

Searching for differential attacks

Patrick Derbez

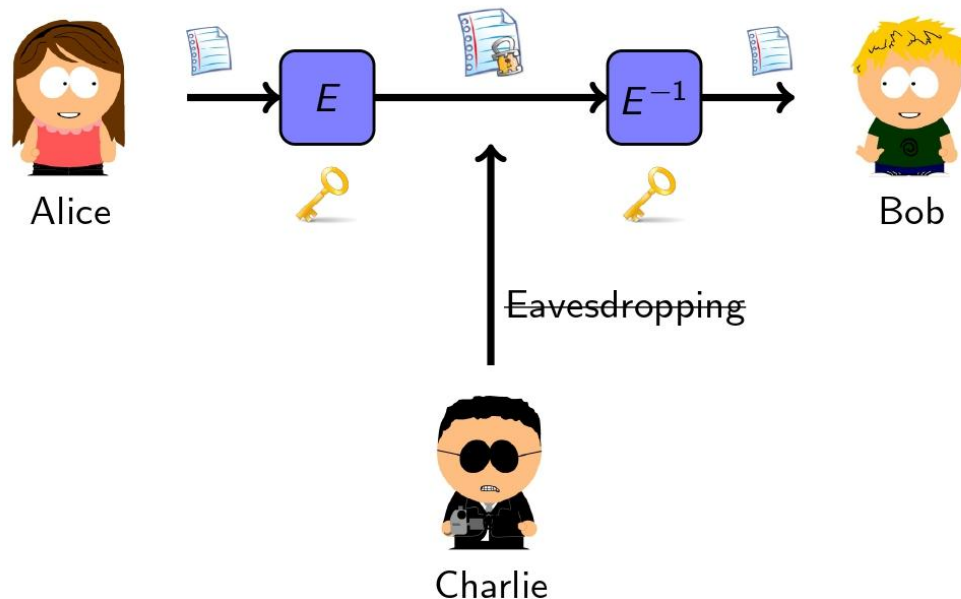
Many slides from: C. Boura, M. Eichlseder, M. Funk, B. Germon, J. Jean, L. Song

Symmetric key cryptography

Alice and Bob want to share a file, and **share a secret key**.

Symmetric-Key Primitives

- block ciphers, stream ciphers
- hash functions
- message authentication code
- authenticated encryption



Block-Cipher Cryptanalysis

a Block Cipher

$$E : \underbrace{\{0, 1\}^k}_{\text{key}} \times \underbrace{\{0, 1\}^n}_{\text{plaintext}} \rightarrow \underbrace{\{0, 1\}^n}_{\text{ciphertext}}$$

- Secure block cipher: no way to distinguish it from a random permutation.

Cryptanalysis

- Recover the secret key
- Highlight unexpected behaviours

Ressources and Constraints

a Block Cipher

$$E : \underbrace{\{0, 1\}^k}_{\text{key}} \times \underbrace{\{0, 1\}^n}_{\text{plaintext}} \rightarrow \underbrace{\{0, 1\}^n}_{\text{ciphertext}}$$

Constraints

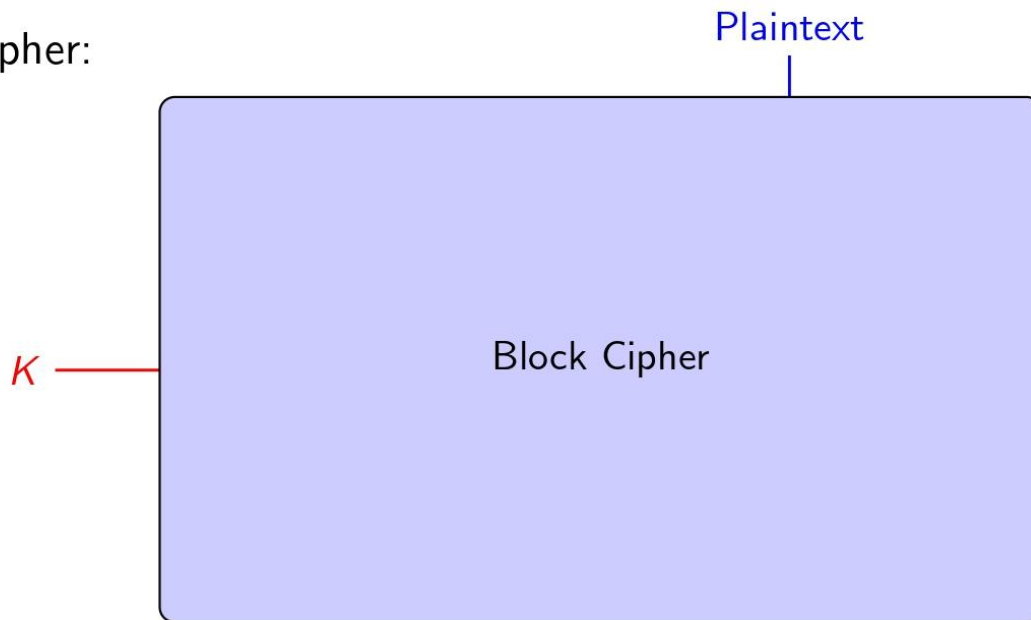
- Time: less than 2^k encryptions
- Data: less than 2^n plaintext/ciphertext pairs

Adversarial Models

- known plaintexts
- (adaptively) chosen plaintexts
- (adaptively) chosen ciphertexts
- related keys, related subkeys
- side channels, faults injection

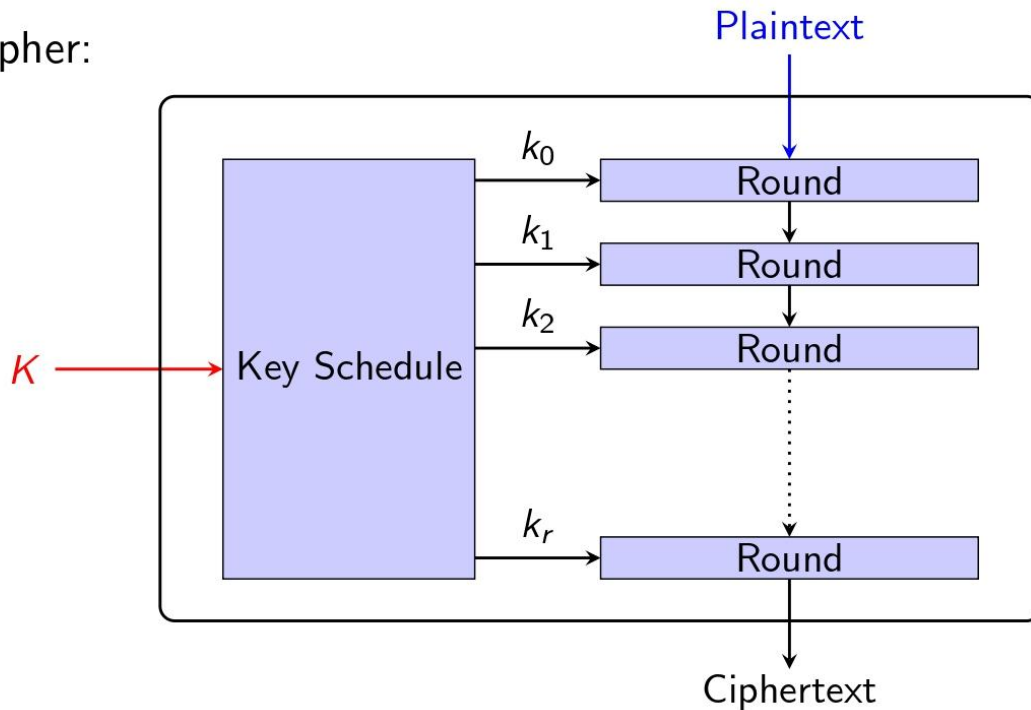
Security Margin

If no attack is found on a given cipher:



Security Margin

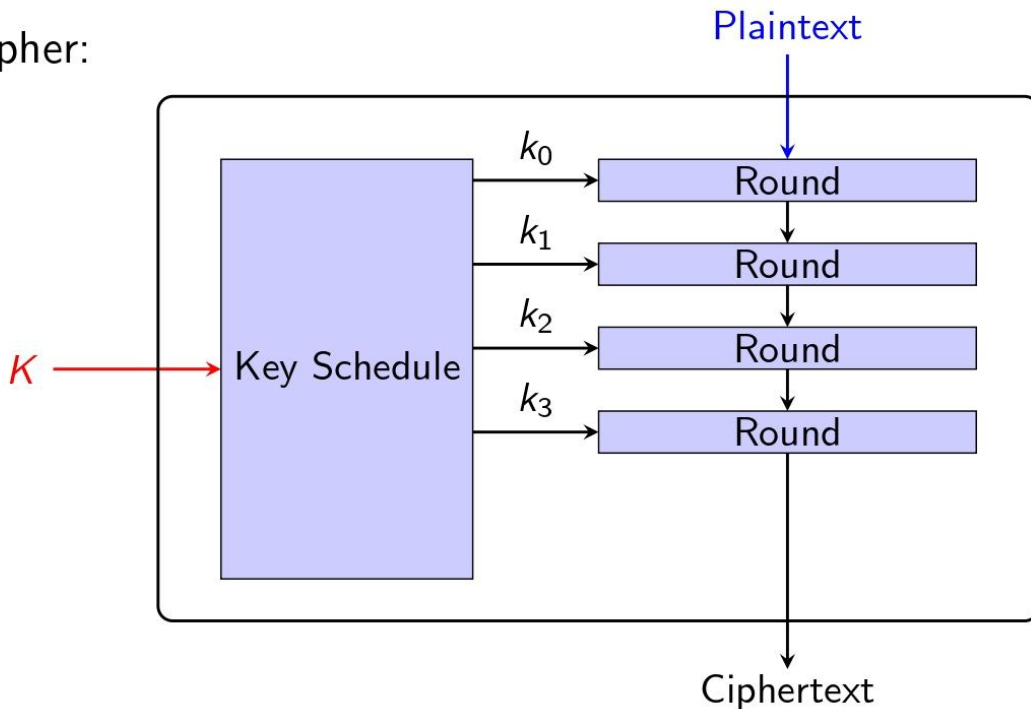
If no attack is found on a given cipher:



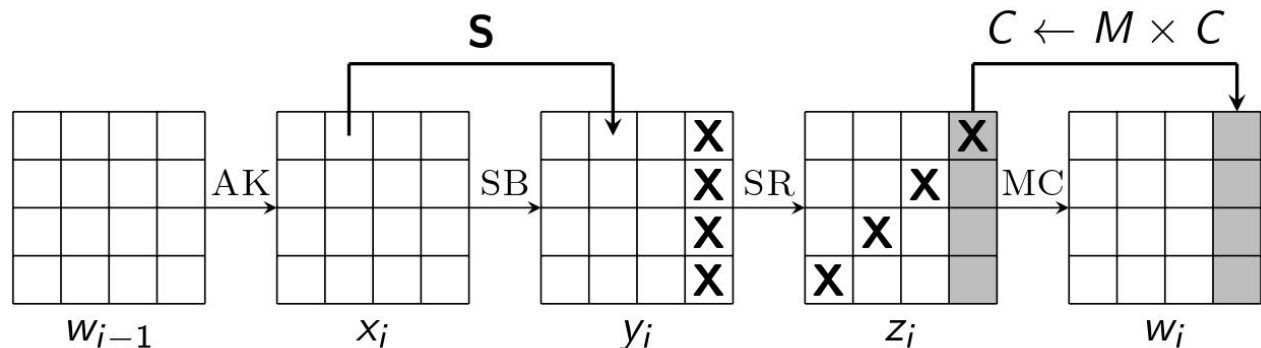
Security Margin

If no attack is found on a given cipher:

- Study **round-reduced** versions
- Study **internal** components
- Artificial?
 - Attacks only get better
 - **Better safe than sorry!**



AES



- Standardized in 2001 for 3 key lengths: 128, 192 and 256 bits
- Block size of 128 bits : 4×4 matrix of bytes
- An AES round applies $MC \circ SR \circ SB \circ AK$ to the state
- No MixColumns in the last round

Known Cryptanalysis Techniques

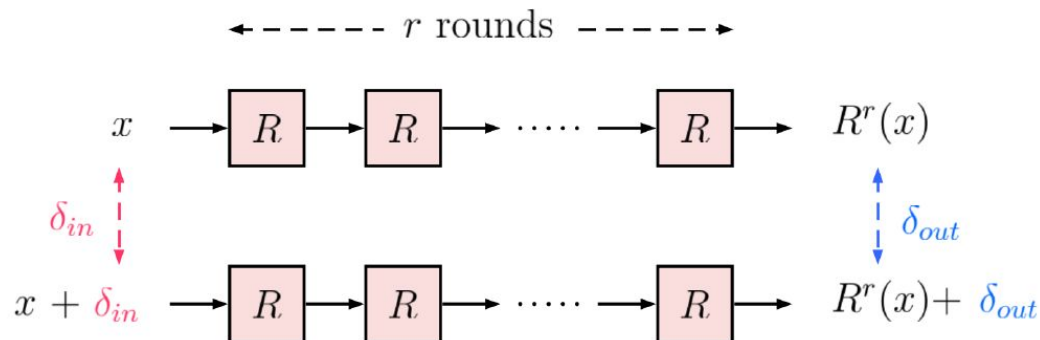
- Differential cryptanalysis
- Linear cryptanalysis
- Integral distinguishers
- Invariant attacks
- Meet-in-the-Middle attacks
- Algebraic attacks

Goals

- Find the **best** cryptanalysis technique against a particular target
- Apply the technique with **optimal** settings

Differential cryptanalysis

- Cryptanalysis technique introduced by [Biham](#) and [Shamir](#) in **1990**.
- Based on the existence of a high-probability **differential** (δ_{in} , δ_{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 Cryptanalysis – Overview

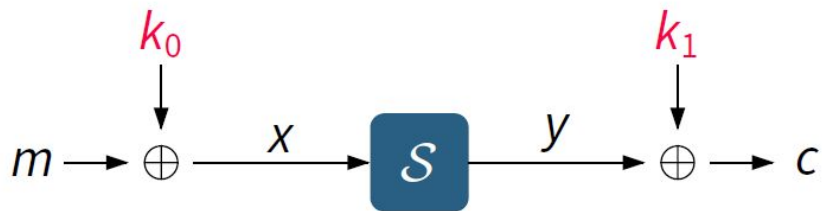
💡 Main idea:

1. Predict the effect of a plaintext difference $\Delta M = \text{📄 } M \oplus \text{📄 } M^*$ on the ciphertext difference $\Delta C = \text{✉️ } C \oplus \text{✉️ } C^*$ without knowing 🔑 K
2. Use prediction to recover the key

A Simple Toy Block Cipher

The block cipher $E_{k_0||k_1}(m)$ encrypts 4 bits of plaintext using two 4-bit keys:

$$c = E_{k_0||k_1}(m) = \mathcal{S}(m \oplus k_0) \oplus k_1$$



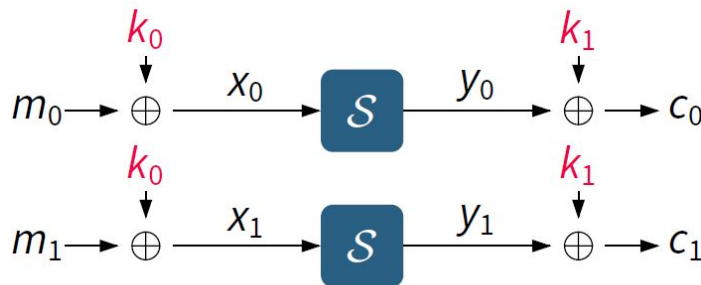
S-box	x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

Given $(m_0, c_0) = (a, 9)$ and $(m_1, c_1) = (5, 6)$, what is the key?

Brute force (exhaustive search): try all $2^4 \cdot 2^4 = 256$ keys.

The Basic Idea

Assume we know two plaintext-ciphertext pairs (m_0, c_0) , (m_1, c_1) :



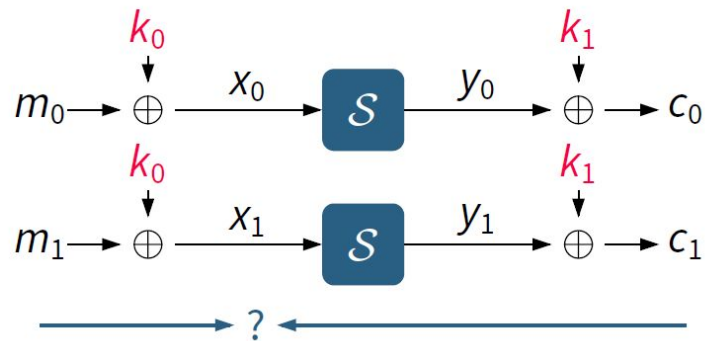
Observation

Even though we do not know k_0 and k_1 , we can derive

$$x_0 \oplus x_1 = (m_0 \oplus k_0) \oplus (m_1 \oplus k_0) = m_0 \oplus m_1$$

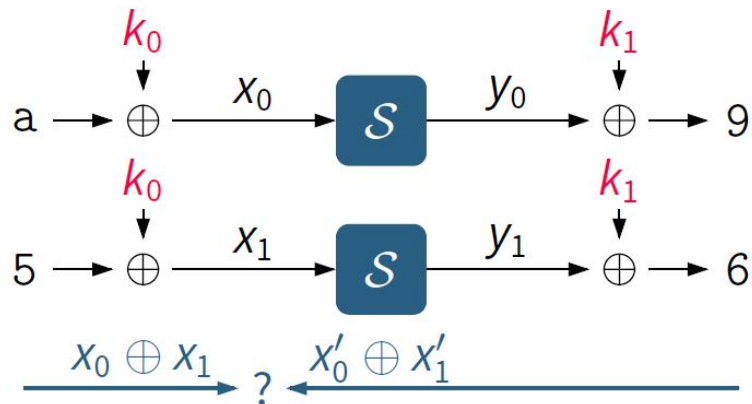
$$y_0 \oplus y_1 = (c_0 \oplus k_1) \oplus (c_1 \oplus k_1) = c_0 \oplus c_1$$

Differential Attack (1-Round Toy Version)



1. Compute $x_0 \oplus x_1$
2. Guess k_1 (iterate over all values)
3. Compute $x'_0 = \mathcal{S}^{-1}(c_0 \oplus k'_1)$ and $x'_1 = \mathcal{S}^{-1}(c_1 \oplus k'_1)$
4. Check if $x_0 \oplus x_1 = x'_0 \oplus x'_1$
5. If not: key guess was definitely wrong! (filtering)

Example for $(m_0, c_0) = (a, 9)$ and $(m_1, c_1) = (5, 6)$

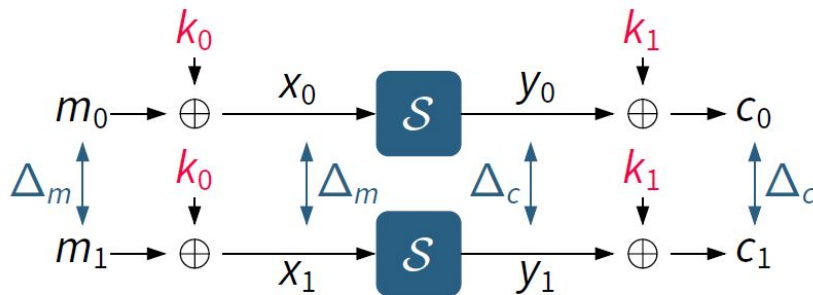


1. Compute $x_0 \oplus x_1 = m_0 \oplus m_1 = a \oplus 5 = f$
2. Guess k_1 and compute $x'_0 \oplus x'_1$:

k'_1	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$x'_0 \oplus x'_1$	e	9	e	e	d	8	d	f	f	d	8	d	e	e	9	e

3. Only two candidates for k_1 are valid: $k_1 \in \{7, 8\}$

Differential Cryptanalysis: Observations



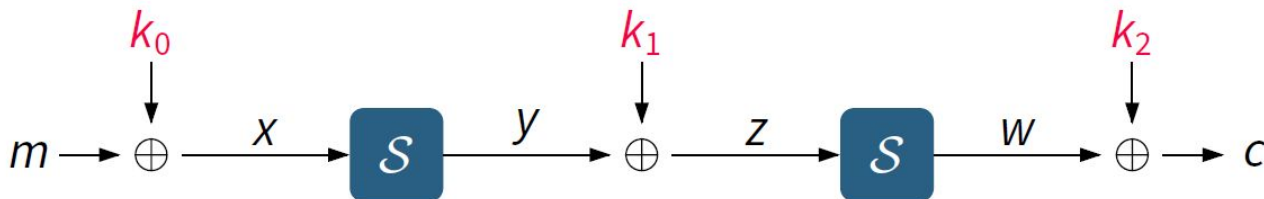
What happened?

- We can get information about the differences, even though we do not know the values
- We can make a guess for the (last) key and verify it by computing backwards

Let's Extend it to a 2-Round Cipher

The block cipher $E_{k_0||k_1||k_2}(m)$ encrypts 4 bits of plaintext using three 4-bit keys:

$$c = E_{k_0||k_1||k_2}(m) = \mathcal{S}(\mathcal{S}(m \oplus k_0) \oplus k_1) \oplus k_2$$

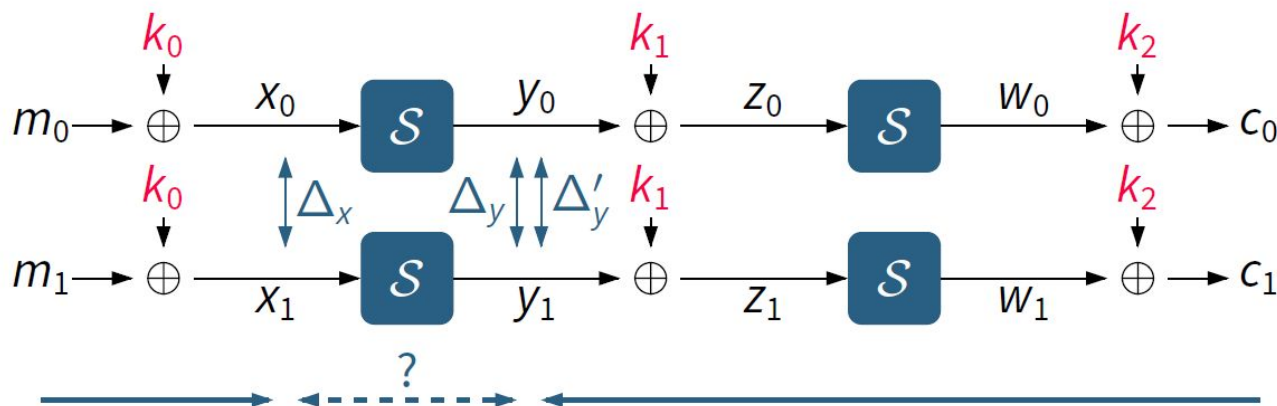


We use the same 4-bit S-box \mathcal{S} :

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

Brute force: $2^{4+4+4} = 4096$ keys.

Differential Attack (2-Round Toy Version)



- ✓ We can again deduce $\Delta x = \Delta m = m_0 \oplus m_1$
- ✓ We can again guess k_2 to compute a candidate $\Delta y'$
- ❓ Can we get more information on the real Δy using only Δx , but not x_0, x_1 ?

The Influence of the S-Box: Example for Input Difference $\Delta x = f$

x_0	$x_1 = x_0 \oplus f$	$y_0 = S(x_0)$	$y_1 = S(x_1)$	$\Delta y = y_0 \oplus y_1$
0	f	6	b	d
1	e	4	9	d
2	d	c	a	6
3	c	5	8	d
4	b	0	d	d
5	a	7	3	4
6	9	2	f	d
7	8	e	1	f
8	7	1	e	f
9	6	f	2	d
a	5	3	7	4
b	4	d	0	d
c	3	8	5	d
d	2	a	c	6
e	1	9	4	d
f	0	b	6	d

Observations

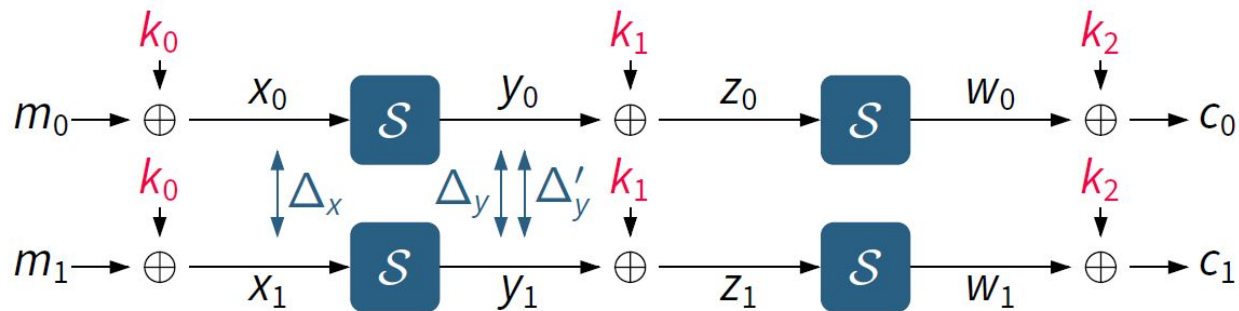
- Only 4 differences for Δy are possible for $\Delta x = f$.
- One difference, $\Delta y = d$, occurs very often (10 of 16 times)

💡 Let's assume $\Delta y = (\Delta y' =) d$

➤ Correct with prob. 10/16

➤ Verify our guess k'_2 by checking whether $\Delta y' = d$

Differential Attack (2-Round Toy Version)



1. Consider 16 plaintext-ciphertext pairs (m_0^i, c_0^i) and (m_1^i, c_1^i) such that $m_0^i \oplus m_1^i = \Delta_x = \text{f}$ for all $i = 0, \dots, 15$
2. Guess the last round key k_2 (iterate over all values)
3. For each plaintext-ciphertext pair: compute w'_0, w'_1, z'_0, z'_1 and count the number of pairs for which $\Delta y' = \Delta z' = \text{d}$;
4. For the *right key*, approx. $16 \cdot \frac{10}{16} = 10$ pairs satisfy $\Delta y' = \text{d}$;
for a *wrong key*, approx. $16 \cdot \frac{1}{16} = 1$ pair satisfies $\Delta y' = \text{d}$ (why? really?)

Difference Distribution Table (DDT)

How can we find differences with a good probability?

- compute all possible output differences for all input differences of an S-box
- or equivalently: compute the number of solutions x to the equation

$$\mathcal{S}(x \oplus \Delta x) \oplus \mathcal{S}(x) = \Delta y$$

Definition

Let f be an n -bit to m -bit function. The **difference distribution table** of f is a $2^n \times 2^m$ table whose entries are the number of valid solutions x for each differential $\Delta x \rightarrow \Delta y$.

Maximum Differential Probability

The *Differential Probability* (DP) of the function f is defined as

$$\begin{aligned}\text{DP}_f(\Delta x \rightarrow \Delta y) &:= \mathbb{P}[f(x \oplus \Delta x) \oplus f(x) = \Delta y] \\ &= \frac{|\{x \in \mathbb{F}_2^n \mid f(x \oplus \Delta x) \oplus f(x) = \Delta y\}|}{2^n}\end{aligned}$$

Definition

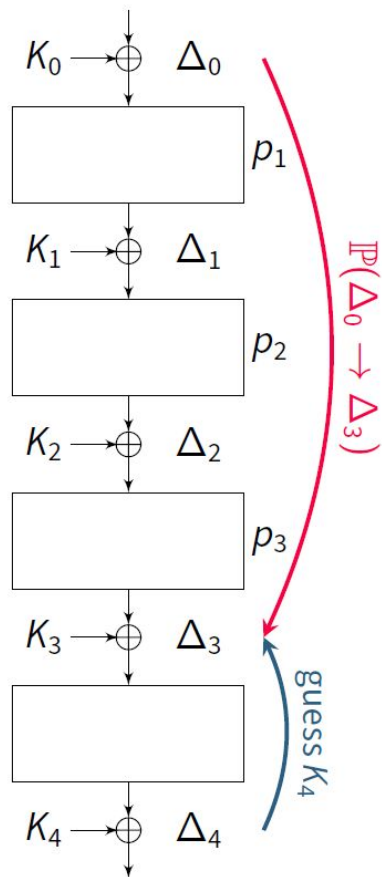
The **Maximum Differential Probability** (MDP) is defined as

$$\text{MDP}_f := \max_{\Delta x \neq 0, \Delta y} \text{DP}_f(\Delta x \rightarrow \Delta y).$$

In the previous example:

$$\text{MDP}_{\mathcal{S}} = \max_{\Delta x \neq 0, \Delta y} \frac{|\{x \in \mathbb{F}_2^n \mid \mathcal{S}(x \oplus \Delta x) \oplus \mathcal{S}(x) = \Delta y\}|}{16} = \frac{10}{16}$$

Basic Approach of a Differential Attack



1. Find a “good” differential characteristic

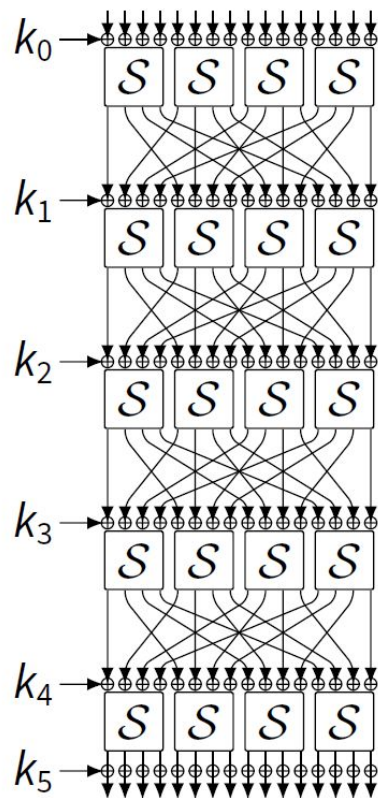
$$\Delta_0 \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta_3$$

2. Guess final key K'_4 and compute backward through the S-boxes to determine Δ'_3
3. The right key satisfies $\Delta'_3 = \Delta_3$ with prob. $\mathbb{P}(\Delta_0 \rightarrow \Delta_3)$; a wrong key satisfies $\Delta'_3 = \Delta_3$ with prob. $1/2^n = 2^{-n}$ (for n -bit block size)

4. *Necessary condition* for the attack to work:

$$\mathbb{P}(\Delta_0 \rightarrow \Delta_3) \gg 2^{-n}$$

Diff. Probability for Multiple Rounds – A 4-Round Toy-Cipher



x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

➤ Consider the input difference $(0 \ 0 \ 2 \ 0)$ and the DDT:

$\Delta x \setminus \Delta y$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
\vdots																

➤ This 1-round characteristic holds with prob. $1 \cdot \frac{6}{16} \cdot 1 \cdot 1$:

$$(0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0)$$

Characteristic and Differential

This 1-round characteristic for ToyCipher holds with probability $6/16$:

$$(0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0)$$

This 4-round **characteristic** for ToyCipher holds with probability $(6/16)^4 = \frac{81}{4096}$:

$$(0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0) \rightarrow (0 \ 0 \ 2 \ 0)$$

This 4-round **differential** for ToyCipher holds with a *higher* probability of $\frac{324}{4096}$:

$$(0 \ 0 \ 2 \ 0) \rightarrow ? \rightarrow ? \rightarrow ? \rightarrow (0 \ 0 \ 2 \ 0)$$

How to find the best differential attack?

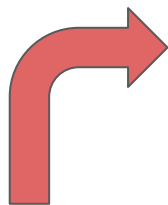
Main Problems:

- Very large search space → impossible to exhaust
 - How to find small enough subspaces containing optimal solutions?
 - How to use our intuition?
- Evaluating objectives might be an hard task
 - Computing entropy of round key bits
 - Computing probability of transitions
 - Rank problems
- Use the right tools and modelizations:
 - Dedicated algorithms
 - Generic solvers: CP, SAT/SMT, MILP

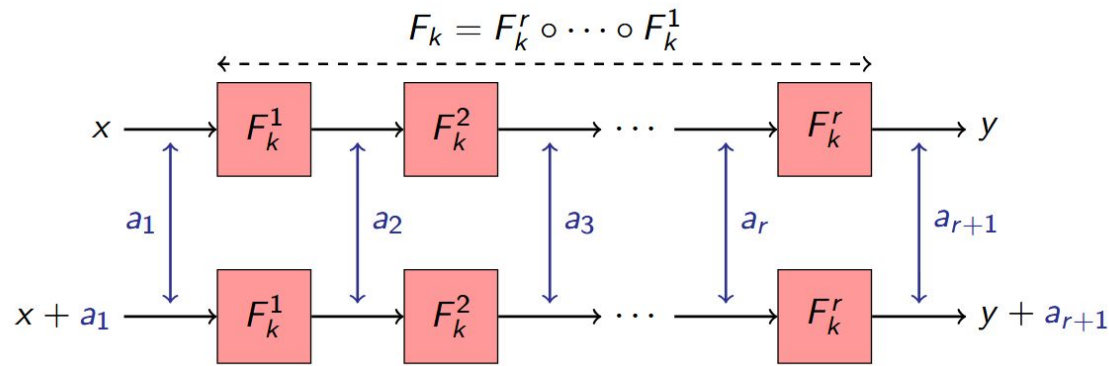
How to find the best differential attack?

Main Problems:

- Very large search space → impossible to exhaust
 - How to find small enough subspaces containing optimal solutions?
 - How to use our intuition?
- Evaluating objectives might be an hard task
 - Computing entropy of round key bits
 - Computing probability of transitions
 - Rank problems
- Use the right tools and modelizations:
 - Dedicated algorithms
 - Generic solvers: CP, SAT/SMT, MILP



Let first focus on finding good distinguishers!



Distinguisher: **differential** (a_1, a_{r+1}) such that $\Pr_{\mathbf{x}}[F_k(\mathbf{x}) + F_k(\mathbf{x} + a_1) = a_{r+1}] \gg \frac{1}{2^n}$.

Distinguisher probability estimation: **characteristic** $(a_1, a_2, \dots, a_{r+1})$ such that

$$\Pr_{\mathbf{x}}[F_k(\mathbf{x}) + F_k(\mathbf{x} + a_1) = a_{r+1}] \geq \Pr_{\mathbf{x}}\left[\bigwedge_{i=1}^r F_k^i(\mathbf{x}_i) + F_k^i(\mathbf{x}_i + a_i) = a_{i+1}\right] \gg \frac{1}{2^n}$$

with $\mathbf{x}_{i+1} = F_i(\mathbf{x}_i)$ and \mathbf{x}_1 uniform.

Classical Assumptions

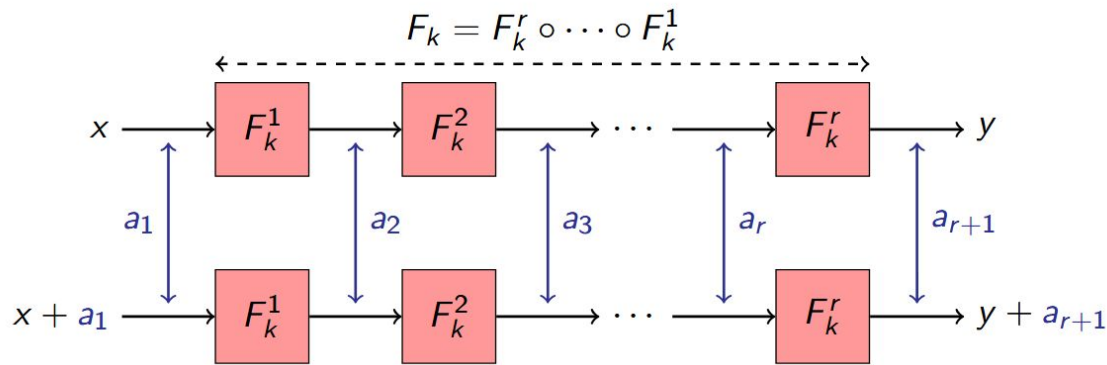
Stochastic Equivalence Hypothesis

$$\underbrace{\Pr_{\mathbf{x}} \left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1} \right]}_{\text{Fixed-key probability}} \approx \underbrace{\frac{1}{|\mathcal{K}|} \sum_{\mathbf{k} \in \mathcal{K}} \Pr_{\mathbf{x}} \left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1} \right]}_{\text{Expected Differential Probability (EDP)}}$$

Round Independence

$$\text{EDP}[a_1, \dots, a_{r+1}] \approx \prod_{i=1}^r \Pr_{\mathbf{x}_i, \mathbf{k}} [F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1}]$$

with \mathbf{x}_i, \mathbf{k} uniform.

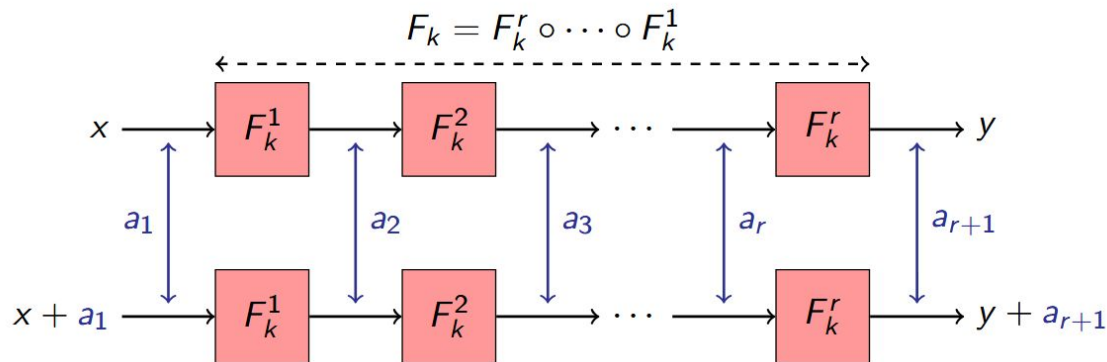


Finding the best differential:

- Pick a key k (since the probability is independent of the key)
- Init N to 0
- For each of the 2^n possible values of a_1
 - Init a table T of size 2^n to 0
 - For each of the 2^n possible values of x
 - $T[F_k(x) \oplus F_k(x \oplus a_1)] + = 1$
 - If N strictly lower than $\max(T)$ then update N

Basically, we computed the DDT for a particular value of k (and we did not fully store it)

Complexity: $O(2^{2n})$ in time and $O(2^n)$ in memory



Finding "all" differentials with probability $\geq p$ (heuristic):

- **Pick** a key k (since the probability is **independent** of the key)
- For each of the 2^n possible values of a_1
 - Init a hash table T
 - For $O(p^{-1})$ random possible values of x
 - Add $F_k(x) \oplus F_k(x \oplus a_1)$ to T
 - Output $a_1 \rightarrow a_{r+1}$ for all $a_{r+1} \in T$ appearing at least twice

Complexity: $O(2^n/p)$ in time and $O(1/p)$ in memory

Finding "all" differentials with probability $\geq p$ (heuristic):

- Define $G_k^\alpha(x) = F_k(x) \oplus F_k(x \oplus \alpha)$
- Pick a random key k , a random α and init a hash table T
- For N random possible values of x
 - For each $x' \in T[G_k^\alpha(x)]$
 - $a_1 = x \oplus x'$ and $a_{r+1} = F_k(x) \oplus F_k(x')$
 - Output (a_1, a_{r+1})
 - Add x to $T[G_k^\alpha(x)]$

Here we are looking for collisions on $G_k^\alpha(x)$ because if $F_k(x) \oplus F_k(x \oplus \alpha) = F_k(x') \oplus F_k(x' \oplus \alpha)$ then

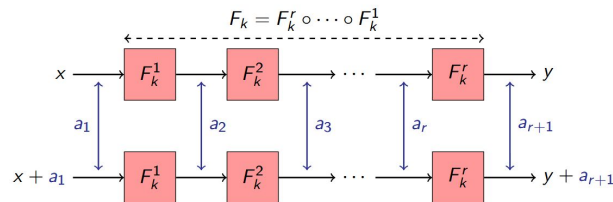
$$F_k(x) \oplus F_k(x') = F_k(x \oplus \alpha) \oplus F_k(x' \oplus \alpha)$$

The two pairs (x, x') and $(x \oplus \alpha, x' \oplus \alpha)$ follow the same differential!

Given a random pair of pairs $((x, x'), (x \oplus \alpha, x' \oplus \alpha))$

- the probability that both pairs follow the differential is p^2
- we need $O(2^n/p^2)$ pair of pairs to detect a collision

EC23: *Efficient Detection of High Probability Statistical Properties of Cryptosystems via Surrogate Differentiation*



Complexity: $O(2^{n/2}/p)$ in time and memory

Classical Assumptions

Stochastic Equivalence Hypothesis

$$\underbrace{\Pr_{\mathbf{x}} \left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1} \right]}_{\text{Fixed-key probability}} \approx \underbrace{\frac{1}{|\mathcal{K}|} \sum_{\mathbf{k} \in \mathcal{K}} \Pr_{\mathbf{x}} \left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1} \right]}_{\text{Expected Differential Probability (EDP)}}$$

Round Independence

$$\text{EDP}[a_1, \dots, a_{r+1}] \approx \prod_{i=1}^r \Pr_{\mathbf{x}_i, \mathbf{k}} [F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1}]$$

with \mathbf{x}_i, \mathbf{k} uniform.

Classical Assumptions

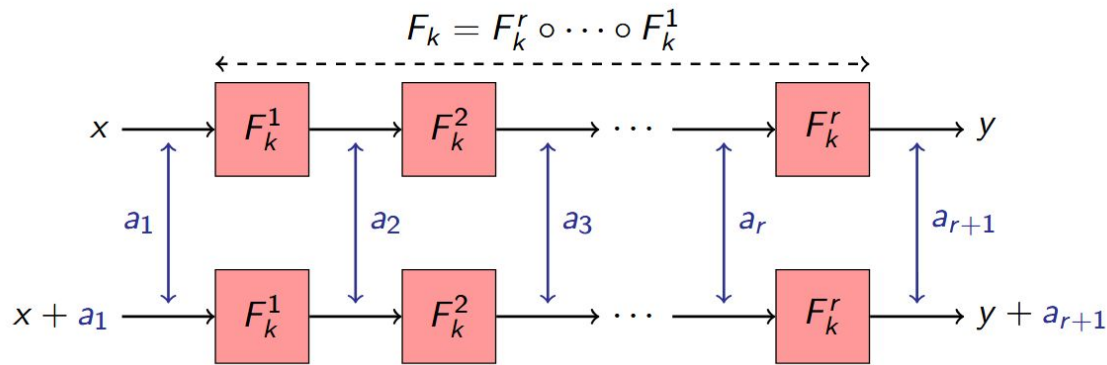
Stochastic Equivalence Hypothesis

$$\underbrace{\Pr_{\mathbf{x}}\left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1}\right]}_{\text{Fixed-key probability}} \approx \underbrace{\frac{1}{|\mathcal{K}|} \sum_{\mathbf{k} \in \mathcal{K}} \Pr_{\mathbf{x}}\left[\bigwedge_{i=1}^r F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1}\right]}_{\text{Expected Differential Probability (EDP)}}$$

Round Independence

$$\text{EDP}[a_1, \dots, a_{r+1}] \approx \prod_{i=1}^r \Pr_{\mathbf{x}_i, \mathbf{k}}[F_{\mathbf{k}}^i(\mathbf{x}_i) + F_{\mathbf{k}}^i(\mathbf{x}_i + a_i) = a_{i+1}]$$

with \mathbf{x}_i, \mathbf{k} uniform.



Using **round independence assumption**, an **optimal** differential characteristic on r rounds is composed of

- a difference a_r
- an **optimal** differential characteristic on $r - 1$ rounds, ending by a_r
- a difference a_{r+1} such that the transition $a_r \rightarrow a_{r+1}$ is optimal on one round

$$\begin{cases} \text{opt}_r = \max_{a_{r+1}} \text{opt}_r^{a_{r+1}} \\ \text{opt}_i^{a_{i+1}} = \max_{a_i} \text{opt}_i^{a_i} \times \Pr(a_i \rightarrow a_{i+1}), \text{ for } 1 \leq i \leq r \end{cases}$$

Complexity: $O(2^{2n})$ in time and $O(2^n)$ in memory

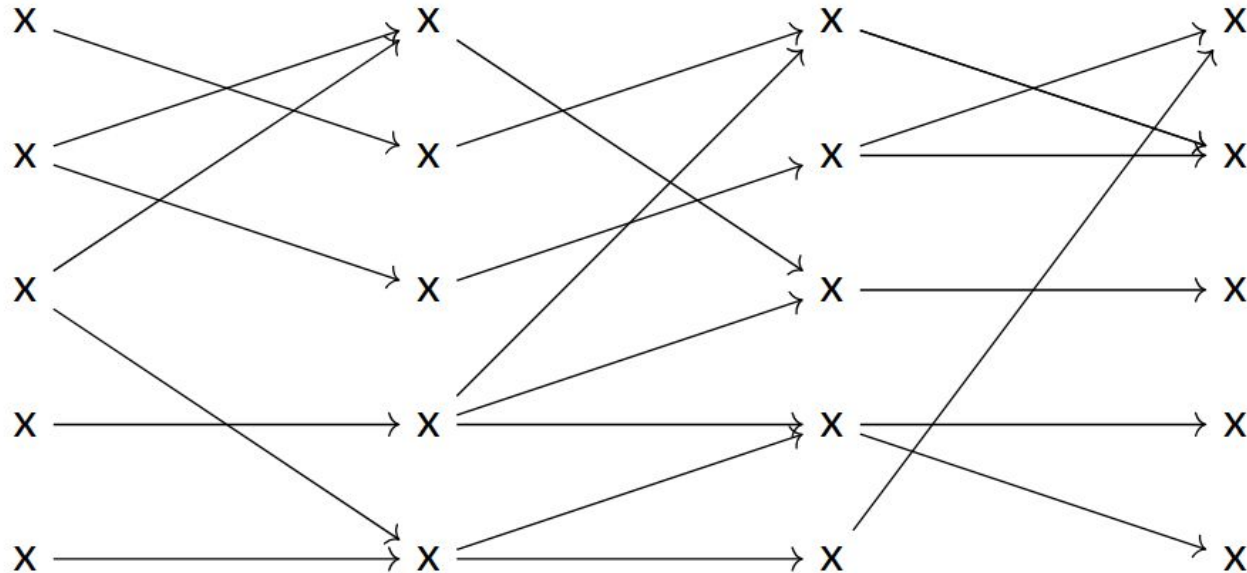
Dynamic Programming

a step-function

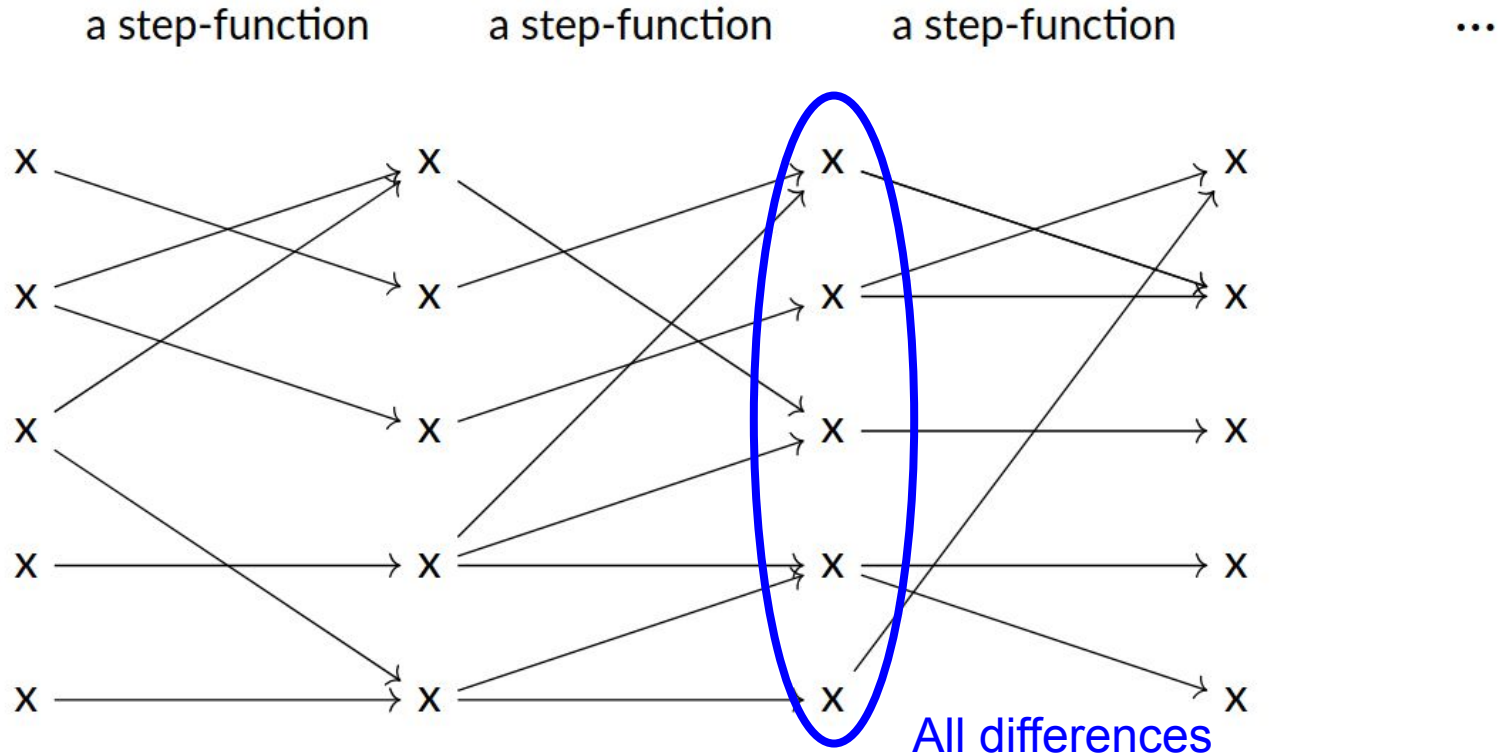
a step-function

a step-function

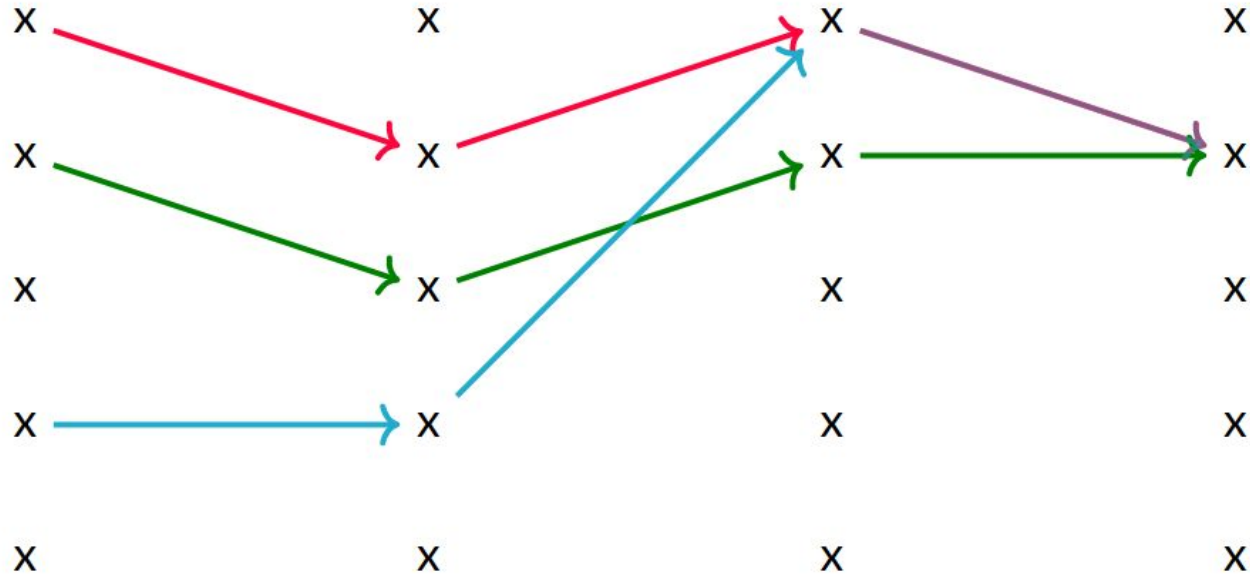
...



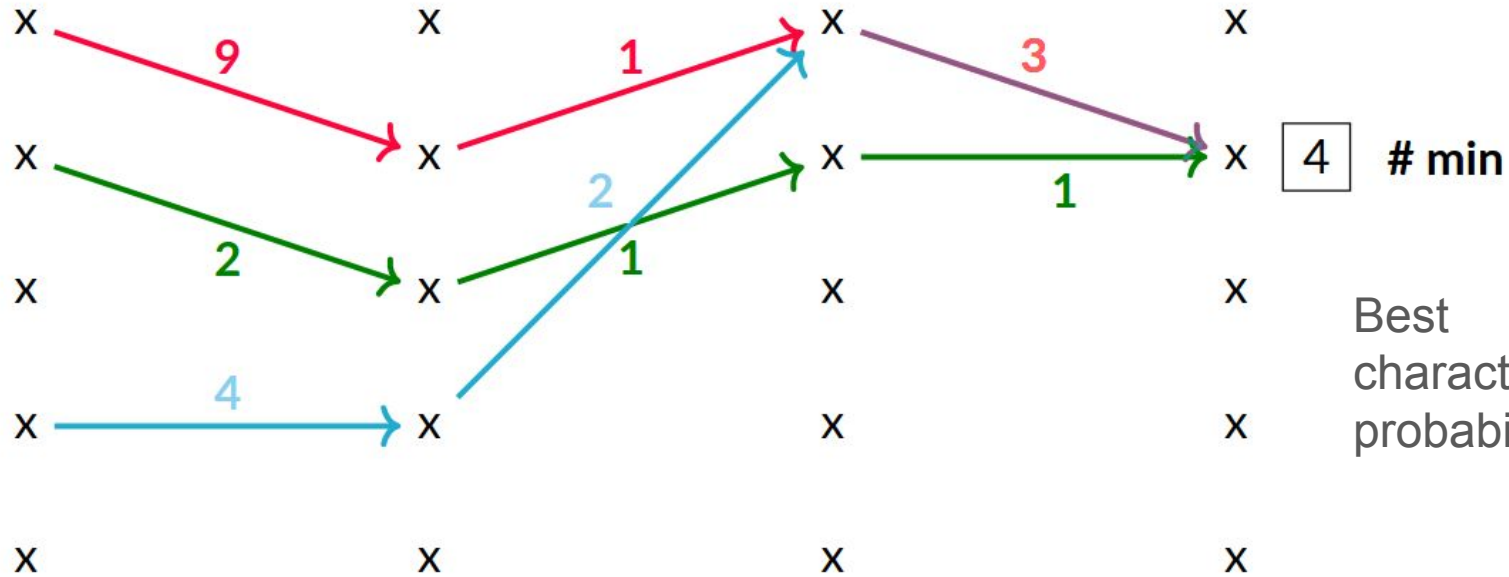
Dynamic Programming



Dynamic Programming



Dynamic Programming



Best
characteristic has
probability 2^{-4}

Dynamic Programming

x

x

x

x

9

x

x

x

x

4

x

x

x

x

6

x

x

x

x

7

x

x

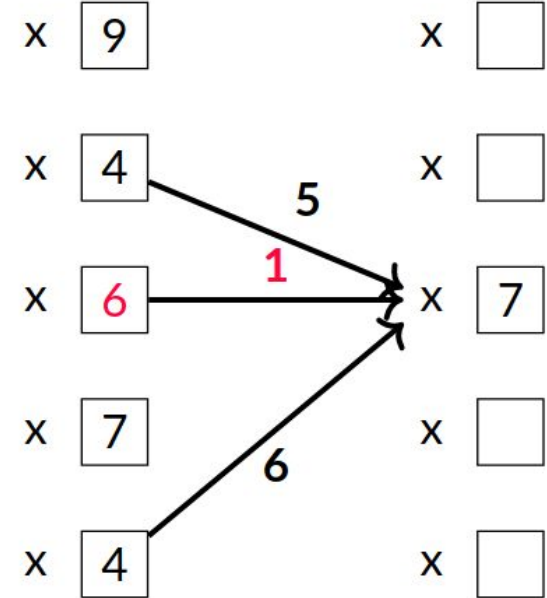
x

x

4

Dynamic Programming

x		x		x	
x		x		x	
x		x		x	
x		x		x	
x		x		x	



```

foreach state  $s$  do  $M[s] \leftarrow$  list of states  $s'$  reachable from  $s$  through one round
foreach state  $s$  do  $C[0][s] \leftarrow 1$ 
for  $1 \leq r < R$  do
    foreach state  $s$  do  $C[r][s] \leftarrow 0$ 
    foreach state  $s$  do
        foreach state  $s' \in M[s]$  do
             $c \leftarrow C[r-1][s] \times \Pr(s \rightarrow s')$ 
            if  $c > C[r][s']$  then  $C[r][s'] \leftarrow c$ 
        end
    end
end
return  $C$ 

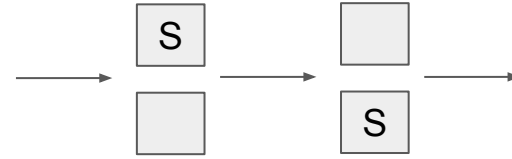
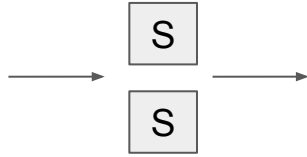
```

```

foreach state  $s$  do  $M[s] \leftarrow$  list of states  $s'$  reachable from  $s$  through one round
foreach state  $s$  do  $C[0][s] \leftarrow 1$ 
for  $1 \leq r < R$  do
  | foreach state  $s$  do  $C[r][s] \leftarrow 0$ 
  | foreach state  $s$  do
  |   | foreach state  $s' \in M[s]$  do
  |   |   |  $c \leftarrow C[r-1][s] \times \Pr(s \rightarrow s')$ 
  |   |   | if  $c > C[r][s']$  then  $C[r][s'] \leftarrow c$ 
  |   | end
  | end
end
return  $C$ 

```

Need to minimize!



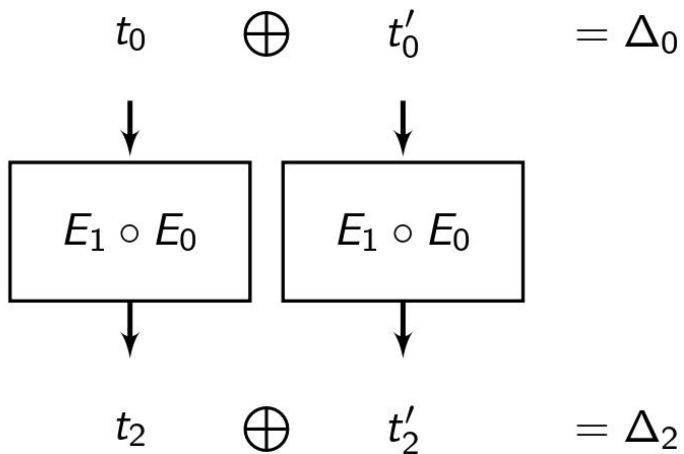
Assume on average $2^{n/2-1}$ reachable differences for S whatever the input difference

- Around $2^{n/2-1} \times 2^{n/2-1} = 2^{n-2}$ reachable differences

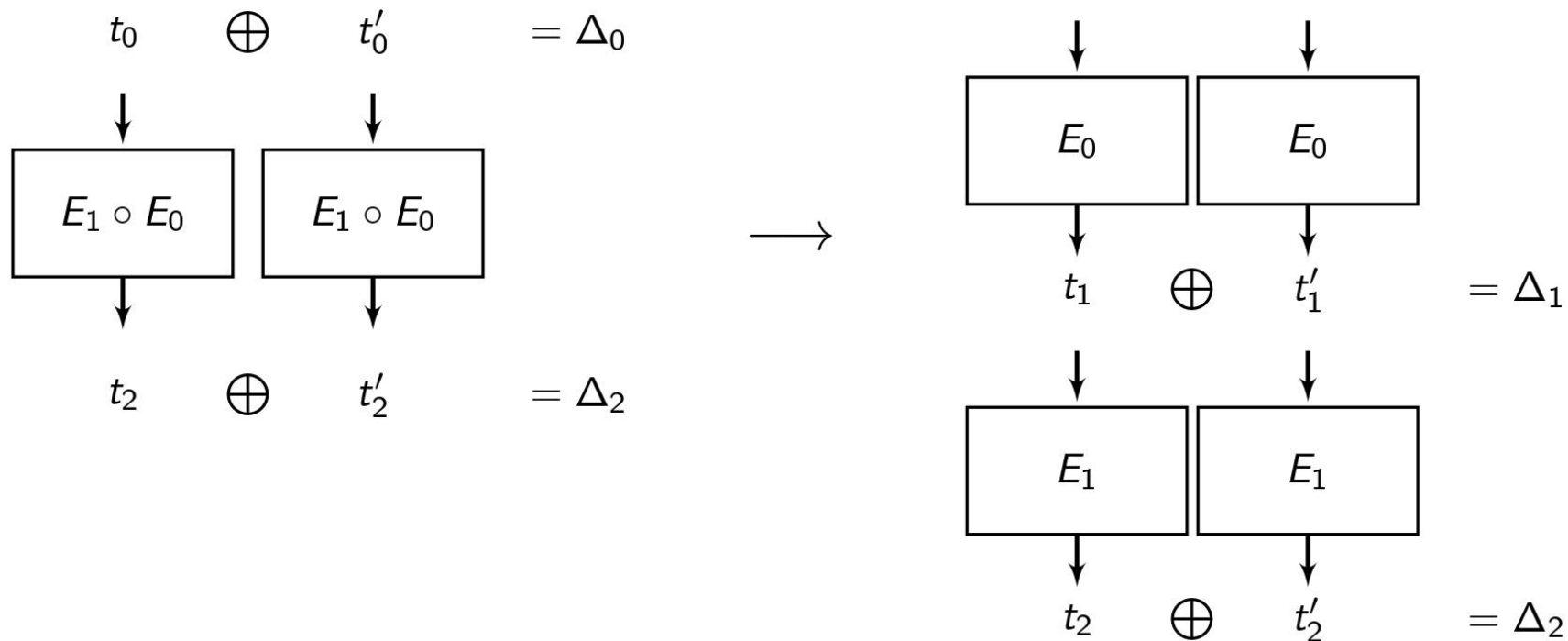
- Around $2^{n/2-1} \times 1 = 2^{n/2-1}$ reachable differences

If one round is composed of n/m m -bit S -boxes, then time complexity is $O(n/m \times 2^{n+m})$

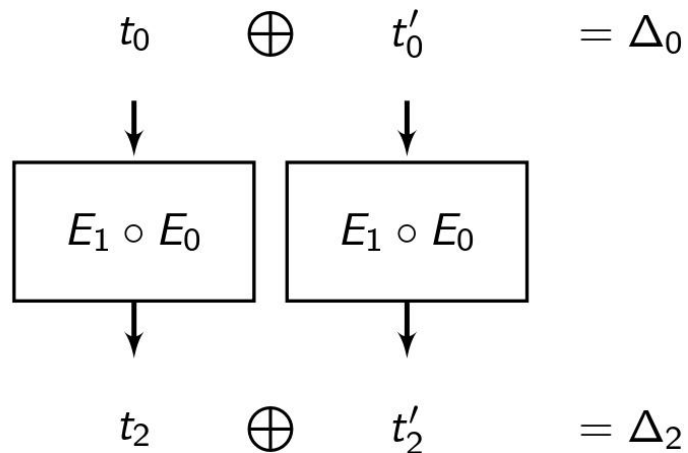
Differential Distinguisher



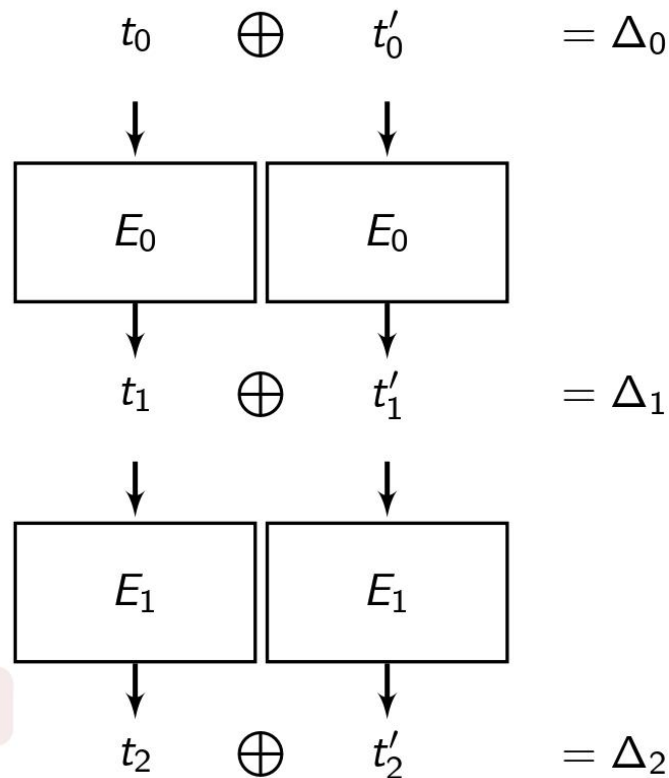
Differential Distinguisher



Differential Distinguisher

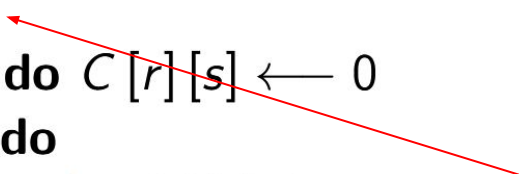


→



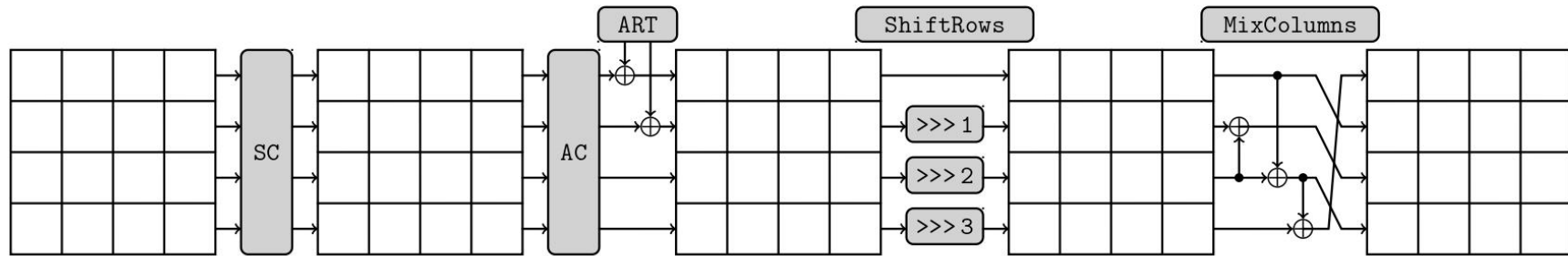
Still very hard!

```
foreach state  $s$  do  $M[s] \leftarrow$  list of states  $s'$  reachable from  $s$  through one round
foreach state  $s$  do  $C[0][s] \leftarrow 1$ 
for  $1 \leq r < R$  do
  foreach state  $s$  do  $C[r][s] \leftarrow 0$ 
  foreach state  $s$  do
    foreach state  $s' \in M[s]$  do
       $c \leftarrow C[r-1][s] \times \text{Pr}(s \rightarrow s')$ 
      if  $c > C[r][s']$  then  $C[r][s'] \leftarrow c$ 
    end
  end
end
return  $C$ 
```



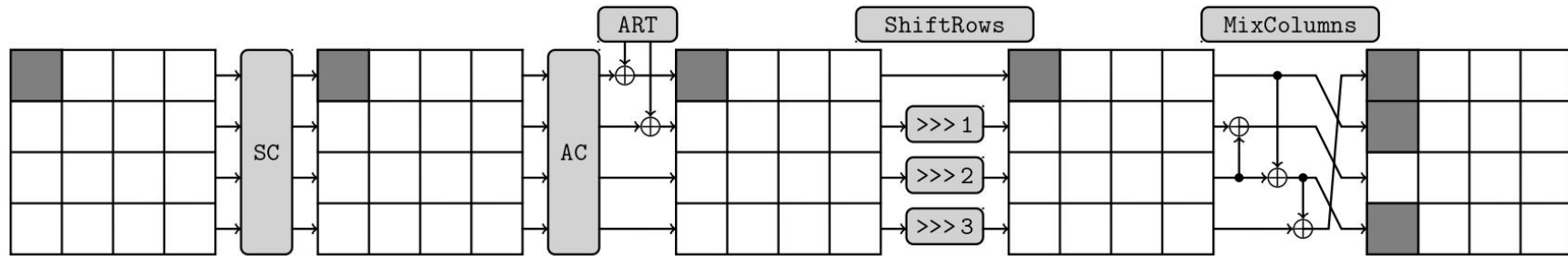
Can we minimize this ?

Truncated Differential Characteristics on SKINNY



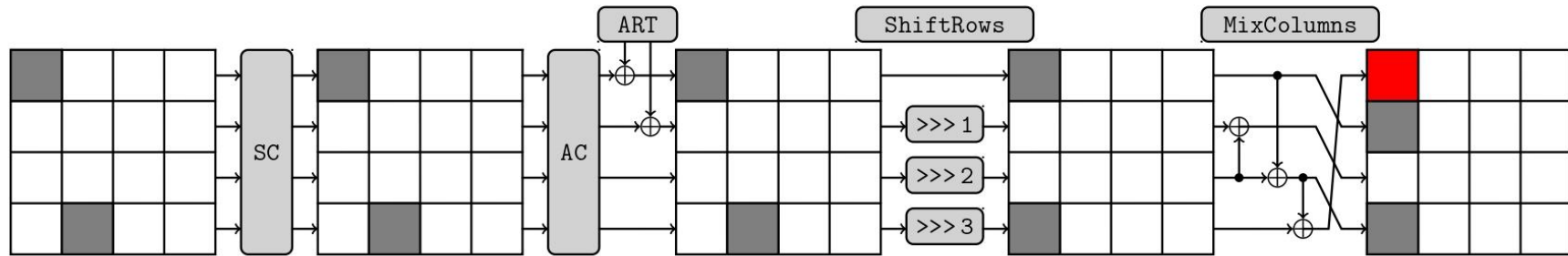
- Binary variables abstracting the presence/absence of non-zero differences
- # active Sboxes \longrightarrow **upper bound** on the probability of differential characteristics
- XOR: two non-zero values can lead to the presence **or** the absence of a difference

Truncated Differential Characteristics on SKINNY



- Binary variables abstracting the presence/absence of non-zero differences
- # active Sboxes \longrightarrow **upper bound** on the probability of differential characteristics
- XOR: two non-zero values can lead to the presence **or** the absence of a difference

Truncated Differential Characteristics on SKINNY



- Binary variables abstracting the presence/absence of non-zero differences
- # active Sboxes \longrightarrow **upper bound** on the probability of differential characteristics
- XOR: two non-zero values can lead to the presence **or** the absence of a difference

Search for Minimal Truncated Characteristics

```
foreach state  $s$  do  $M[s] \leftarrow$  list of states  $s'$  reachable from  $s$  through one round
foreach state  $s$  do  $C[0][s] \leftarrow$  number of active cells of  $s$ 
for  $1 \leq r < R$  do
  | foreach state  $s$  do  $C[r][s] \leftarrow \infty$ 
  | foreach state  $s$  do
  | | foreach state  $s' \in M[s]$  do
  | | |  $c \leftarrow C[r-1][s] +$  number of active cells of  $s'$ 
  | | | if  $c < C[r][s']$  then  $C[r][s'] \leftarrow c$ 
  | | end
  | end
end
return  $C$ 
```

- Dynamic programming
- Complexity: $(R - 1) \times 2^{20}$

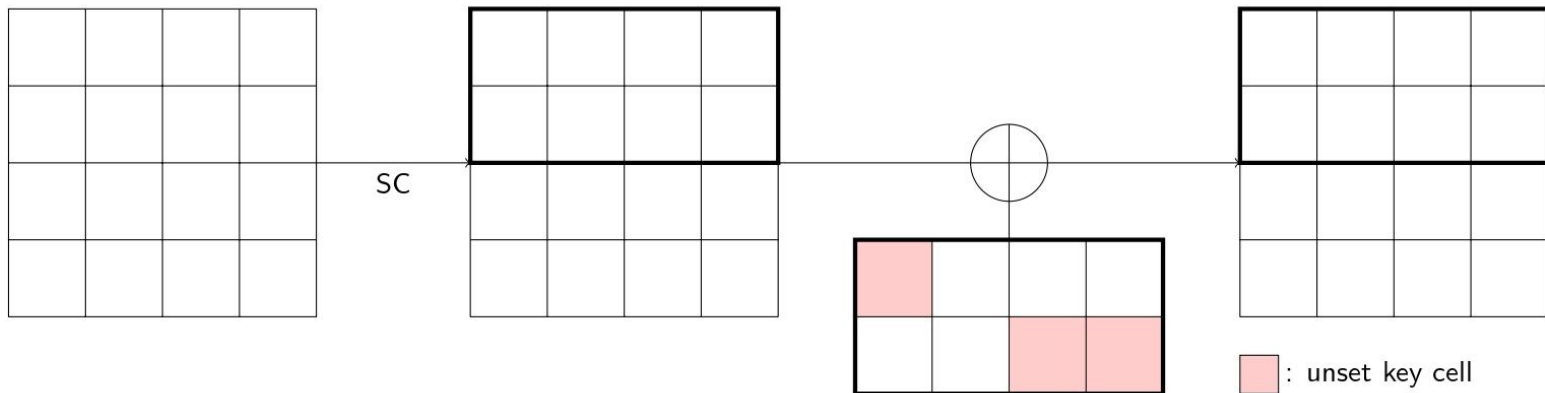
Differences in the key

```
foreach state  $s$ , key  $k$  do  $M[s, k] \leftarrow$  list of  $(s', k')$  reachable from  $(s, k)$ 
foreach state  $s$ , key  $k$  do  $C[0][s, k] \leftarrow$  number of active cells of  $s$ 
for  $1 \leq r < R$  do
  foreach state  $s$ , key  $k$  do  $C[r][s, k] \leftarrow \infty$ 
  foreach state  $s$ , key  $k$  do
    foreach  $(s', k') \in M[s, k]$  do
       $c \leftarrow C[r-1][s, k] +$  number of active cells of  $s'$ 
      if  $c < C[r][s', k']$  then  $C[r][s', k'] \leftarrow c$ 
    end
  end
end
end
return  $C$ 
```

- **Problem:** $\# (s, k) = 2^{64}$ in the TK3 model
- Can be reduced to $2^{48} \rightarrow$ still unpractical

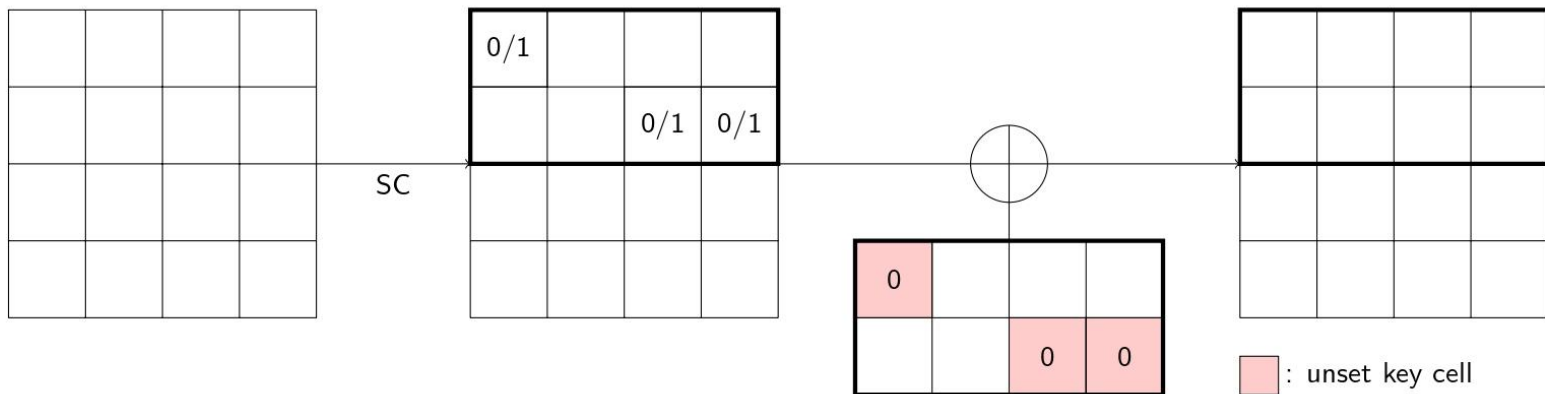
Early Abort Technique

- **Conjecture:** Optimal truncated characteristics have **few active key cells**
- **Idea:** Build a binary search tree on active cells of the key
 - At step i , decide whether cell i of the master key is active or not
 - Run a **degraded search** for minimal number of active Sboxes
 - Cut branches which cannot reach the current bound

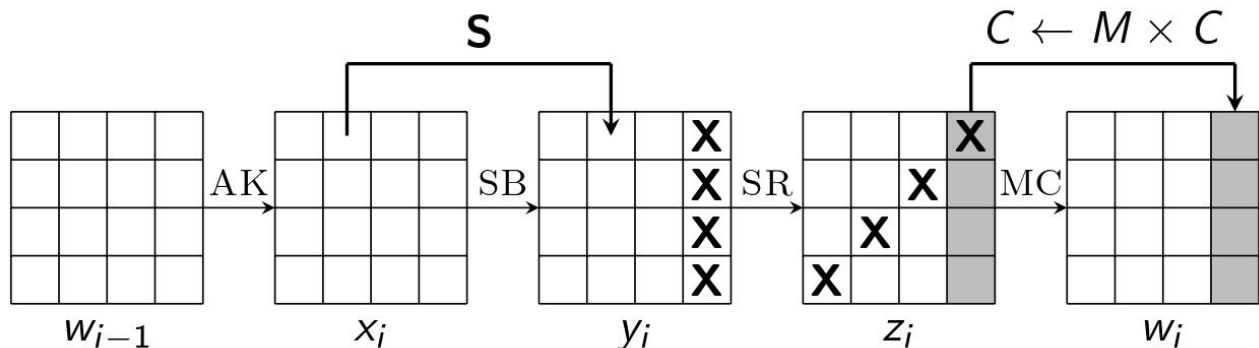


Early Abort Technique

- **Conjecture:** Optimal truncated characteristics have **few active key cells**
- **Idea:** Build a binary search tree on active cells of the key
 - At step i , decide whether cell i of the master key is active or not
 - Run a **degraded search** for minimal number of active Sboxes
 - Cut branches which cannot reach the current bound

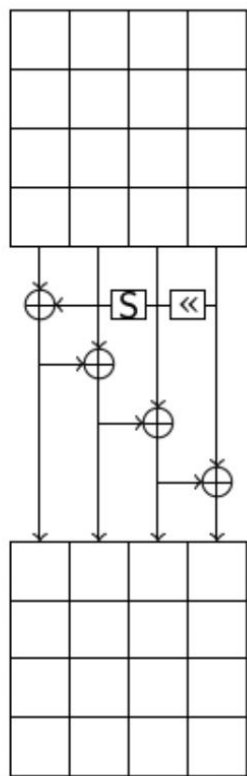


AES

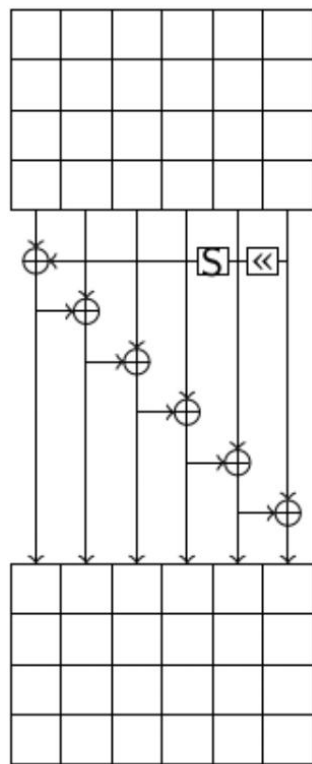


- Standardized in 2001 for 3 key lengths: 128, 192 and 256 bits
- Block size of 128 bits : 4×4 matrix of bytes
- An AES round applies $MC \circ SR \circ SB \circ AK$ to the state
- No MixColumns in the last round

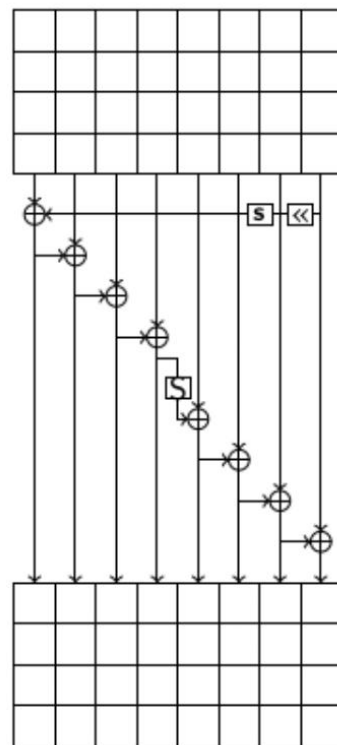
AES Key-Schedules



(a) AES-128



(b) AES-192

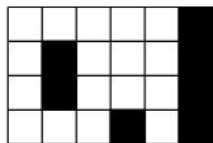


(c) AES-256

Compress more

If the state size is still too big then **compress** it ...

Truncated difference



$ K $	128	192	256
#	2^{32}	2^{40} X	2^{48} X



Compressed difference



$ K $	128	192	256
#	$2^{18.58}$	$2^{23.22}$	$2^{27.86}$

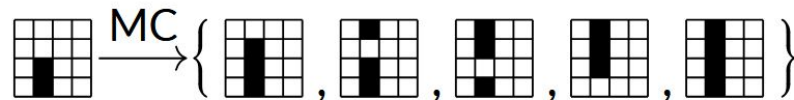
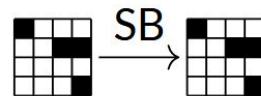
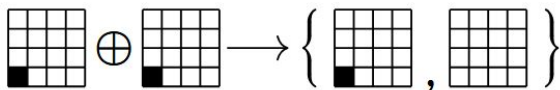
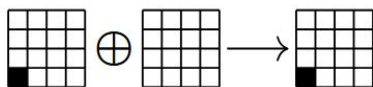
... but this comes at the price of:

- more **complicated** transition rules
- more **invalid** configurations


Example: AES

Basic propagation rules ...

XOR of two bytes



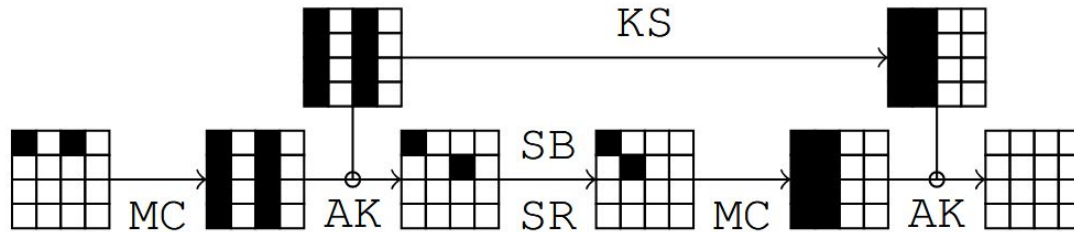
... do not necessarily lead to valid truncated trails.

Ex:  is not instantiable.

Example of linear incompatibility in the case of AES-128

The linearity of the KS imposes all the active columns $[a, b, c, d]^T$ to be equal, which contradicts the first key addition (AK)

$$\mathbf{M} \cdot [x, 0, 0, 0]^T \oplus [x', 0, 0, 0]^T = \mathbf{M} \cdot [y, 0, 0, 0]^T \oplus [0, y', 0, 0]^T .$$



Linear equations \rightsquigarrow Detect inconsistencies of the form $\blacksquare = \sum \square$




Main process

Searching for the best differential characteristic:




1. Search for the **best** truncated/compressed characteristic \leftarrow upper-bound
2. Look for structural inconsistencies, if any go back to 1
3. Find the best **instanciation** and save it \leftarrow lower-bound
4. If upper-bound \neq lower-bound, go back to 1

Generic Solvers




MILP

-  **Constraints:** Linear
-  **Variables:** Integer/real
-  **Optimize** a linear objective

SAT/SMT

-  **Constraints:** CNF
-  **Variables:** Boolean
-  **Find** a satisfiable assignment

CP

-  **Constraints:** Various
-  **Variables:** Integer, set,...
-  **Find** a satisfiable assignment or **optimize** an objective

Mixed-Integer Linear Programming (MILP)

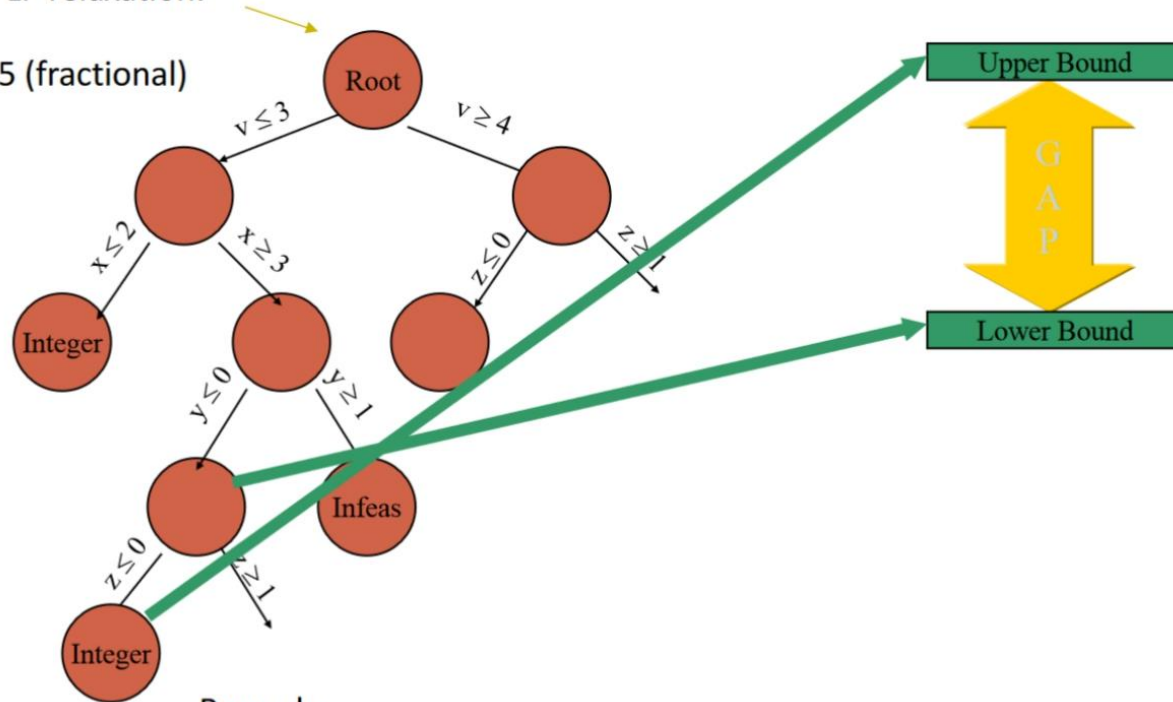
Objectif	$c_1x_1 + \cdots + c_nx_n$	$c \cdot x$
Constraints	$a_{1,1}x_1 + \cdots + a_{1,n}x_n \leq b_1$ $a_{2,1}x_1 + \cdots + a_{2,n}x_n \leq b_2$ \vdots $a_{m,1}x_1 + \cdots + a_{m,n}x_n \leq b_m$	$A \cdot x \leq b$
Domain	$x_1, \dots, x_d \in \mathbb{Z}, \quad x_{d+1}, \dots, x_n \in \mathbb{R}$ <hr/> $x_1, \dots, x_n \in \{0, 1\}$	

- Objective function and all constraints are **linear**.
- Some variables are **integers**, some variables are **continuous**.
- Typically in our applications, almost all variables are **Boolean**.

MIP Solution Framework

Solve LP relaxation:

$v=3.5$ (fractional)



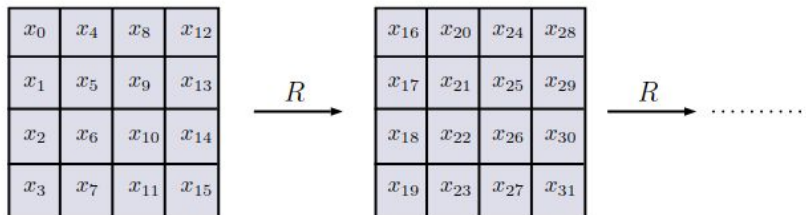
Remarks:

- (1) $GAP = 0 \Rightarrow$ Proof of optimality
- (2) In practice: Often good enough to have good Solution

First use of MILP in Cryptography

In 2011, Mouha et al. and Wu and Wang proposed to use MILP for finding the minimum number of differentially and linearly active Sboxes.

Example AES



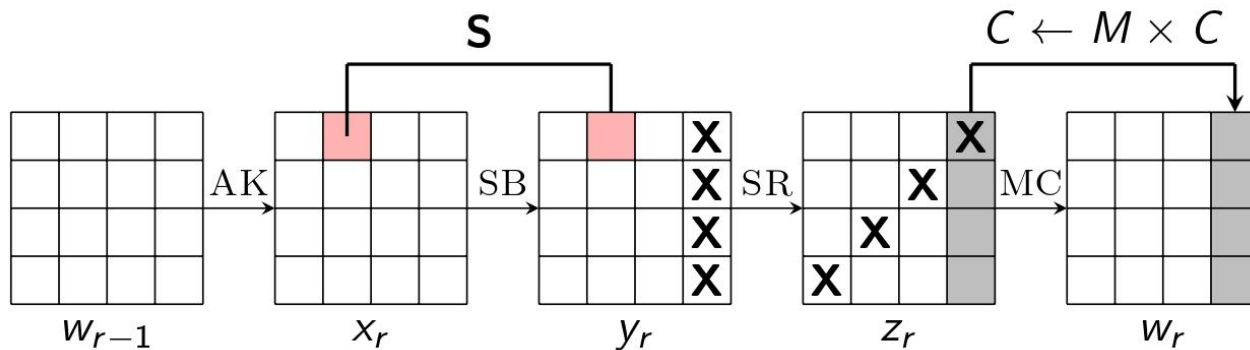
Define $16r$ variables $x_i \in \{0, 1\}$:

- $x_i = 1$ has a non-zero difference (active)
- $x_i = 0$ is (inactive)

Write propagation rules as linear inequalities.

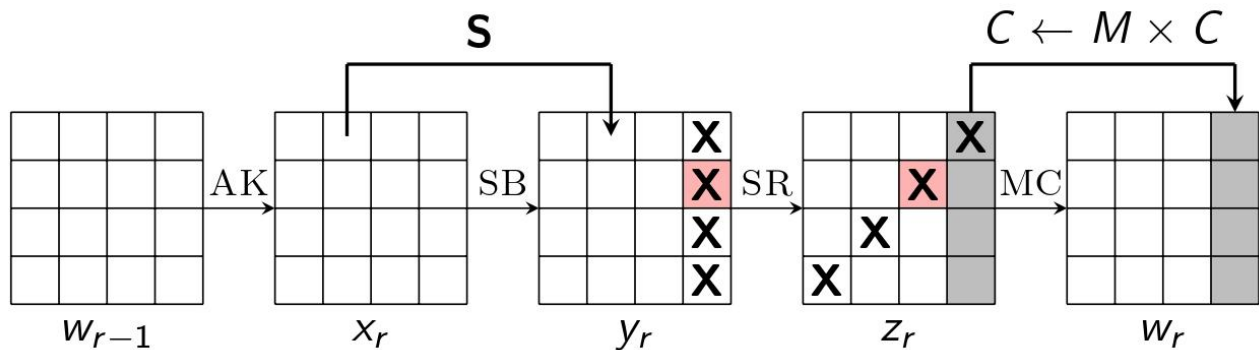
Objective function: Minimize $\sum x_i$.

Application to AES



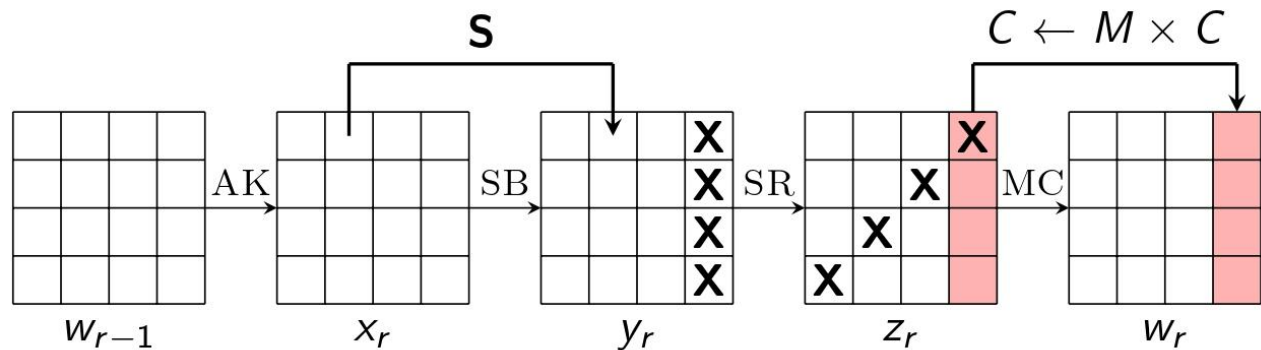
- $x_r[i] = y_r[i]$

Application to AES



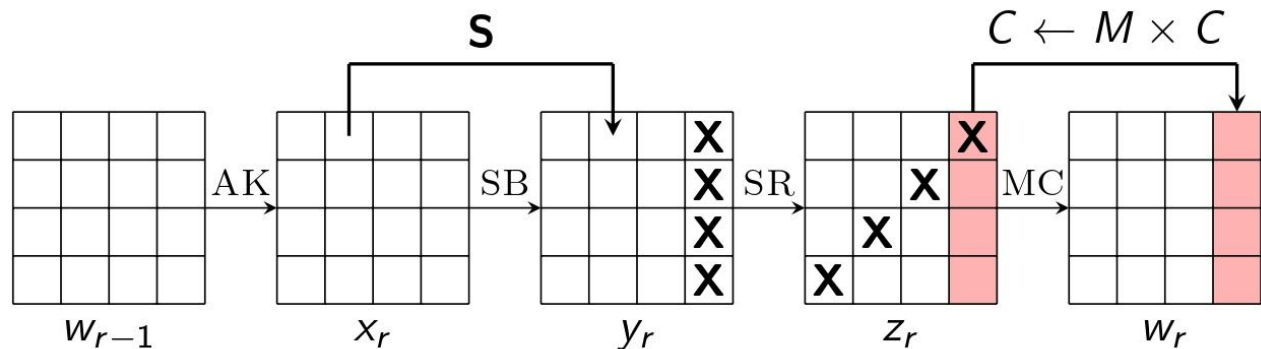
- $x_r[i] = y_r[i]$, $y_r[i] = z_r[\text{SR}[i]]$

Application to AES



- $x_r[i] = y_r[i]$, $y_r[i] = z_r[\text{SR}[i]]$
- $\sum_{i \in C} z_r[i] + w_r[i] = 0 \text{ or } \geq 5$

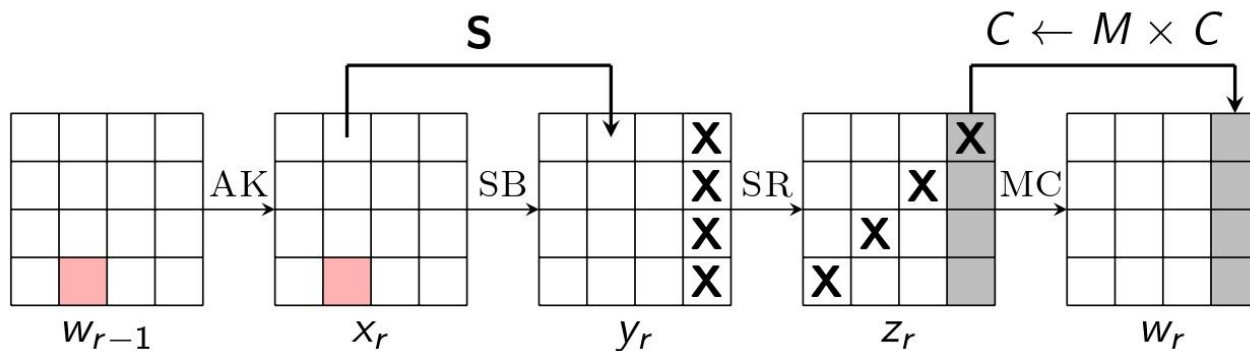
Application to AES



- $x_r[i] = y_r[i]$, $y_r[i] = z_r[\text{SR}[i]]$
- $\sum_{i \in C} z_r[i] + w_r[i] = 0$ or ≥ 5
- Introduce an **extra binary variable** e

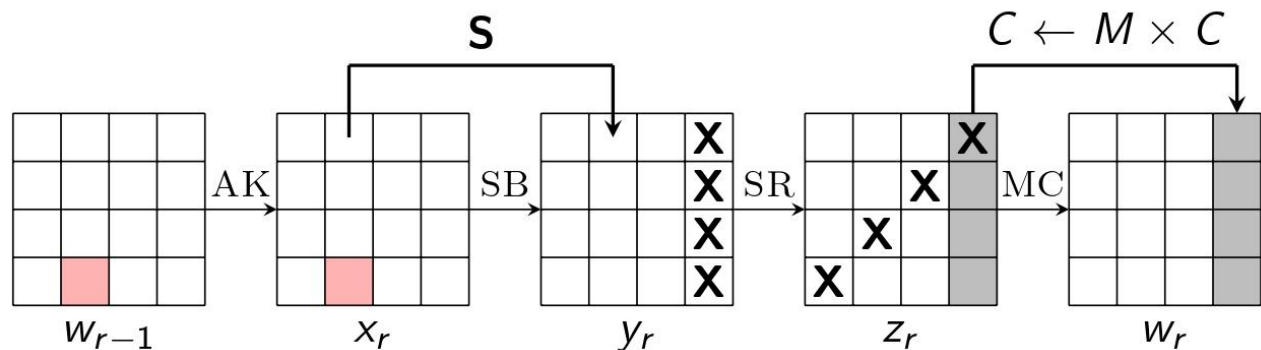
$$\sum_{i \in C} z_r[i] + w_r[i] \geq 5e \text{ and } \sum_{i \in C} z_r[i] + w_r[i] \leq 8e$$

Application to AES



- No difference in key: $w_{r-1}[i] = x_r[i]$

Application to AES



- **No difference in key:** $w_{r-1}[i] = x_r[i]$
- **Difference in key:** $w_{r-1}[i] + k_r[i] + x_r[i] \neq 1$

$$\begin{cases} 1 - w_{r-1}[i] + k_r[i] + x_r[i] \geq 1 \\ w_{r-1}[i] + 1 - k_r[i] + x_r[i] \geq 1 \\ w_{r-1}[i] + k_r[i] + 1 - x_r[i] \geq 1 \end{cases}$$

Bitwise vs Wordwise

Wordwise model

- One variable per word of the state (byte or nibble)
- The properties of the Sbox do not influence the propagation (only activeness counts)
- Only the **branch number** of the linear layer counts.
- Few variables, small system of inequalities
- Not that precise.

Bitwise model

- Binary variables are assigned to each bit of the state.
- Propagation through linear layer becomes **precise** and simple to write down. However, **too many inequalities** may be needed.
- Propagation rules through Sbox **complicated**.

Comparison of Tools

R	MILP				MiniZinc/SAT				Ad-Hoc				Choco			
	SK	TK1	TK2	TK3	SK	TK1	TK2	TK3	SK	TK1	TK2	TK3	SK	TK1	TK2	TK3
7	1s	1s	1s	1s	17s	8s	1s	1s	1s	21s	22s	22s	7s	7s	1s	1s
8	1s	1s	1s	1s	140s	7s	4s	2s	1s	22s	31s	23s	7s	8s	3s	1s
9	2s	2s	2s	2s	57s	11s	7s	1s	1s	22s	24s	26s	8s	9s	7s	1s
10	7s	5s	3s	2s	97s	46s	15s	10s	1s	22s	24s	27s	9s	60s	55s	2s
11	8s	11s	4s	3s	312s	29m	22s	24s	1s	23s	25s	32s	23s	188m	86s	34s
12	13s	35s	7s	3s	468s	> 24h	113s	35s	1s	24s	27s	25s	75s	> 24h	43m	288s
13	9s	53s	17s	6s	14m		14m	104s	1s	24s	30s	27s	249s		> 24h	56m
14	23s	93s	27s	8s	491s		72m	148s	1s	24s	39s	28s	10m			> 24h
15	69s	245s	75s	21s	27m		> 24h	157m	1s	25s	46s	34s	85m			
16	12m	423s	148s	39s	128m			251m	1s	25s	57s	38s	> 24h			
17	46m	22m	213s	53s	106m			> 24h	1s	27s	59s	48s				
18	178m	31m	535s	64s	403m				1s	27s	76s	73s				
19	529m	56m	29m	218s	436m				1s	28s	110s	283s				
20	16h	87m	33m	340s	174m				1s	28s	193s	326s				

Algorithm	R	Min nb of active S-boxes	# char.	CP [RGMS22] Time	MILP Real Time (User Time)	Dynam. Prog. Real Time (User Time)
AES-128	3	5	2	13s	1s (1s)	1s (1s)
	4	12	1	31s	9s (36s)	1s (1s)
	5	17	81	2h24m	26s (2m22s)	40s (5m6s)
AES-192	3	1	14	1s	1s (1s)	1s (2s)
	4	4	3	6s	2s (3s)	1s (4s)
	5	5	2	8s	1s (3s)	1s (5s)
	6	10	3	17s	10s (34s)	1s (8s)
	7	14	2	46s	1m (4m26s)	1s (9s)
	8	18	4	1m23s	1m38s (8m3s)	1m35s (12m37s)
	9	24	6	30m	5m33s (35m18s)	4d5h (20d4h)
AES-256	3	1	33	1s	1s (1s)	8s (46s)
	4	3	10	3s	1s (1s)	12s (1m10s)
	5	3	4	5s	1s (2s)	16s (1m39s)
	6	5	3	13s	3s (5s)	19s (1m57s)
	7	5	1	18s	3s (5s)	23s (2m21s)
	8	10	2	32s	8s (24s)	29s (3m1s)
	9	15	8	5m46s	23s (1m31s)	32s (3m24s)
	10	16	4	2m39s	2m19s (8m59s)	34s (3m31s)
	11	20	4	5m30s	3m20s (15m35s)	42s (4m30s)
	12	20	4	4m37s	6m31s (37m24s)	42s (4m16s)
	13	24	4	7m	23m16 (160m58s)	52s (5m24s)
	14	24	4	9m17s	32m27s (124m28s)	50s (5m5s)

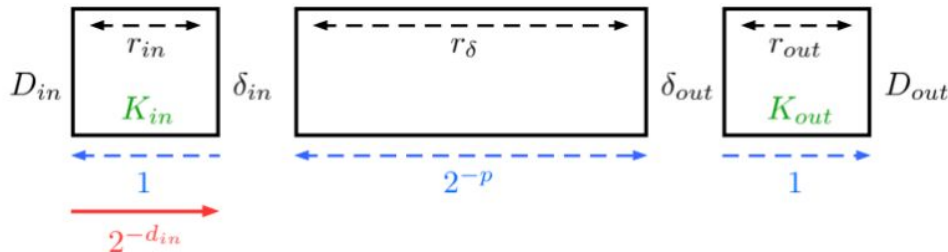
- Use the **right** tools and modelizations:
 - Dedicated algorithms
 - Generic solvers: CP, SAT/SMT, MILP

Key-recovery attacks

A differential distinguisher can be used to mount a **key recovery** attack.

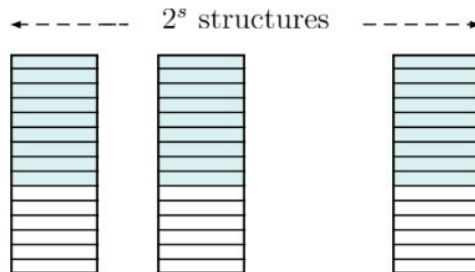
- This technique broke many of the cryptosystems of the 70s-80s, e.g. **DES**, **FEAL**, **Snefru**, **Khafre**, **REDOC-II**, **LOKI**, etc.
- New primitives should come with arguments of resistance **by design** against this technique.
- Most of the arguments used rely on showing that **differential distinguishers of high probability do not exist** after a certain number of rounds.
- Not always enough: A **deep understanding of how the key recovery works** is necessary to claim resistance against these attacks.

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 $2^{d_{in}}$
 $\Rightarrow 2^{2d_{in}-1}$ pairs from a structure.
- As $2^{s+2d_{in}-1} = 2^{p+d_{in}} \Rightarrow s = p - d_{in} + 1$ structures.

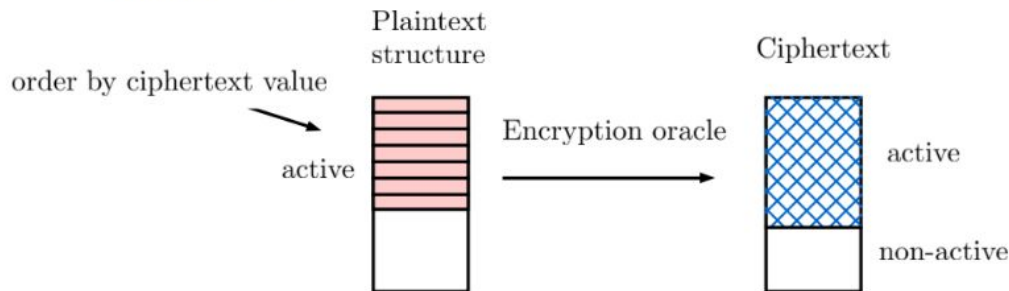


Data complexity: 2^{p+1} , **Memory complexity:** $2^{d_{in}}$

Not all pairs are useful

Idea: Discard pairs that will not follow the differential.

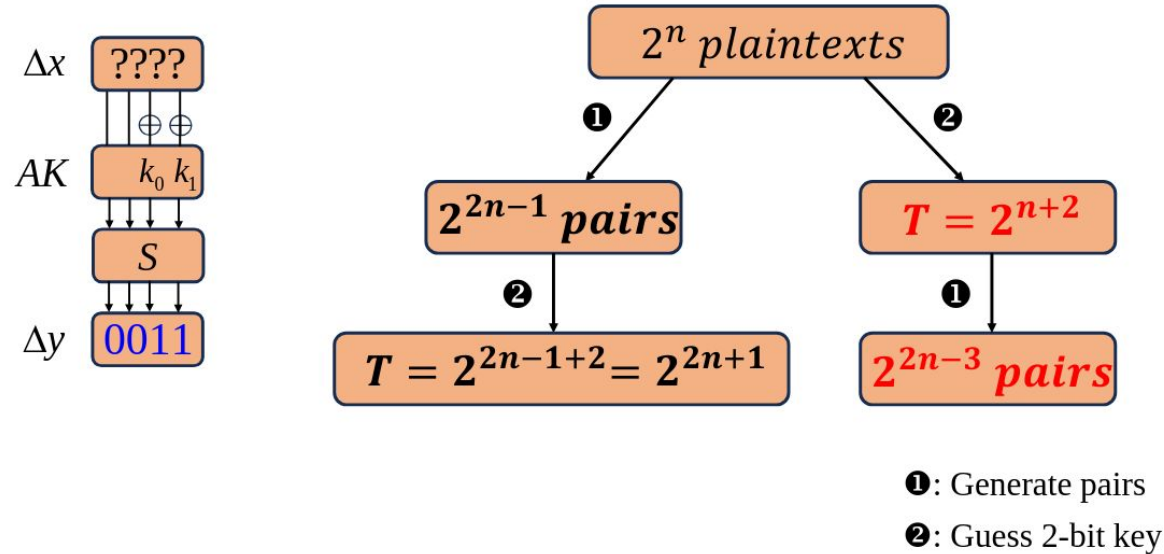
- Keep only those plaintext pairs for which the difference of the corresponding output pairs belongs to D_{out} .
- Order the list of structures with respect to the values of the non-active bits in the ciphertext.



Number of pairs for the attack

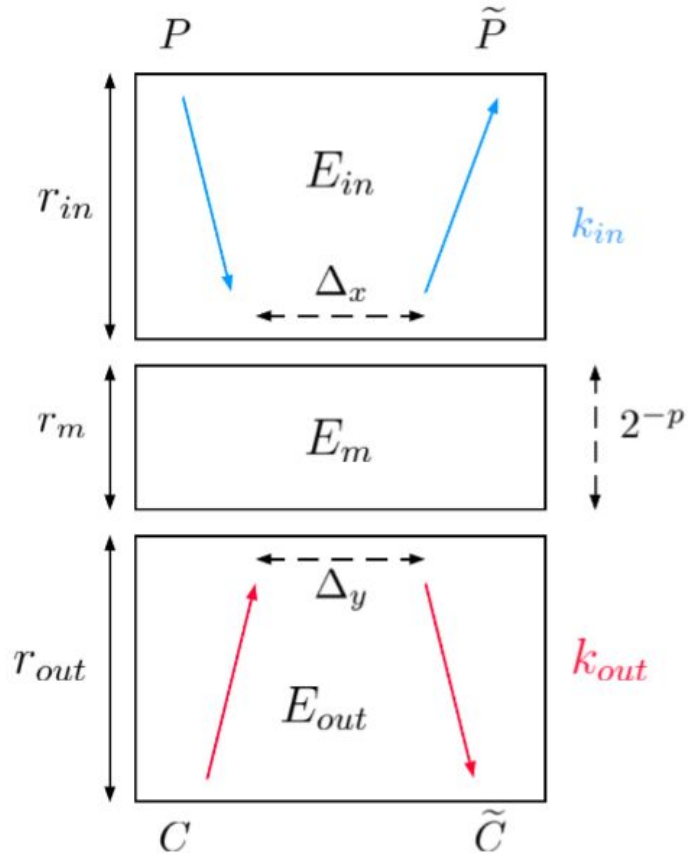
$$N = 2^{p+d_{in}-(n-d_{out})}.$$

Alternative 1: Early key-guessing



- ▶ lower time complexity ✓
- ▶ reduce the number of pairs ✓

Alternative 2: Differential-MitM attacks



Procedure:

1. Ask for one plaintext/ciphertext pair (P, C)
2. Construct the set of the $|k_{in}|$ possible plaintexts \mathcal{P}
3. Construct the set of the $|k_{out}|$ possible ciphertexts \mathcal{C}
4. Search for valid $(P', C') \in \mathcal{P} \times \mathcal{C}$ by looking for a collision

Pro:

- Much **easier** to deal with the key
- **Specific** improvement for ciphers with partial key addition

Con:

- More **memory** than for classical differential attacks

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**.

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.

2 cases:
1) Filter
2) Rank

- **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**.

Which one is the right key?

Distinguishing between two binomial probabilities requires around:

$$N \approx \frac{\left(z_{\alpha/2}\sqrt{p_1(1-p_1)} + z_{\beta}\sqrt{p_2(1-p_2)}\right)^2}{(p_1 - p_2)^2} \text{ samples}$$

where $z_{\alpha/2}$ and z_{β} are constants related to the probability of success.

Which one is the right key?

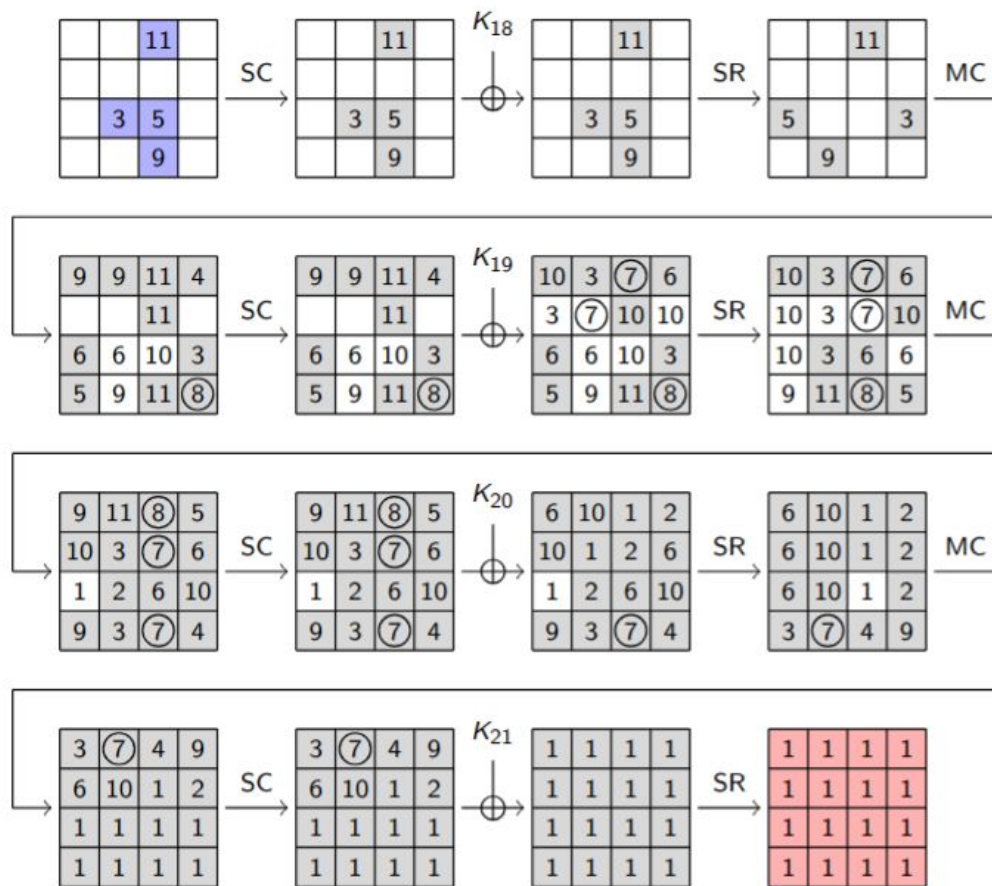
Let replace p_1 by p and p_2 by $p + \epsilon$:

$$N \approx \frac{\left(z_{\alpha/2} \sqrt{p(1-p)} + z_{\beta} \sqrt{(p+\epsilon)(1-p-\epsilon)} \right)^2}{\epsilon^2} \text{ samples}$$

In typical situations, both p and $p + \epsilon \ll 1$ and thus $N \approx \frac{(z_{\alpha/2} + z_{\beta})^2 p}{\epsilon^2}$ samples

In many attacks, both $p + \epsilon \approx 1/2^n \ll p$ and thus

$$N \approx \frac{(z_{\alpha/2} + z_{\beta})^2}{p} \text{ samples}$$



- Early abort technique
- Rebound-like procedure
- Knowing both **input/output differences** around an Sbox leads to the **actual values**
- Might be **very complex** depending on the key schedule and the cipher

Summary

- Searching for good differentials is **hard**
- Searching for good characteristics is **easy**
 - **in some situations only!**
 - look at the **quasidifferential** framework for more advanced estimations
 - many designs are still hard to analyze
- Searching for the best differential attacks is **hard**
 - **the best differential distinguisher does not necessary lead to the best attack**
 - look at **key absorption** for more advanced key recovery processes

Thank you for your attention!