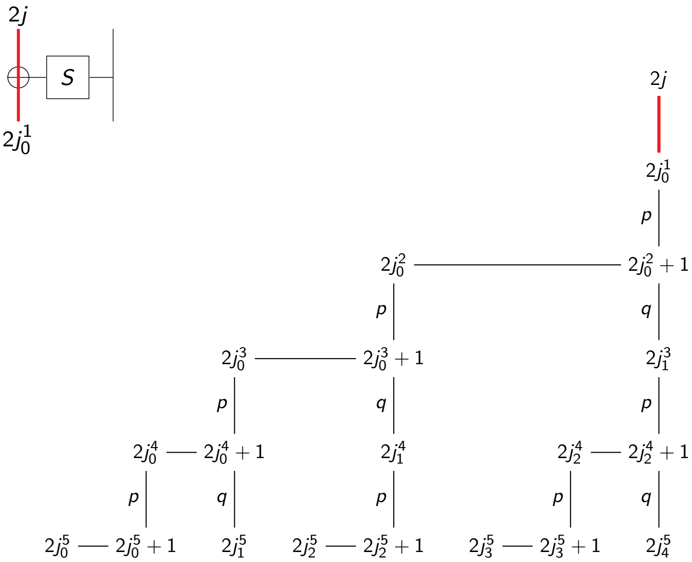
$$p = (1, 2, 0, 3) \qquad \pi(2i) = 2p(i) + 1$$

$$q = (0, 3, 1, 2) \qquad \pi(2i+1) = 2q(i)$$

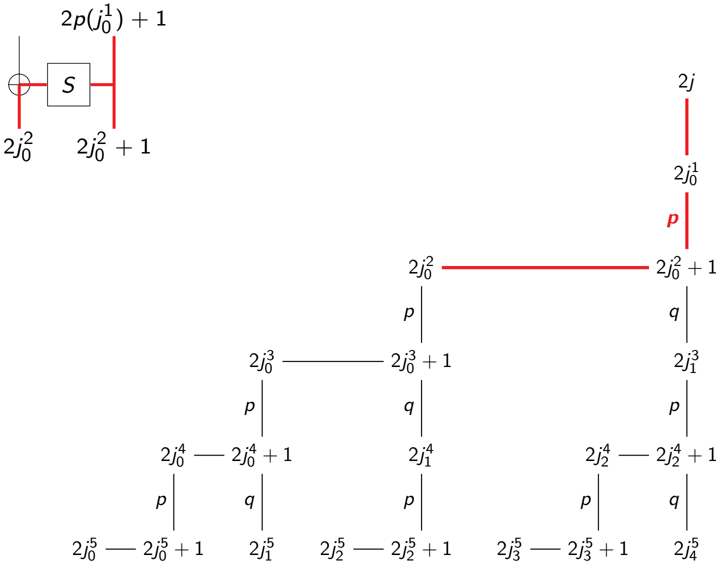
Ideal Diffusion



Ideal Diffusion

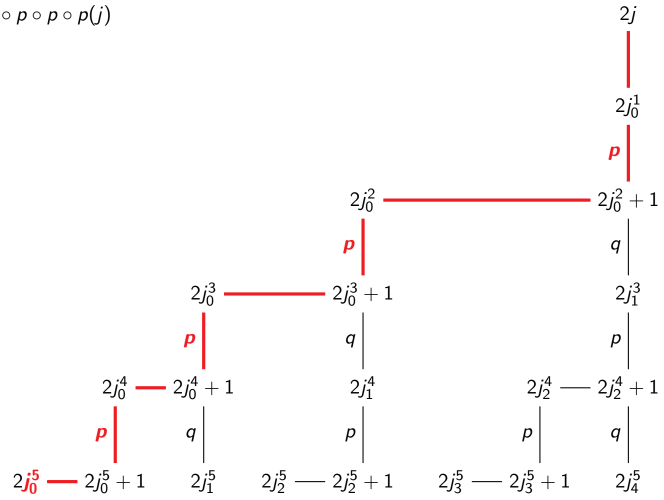


Ideal Diffusion



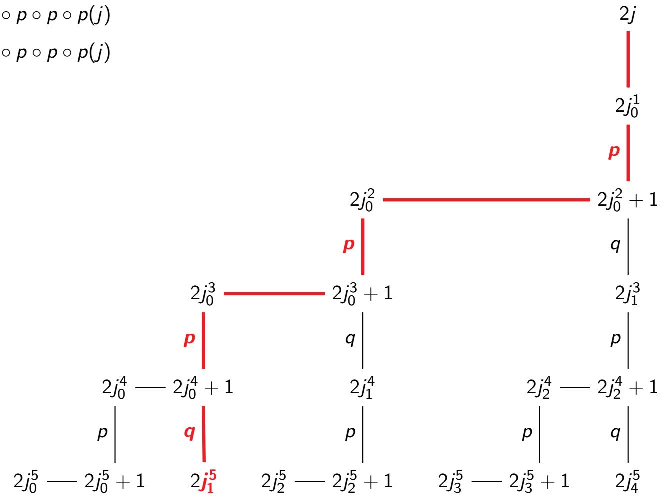
Ideal Diffusion

$j_0^5 = p \circ p \circ p \circ p(j)$



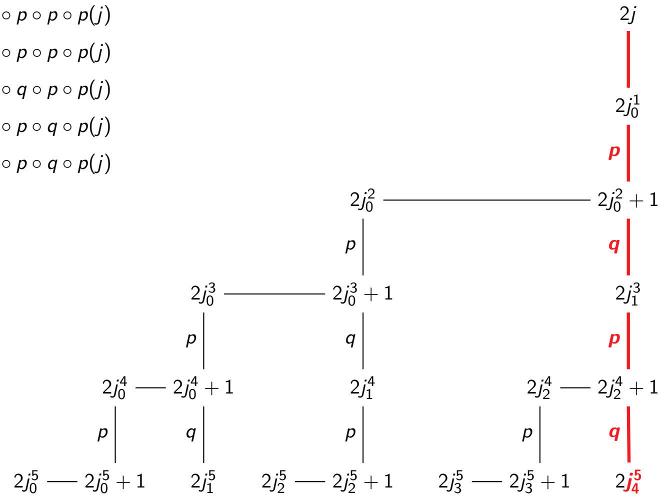
Ideal Diffusion

$$j_0^5 = p \circ p \circ p \circ p(j)$$
$$j_1^5 = q \circ p \circ p \circ p(j)$$



Ideal Diffusion

$$
\boxed{\mathbb{J}_j^5}
\begin{cases}
j_0^5 = p \circ p \circ p \circ p(j) \\
j_1^5 = q \circ p \circ p \circ p(j) \\
j_2^5 = p \circ q \circ p \circ p(j) \\
j_3^5 = p \circ p \circ q \circ p(j) \\
j_4^5 = q \circ p \circ q \circ p(j)
\end{cases}
$$



A Visualization of This Characterization

$\mathbb{J}_j^7$

x	0	1	2	3	4	5	6	7
p^5	3	4	5	6	7	0	1	2
p^4q	4	5	6	7	0	1	2	3
p^3qp	4	5	6	7	0	1	2	3
p^2qp^2	4	5	6	7	0	1	2	3
pqp^3	4	5	6	7	0	1	2	3
qp^4	4	5	6	7	0	1	2	3
p^2qpq	5	6	7	0	1	2	3	4
pqp^2q	5	6	7	0	1	2	3	4
qp^3q	5	6	7	0	1	2	3	4
$pqpqp$	5	6	7	0	1	2	3	4
qp^2qp	5	6	7	0	1	2	3	4
$qpqp^2$	5	6	7	0	1	2	3	4
$qpqpq$	6	7	0	1	2	3	4	5
diff	4	4	4	4	4	4	4	4

Cyclic Shift

$p = (7, 0, 1, 2, 3, 4, 5, 6)$

$q = (0, 1, 2, 3, 4, 5, 6, 7)$

A Visualization of This Characterization

x	**0**	1	2	3	4	5	6	7
p^5	3	4	5	6	7	0	1	2
p^4q	4	5	6	7	0	1	2	3
p^3qp	4	5	6	7	0	1	2	3
p^2qp^2	4	5	6	7	0	1	2	3
pqp^3	4	5	6	7	0	1	2	3
qp^4	4	5	6	7	0	1	2	3
p^2qpq	5	6	7	0	1	2	3	4
pqp^2q	5	6	7	0	1	2	3	4
qp^3q	5	6	7	0	1	2	3	4
$pqpqp$	5	6	7	0	1	2	3	4
qp^2qp	5	6	7	0	1	2	3	4
$qpqp^2$	5	6	7	0	1	2	3	4
$qpqpq$	6	7	0	1	2	3	4	5
diff	**4**	4	4	4	4	4	4	4

$\mathbb{J}_{\mathbf{0}}^{7}$ (rows p^5 through $qpqpq$)

Cyclic Shift

$p = (7, 0, 1, 2, 3, 4, 5, 6)$

$q = (0, 1, 2, 3, 4, 5, 6, 7)$

A Visualization of This Characterization

	x	0	1	2	3	4	5	6	7
$\mathbb{J}_1^7$	p^5	3	4	5	6	7	0	1	2
	p^4q	4	5	6	7	0	1	2	3
	p^3qp	4	5	6	7	0	1	2	3
	p^2qp^2	4	5	6	7	0	1	2	3
	pqp^3	4	5	6	7	0	1	2	3
	qp^4	4	5	6	7	0	1	2	3
	p^2qpq	5	6	7	0	1	2	3	4
	pqp^2q	5	6	7	0	1	2	3	4
	qp^3q	5	6	7	0	1	2	3	4
	$pqpqp$	5	6	7	0	1	2	3	4
	qp^2qp	5	6	7	0	1	2	3	4
	$qpqp^2$	5	6	7	0	1	2	3	4
	$qpqpq$	6	7	0	1	2	3	4	5
	diff	4	**4**	4	4	4	4	4	4

Cyclic Shift

$p = (7, 0, 1, 2, 3, 4, 5, 6)$

$q = (0, 1, 2, 3, 4, 5, 6, 7)$

A Visualization of This Characterization

x	0	1	2	3	4	5	6	7
p^5	4	3	5	1	6	7	0	2
p^4q	3	2	1	4	0	6	7	5
p^3qp	2	6	7	5	1	3	4	0
p^2qp^2	6	7	4	0	5	2	3	1
pqp^3	1	4	3	2	0	6	7	5
qp^4	2	5	7	6	3	1	4	0
p^2qpq	7	1	0	6	3	5	2	4
pqp^2q	4	5	2	1	7	0	6	3
qp^3q	5	0	6	2	4	3	1	7
$pqpqp$	5	0	6	3	2	4	1	7
qp^2qp	0	3	1	7	6	5	2	4
$qpqp^2$	3	1	2	4	7	0	5	6
$qpqpq$	1	6	4	3	5	7	0	2
diff	8	8	8	8	8	8	8	8

$\mathbb{J}_j^7$

Optimal Permutation

$p = (6, 3, 7, 1, 0, 2, 4, 5)$

$q = (3, 5, 1, 6, 4, 0, 2, 7)$

A Visualization of This Characterization

x	**0**	1	2	3	4	5	6	7
p^5	4	3	5	1	6	7	0	2
p^4q	3	2	1	4	0	6	7	5
p^3qp	2	6	7	5	1	3	4	0
p^2qp^2	6	7	4	0	5	2	3	1
pqp^3	1	4	3	2	0	6	7	5
qp^4	2	5	7	6	3	1	4	0
p^2qpq	7	1	0	6	3	5	2	4
pqp^2q	4	5	2	1	7	0	6	3
qp^3q	5	0	6	2	4	3	1	7
$pqpqp$	5	0	6	3	2	4	1	7
qp^2qp	0	3	1	7	6	5	2	4
$qpqp^2$	3	1	2	4	7	0	5	6
$qpqpq$	1	6	4	3	5	7	0	2
diff	**8**	8	8	8	8	8	8	8

$\mathbb{J}_{\mathbf{0}}^{7}$ (rows p^5 through $qpqpq$)

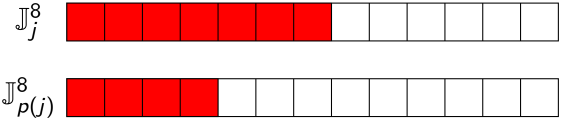
Optimal Permutation

$p = (6, 3, 7, 1, 0, 2, 4, 5)$

$q = (3, 5, 1, 6, 4, 0, 2, 7)$

Searching for an optimal permutation

- $(k!)^2$ even-odd permutations, reduced to $\mathcal{N}_k.k!$ with an equivalence relation.
  $\mathcal{N}_k :=$ number of partitions of the integer k.
  $\Rightarrow$ For $2k = 32$, $\sim 2^{52}$ permutations instead of $(16!)^2 \simeq 2^{88}$.
- Main idea : partially compute some $\mathbb{J}_j^r$ + Branch-and-Bound



Searching for Optimal Permutations : Observations 1

- Can be efficiently implemented with table lookups
  $\Rightarrow$ Very efficient exhaustive search for $2k \leq 26$ (but already known)
- Focus on $28 \leq 2k \leq 42$, lower bound for full diffusion at 9 rounds

Searching for Optimal Permutations : Observations 1

- Can be efficiently implemented with table lookups
  $\Rightarrow$ Very efficient exhaustive search for $2k \leq 26$ (but already known)
- Focus on $28 \leq 2k \leq 42$, lower bound for full diffusion at 9 rounds
- Main idea : fix p with a given cycle structure and search q
- Need to consider $\mathbb{J}_j^8$, but computing $\mathbb{J}_j^8$ requires to known (most of) q

Searching for Optimal Permutations : Observations 1

- Can be efficiently implemented with table lookups
  $\Rightarrow$ Very efficient exhaustive search for $2k \leq 26$ (but already known)
- Focus on $28 \leq 2k \leq 42$, lower bound for full diffusion at 9 rounds
- Main idea : fix p with a given cycle structure and search q
- Need to consider $\mathbb{J}_j^8$, but computing $\mathbb{J}_j^8$ requires to known (most of) q
- But !
  - Computing $\mathbb{J}_j^i$ requires to compute $\mathbb{J}_j^{i'}$ for $i' < i$
  - Some computations for $\mathbb{J}_j^i$ and $\mathbb{J}_{j'}^i$, $j \neq j'$, can be the same

Searching for Optimal Permutations : Observations 2

- Knowing p, computing $\mathbb{J}_j^6$ requires to make 7 guesses on q
- Computing $\mathbb{J}_{p(j)}^6$ requires (at most) only 3 additional guesses on q

Searching for Optimal Permutations : Observations 2

- Knowing p, computing $\mathbb{J}_j^6$ requires to make 7 guesses on q
- Computing $\mathbb{J}_{p(j)}^6$ requires (at most) only 3 additional guesses on q
- $\mathbb{J}_j^6$ can be written as $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ with $\mathbb{X}_j^6 \cap \mathbb{Y}_j^6 = \emptyset$ such that

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$$

Make the 7 seven guesses on q to compute $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ so that

$$\mathbb{J}_j^8 \quad = \quad p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \quad \cup \quad (pq)(\mathbb{X}_j^6) \quad \cup \quad (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$$

Make the 7 seven guesses on q to compute $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ so that

$$\begin{aligned}
\mathbb{J}_j^8 &= \underbrace{p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)}_{\text{known}} \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \\
&= \mathbb{K}_j
\end{aligned}$$

Make the 7 seven guesses on q to compute $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ so that

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$$

known

known

$$= \mathbb{K}_j$$

Make the 7 seven guesses on q to compute $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ so that

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$$

known

known

$$= \mathbb{K}_j \cup (\mathrm{pq})(\widetilde{\mathbb{X}}_j^6)$$

- $\widetilde{\mathbb{X}}_j^6 \subset \mathbb{X}_j^6$ s.t. $\forall x \in \widetilde{\mathbb{X}}_j^6$ $q(x)$ is unknown

Make the 7 seven guesses on q to compute $\mathbb{J}_j^6 = \mathbb{X}_j^6 \cup \mathbb{Y}_j^6$ so that

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$$

known

known

known

$$= \mathbb{K}_j \cup (\mathrm{pq})(\widetilde{\mathbb{X}}_j^6) \cup q(\widetilde{\mathbb{Y}}_j^6)$$

- $\widetilde{\mathbb{X}}_j^6 \subset \mathbb{X}_j^6$ s.t. $\forall x \in \widetilde{\mathbb{X}}_j^6$ $q(x)$ is unknown
- $\widetilde{\mathbb{Y}}_j^6 \subset p(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$, s.t. $\forall x \in \widetilde{\mathbb{Y}}_j^6$ $q(x)$ is unknown

Make the 7 guesses on q to compute $\mathbb{J}_j^6$, thus

$$\mathbb{J}_j^8 = \mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)$$

Make the 7 guesses on q to compute $\mathbb{J}_j^6$, thus

$$\mathbb{J}_j^8 = \mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)$$

Full diffusion for j means that we have the constraint

$$C_j : \left|\mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)\right| \geq k$$

Make the 7 guesses on q to compute $\mathbb{J}_j^6$, thus

$$\mathbb{J}_j^8 = \mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)$$

Full diffusion for j means that we have the constraint

$$C_j : \left|\mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)\right| \geq k$$

Make 3 additional guesses on q, update and check[1] C_j, and then we get

$$C_{j'} : \left|\mathbb{K}_{j'} \cup q(\widetilde{\mathbb{Y}}_{j'}^6) \cup (pq)(\widetilde{\mathbb{X}}_{j'}^6)\right| \geq k, \quad j' = p(j)$$

[1] Use voodoo magic to check if a constraint C_j can be satisfied, see paper

Make the 7 guesses on q to compute $\mathbb{J}_j^6$, thus

$$\mathbb{J}_j^8 = \mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)$$

Full diffusion for j means that we have the constraint

$$C_j : \left|\mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)\right| \geq k$$

Make 3 additional guesses on q, update and check[1] C_j, and then we get

$$C_{j'} : \left|\mathbb{K}_{j'} \cup q(\widetilde{\mathbb{Y}}_{j'}^6) \cup (pq)(\widetilde{\mathbb{X}}_{j'}^6)\right| \geq k, \quad j' = p(j)$$

Keep going until q is fully defined (or constraints can never be all satisfied)

[1] Use voodoo magic to check if a constraint C_j can be satisfied, see paper

Results and Summary

- New characterization for the diffusion round in a GFN
- Very efficient search algorithm, highly parallelizable ($< 1h$ for each case with 72 threads)

Results and Summary

- New characterization for the diffusion round in a GFN
- Very efficient search algorithm, highly parallelizable ($< 1h$ for each case with 72 threads)
- For $2k = 28, 30, 32$ and 36, the optimal number of rounds for full diffusion is 9.

Results and Summary

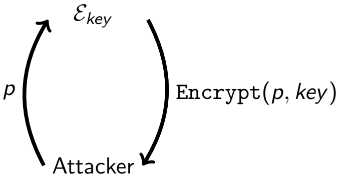
- New characterization for the diffusion round in a GFN
- Very efficient search algorithm, highly parallelizable ($< 1h$ for each case with 72 threads)
- For $2k = 28, 30, 32$ and 36, the optimal number of rounds for full diffusion is 9.
- For $2k = 34$, the optimal number of rounds for full diffusion is 10.

Results and Summary

- New characterization for the diffusion round in a GFN
- Very efficient search algorithm, highly parallelizable ($< 1h$ for each case with 72 threads)
- For $2k = 28, 30, 32$ and 36, the optimal number of rounds for full diffusion is 9.
- For $2k = 34$, the optimal number of rounds for full diffusion is 10.
- For $2k = 38, 40$ and 42, the optimal number of rounds for full diffusion is at least 10 and at most 11.

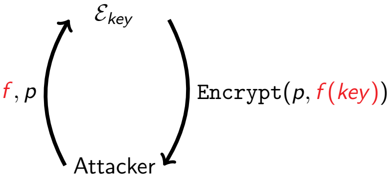
1. Introduction
2. Efficient Search for Optimal Diffusion Layers of GFNs
3. Variants of the AES Key-Schedule for Better Truncated Differential Bounds
4. Perspectives

Security model



<u>Standard model</u>

Can only ask the encryption of some plaintexts p.



<u>Related-key model</u>

Can ask the encryption of some plaintexts p with a modified key.

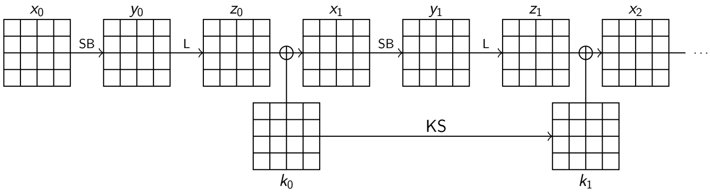
(Related-key) Differentials attacks

Given an n-bit block cipher E, can we find a tuple $(\Delta_{in}, \Delta_{out}, \Delta_k) \in \mathbb{F}_2^{3n}$ such that for any message p,

$$E(p \oplus \Delta_{in}, k \oplus \Delta_k) = E(p, k) \oplus \Delta_{out}$$

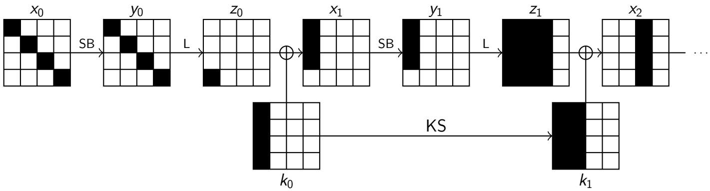
holds independently from the value of the key with high probability ?

AES



- 128-bit block cipher, $\{128, 192, 256\}$-bit key
- Round function :
  - SubBytes (SB,non-linear)
  - $\mathrm{L} = \mathrm{MixColumns} \circ \mathrm{ShiftRows}$ (linear)
  - AddRoundKey ($\oplus$)
- Round keys are derived from the master key using a key schedule KS (non-linear)

Truncated differential characteristic

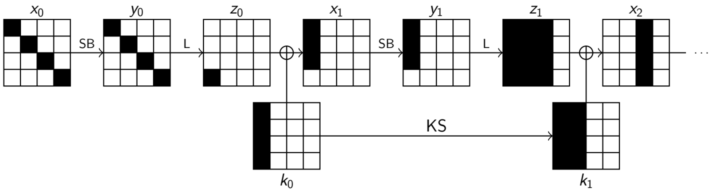


Only consider whether a difference is zero or not (active byte).

$\Rightarrow$ Easier to search than regular differentials

$\Rightarrow$ Can still give some security results for differential attacks

Truncated differential characteristic



Only consider whether a difference is zero or not (active byte).

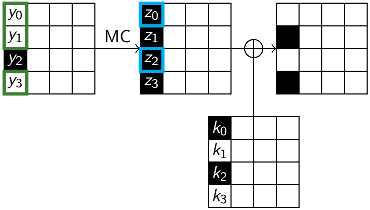
⇒ Easier to search than regular differentials

⇒ Can still give some security results for differential attacks

May be impossible to instantiate with regular differentials

⇒ We can consider some additional information to avoid this ! (Induced equations !)

Equations induced by MixColumns (MDS property)

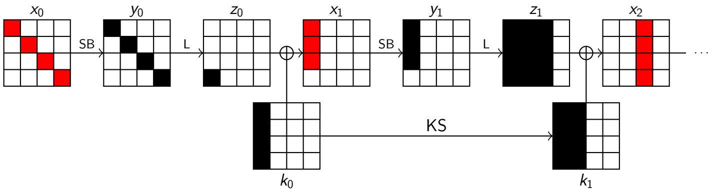


Let $z = \mathrm{MC}(y)$ with $y, z \in \left(\mathbb{F}_2^8\right)^4$. Then there is a linear equation between any 5 bytes in y and z.

$$5.y_0 \oplus 7.y_1 \oplus y_3 = 2.z_0 \oplus z_2$$

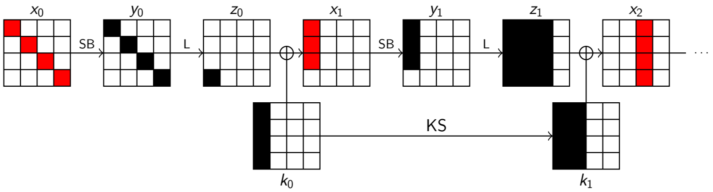
But y_0, y_1 and y_3 are zero differences, and (z_0, z_2) is cancelled by (k_0, k_2). Hence $2.k_0 \oplus k_2 = 0$.

Active S-Boxes



Number of active S-boxes $\Rightarrow$ maximal probability of the (truncated) differential characteristic.

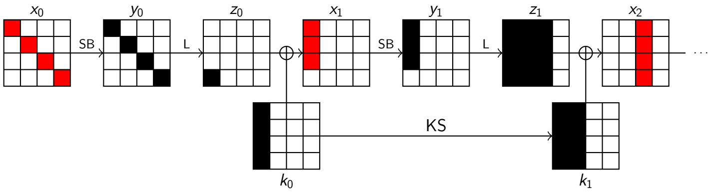
Active S-Boxes



Number of active S-boxes ⇒ maximal probability of the (truncated) differential characteristic.

The higher the minimal number of active S-boxes is, the better.

Active S-Boxes

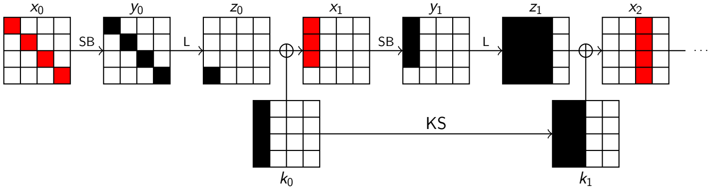


Number of active S-boxes $\Rightarrow$ maximal probability of the (truncated) differential characteristic.

The higher the minimal number of active S-boxes is, the better.

How to choose the key schedule to maximize the minimal number of active S-Boxes ?

Active S-Boxes



Number of active S-boxes ⇒ maximal probability of the (truncated) differential characteristic.

The higher the minimal number of active S-boxes is, the better.

How to choose the key schedule to maximize the minimal number of active S-Boxes ?

⇒ What if we use a byte-permutation instead of the original KS ?

Changing the key schedule for a permutation

Using a permutation as key schedule :

- Efficient in both hardware and software
- Easier to analyze
- Better security with simpler design ?
- Khoo *et al.*[2] gave an example of a permutation for AES-128 reaching 22 S-boxes in 7 rounds at FSE'18

[2] Khoo, K., Lee, E., Peyrin, T., Sim, S.M.: Human-readable Proof of the Related-Key Security of AES-128, FSE'18

About Khoo *et al.* 's permutation

- Built according to some results in their paper and two criteria :
    - Only having one cycle (of length 16)
    - Minimizing the "overlap" between the Key Schedule and the round function

About Khoo *et al.* 's permutation

- Built according to some results in their paper and two criteria :
  - Only having one cycle (of length 16)
  - Minimizing the "overlap" between the Key Schedule and the round function
- Reach 14, 18 and 21 active S-boxes over respectively 5, 6 and 7 rounds

About Khoo *et al.* 's permutation

- Built according to some results in their paper and two criteria :
  - Only having one cycle (of length 16)
  - Minimizing the "overlap" between the Key Schedule and the round function
- Reach 14, 18 and 21 active S-boxes over respectively 5, 6 and 7 rounds

But actually...

- Reach 22 S-boxes over 7 rounds when considering equations
- Easy to generate randomly ($\sim$ 100 trials)

About Khoo *et al.* 's permutation

- Built according to some results in their paper and two criteria :
  - Only having one cycle (of length 16)
  - Minimizing the "overlap" between the Key Schedule and the round function
- Reach 14, 18 and 21 active S-boxes over respectively 5, 6 and 7 rounds

But actually...

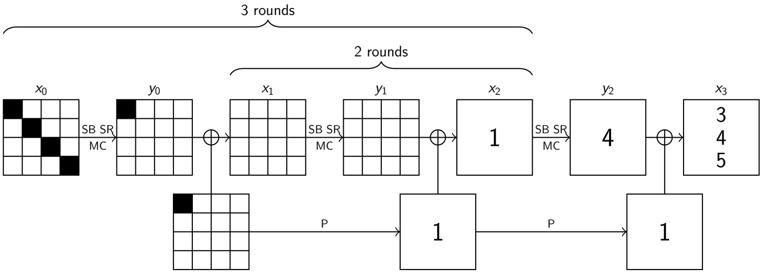
- Reach 22 S-boxes over 7 rounds when considering equations
- Easy to generate randomly ($\sim$ 100 trials)

Goal : Find a permutation to use instead of the key schedule reaching 22 S-Boxes in 6 rounds (or less ?)

Generic Bounds on 2, 3 and 4 rounds

Formally proven [Our paper]

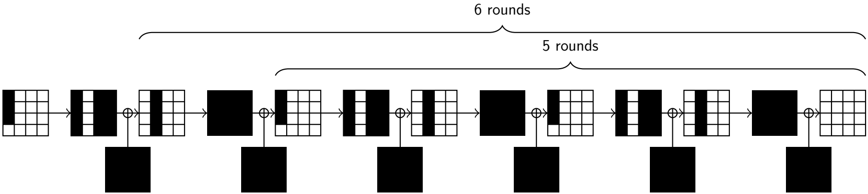
The optimal bounds for 2, 3 and 4 rounds are respectively 1, 5 and 10 active S-boxes, even when considering induced equations



Generic Bounds on 5, 6 and 7 rounds

Formally proven [Our paper]

The optimal bounds for 5, 6 and 7 rounds are respectively 14, 18 and 21 active S-boxes, *without considering equations*



More precise bound over 5 rounds

Computer aided [Our paper]

There is no permutation that, when used as key schedule, can reach a minimal number of active S-boxes of 18 or higher over 5 rounds.
There is at least one permutation that can reach 16 S-boxes over 5 rounds.

Main idea to search for s S-boxes:

- Build a list of cycles which don't lead to any characteristic of weight $< s$.
- Combine all of them to see if we can find a permutation reaching s S-boxes.

Iteratively building cycles

$(x_0\ x_1\ x_2\ ?\ ?\ \ldots)$

Iteratively building cycles

$(x_0\ x_1\ x_2\ ?\ ?\ \dots)$

Closed cycle

$(x_0\ x_1\ x_2)$

Keep if
no characteristic
of weight $< s$

Iteratively building cycles

$(x_0\ x_1\ x_2\ ?\ ?\ \ldots)$

Closed cycle

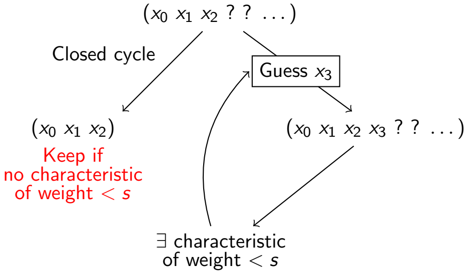
$\boxed{\text{Guess } x_3}$

$(x_0\ x_1\ x_2)$

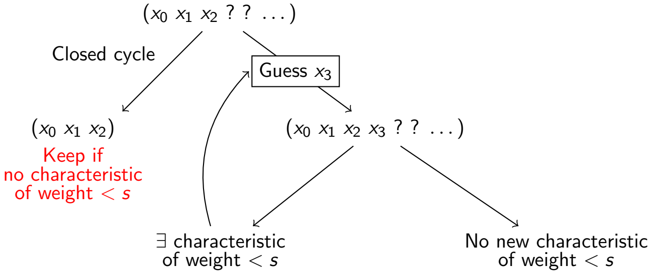
Keep if
no characteristic
of weight $< s$

$(x_0\ x_1\ x_2\ x_3\ ?\ ?\ \ldots)$

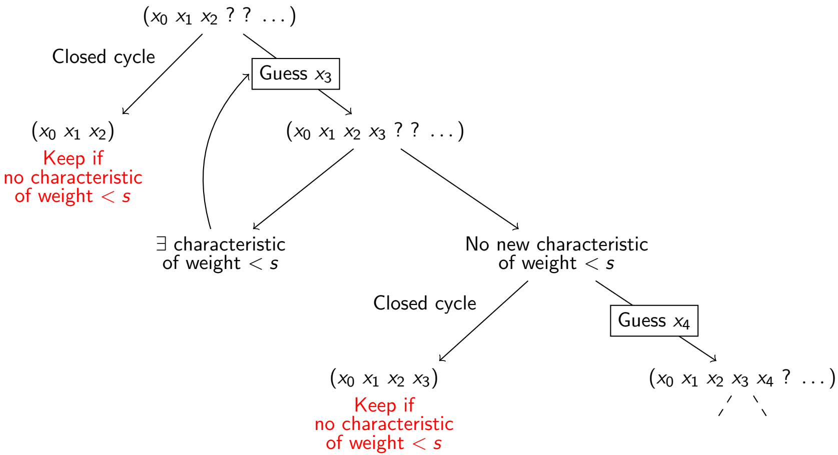
Iteratively building cycles



Iteratively building cycles



Iteratively building cycles



Over 6 rounds

More than 2^{44} possible permutations + cost of finding the minimal number of active S-boxes
$\Rightarrow$ Too expensive to try them all !

We have an optimization problem :

Maximize the minimal number of active S-boxes over 6 rounds

Over 6 rounds

More than 2^{44} possible permutations + cost of finding the minimal number of active S-boxes
$\Rightarrow$ Too expensive to try them all !

We have an optimization problem :

Get a high enough minimal number of active S-boxes over 6 rounds

Over 6 rounds

More than 2^{44} possible permutations + cost of finding the minimal number of active S-boxes
$\Rightarrow$ Too expensive to try them all !

We have an optimization problem :

Get a high enough minimal number of active S-boxes over 6 rounds

Metaheuristic + Constraint Programming

We used a meta-heuristic called *simulated annealing*[3]. Main idea :

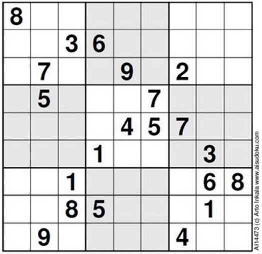
- Generate a sequence $x_0, x_1, \dots$ where x_i and x_{i+1} are "close"
- If $f(x_i) > f(x_{i-1})$, accept x_i and search for the next one
- Otherwise only accept x_i with a certain (decreasing) probability
- Choose another x_i if it was rejected
- Stop when $f(x_i)$ reach a certain threshold

[3] Nikolić, *How to use metaheuristics for design of symmetric-key primitives* - ASIACRYPT'17

Constraint Programming

Sudoku's rules :

- All values in a row are different
- All values in a column are different
- All values in a square are different
- You have knowledge of a few values to start with



Claimed to be the "World's Hardest Sudoku"

Constraint Programming

```
allDifferent(x[i][0],...,x[i][8]) for i in {0,...,8}
allDifferent(x[0][i],...,x[8][i]) for i in {0,...,8}
allDifferent(s[i][0],...,s[i][8]) for i in {0,...,8}
```
Initial values : `x[0][0] = 8, x[1][2] = 3, etc.`

↓

Constraint Solver

↓

Solution

(Previous sudoku solved in less than 0.1 seconds)

Efficient evaluation of f

Efficiency of the meta-heuristic
= Efficiency of evaluating the minimal number of active S-boxes !

Candidate permutation P
s = Target number of S-boxes

Quick search
a = weight of a valid characteristic

$a < s$

Return a
P cannot reach our target

$a \geq s$

Full search with Constraint Programming model
We manage equations here!

Return the true minimal weight

Summary of the search over 6 rounds

- We used a meta-heuristic for an efficient search.
- We proposed a new CP model which directly manages induced equations.
- We found a permutation reaching 20 active S-boxes over 6 rounds, and no characteristic with a probability better than 2^{-128} exists !

Conclusion

Number of rounds	2	3	4	5	6	7
Original key schedule	1	3	9	11	$13^\dagger$	$15^\dagger$
Khoo *et al.*'s permutation	1	5	10	14	$18^\dagger$	$22^\dagger$
Our permutation	1	5	10	15	$20^\dagger$	$23^\dagger$

- We cannot reach 18 S-boxes over 5 rounds, and 17 is still an open question.
- Modifying the ShiftRows operation, we can reach $21^\dagger$ S-boxes over 6 rounds.
- 22 S-boxes is an open question

† no characteristic with probability $> 2^{-128}$

- Long term goal : The "Ultimate" GFN
  $\Rightarrow$ Probably not unique, need to consider trade-offs (harder than focusing on optimality)
  $\Rightarrow$ Would lead to a nice generic tool for evaluating the security of any GFN (to some extend)
- "Provable" key-schedules $\Rightarrow$ Adding concrete and well defined security arguments for the key-schedule
  $\Rightarrow$ In the end, I would like to show that using a very simple key-schedule is enough, *i.e.* convoluted key-schedules are not better than a carefully crafted simple one
- Automatic tools for cryptanalysis
  $\Rightarrow$ Improving the current ones
  $\Rightarrow$ New tools for new attacks