# Lightweight Cryptography

Gregor Leander[1]

DTU Mathematics, Denmark

Finse 1222
May 2012

[1] Thanks to Matt Robshaw, Orange Labs, for some of the slides

# Outline

# Outline

# Upcoming IT-Landscape


Figure: Upcoming IT-Landscape

# More Precisely: RFID-Tags

RFID Tag

RFID=Radio-Frequency IDentification

# Example I


Electronic Passports

# Example II


Logistik

# Example III


Library in Copenhagen

# Example IV

School in Texas

# Example V


Bar in Spain

# Security

**Question**

Do we want this?

# Security

**Question**

Do we want this?

If we want it, we want it secure!

# RFID Controversy

# Types of RFID-Tags

## UHF Tags

These are small, cheap, communicating devices

- No internal power source
- Operational range of 4-8 m
- Cost $\approx$ 0.15 USD

# Types of RFID-Tags

Very different to HF devices

## HF Tags

- Much shorter operational range and more power
- Larger and more expensive
- Standard security possible.

# UHF-Tags Today

What do UHF-tags (currently) do?

- The basic UHF tag identifies itself at a distance
- This gives opportunities for "track-and-trace" of goods

# UHF-Tags Tomorrow

## Anticounterfeiting

There is demand to use UHF RFID tags as part of an anticounterfeiting solution

11% of global pharmaceutical commerce is counterfeit (39 billion USD)

## Enhanced Functionality

We need to show that the tag is authentic.

# UHF-Tags Tomorrow

## Enhanced Functionality

We need to show that the tag is authentic.

Possibilities:

- Network Solution
- Cryptographic Solution

Cryptographic Solution

$$\Rightarrow$$

Lightweight Cryptography

# Lightweight Cryptography

What is (not) Lightweight Cryptography

- Cryptography tailored to (extremely) constrained devices
- Not intended for everything
- Not intended for extremely strong adversaries
- Not weak cryptography

Here we focus on symmetric cryptography

# Lightweight Cryptography

## Question

Why do we need Lightweight Crypto?

- Upcoming IT-Landscape is pervasive
- Many cheap devices
- (Extremely) constrained in
  - computational power
  - battery
  - memory

# Lightweight Cryptography

**Question**

What about standard algorithms?

- AES is great for almost everywhere
- Mainly designed for software
- It is too expensive for very small devices
- It protects data stronger than needed

# Lightweight Cryptography

Why not simply wait 18 month?

**Moore's Law**

Computational power doubles each 18 month.

# Lightweight Cryptography

Why not simply wait 18 month?

**Moore's Law**

Computational power doubles each 18 month.

**Moore's Law**

Devices become cheaper.

# Lightweight Cryptography

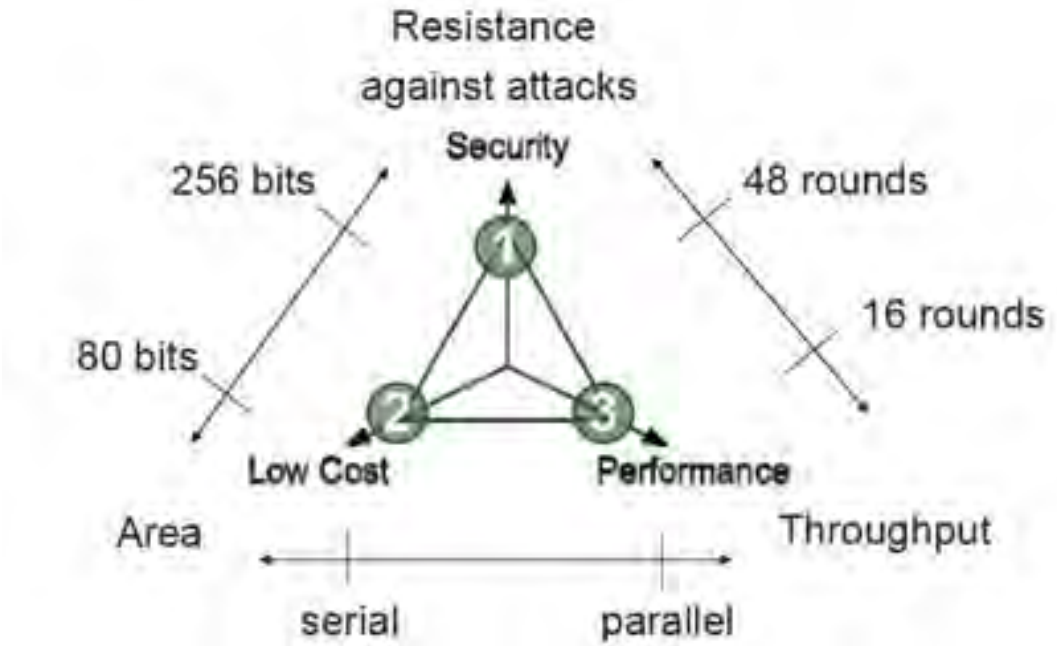



Figure: Tradeoffs between Security/Throughput/Area

# Lightweight Cryptography: Industry vs. Academia

## Industry

Non-existence of lightweight block ciphers a real problem since the 90's.

- Many proprietary solutions
- Often: not very good.

## Academia

Research on Lightweight block ciphers started only recently.

- Several proposals available.
- Still: some open questions.

# Outline

# Block Cipher- Short Intro

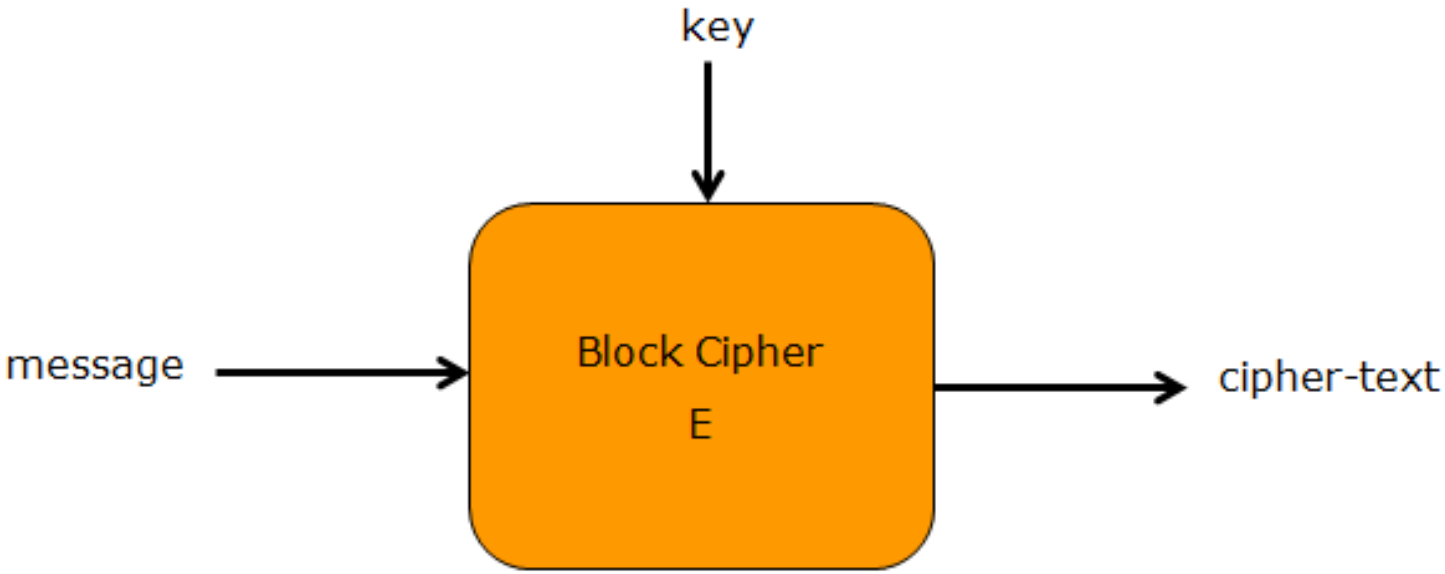



Figure: A Block Cipher

# Block Cipher- Short Intro

**Block Cipher**

The working horses in cryptography.

- Large fraction of secure communication based on block ciphers
- Well understood topic
- Very good block ciphers available
- Most prominent: AES

# AES

AES= Advanced Encryption Standard

- Developed by J. Daemen and V. Rijmen
- A NIST-standard since 2001
- Supersedes DES (Data Encryption Standard)

# AES - Advanced Encryption Standard

- US governmental encryption standard
- Keys: choice of 128-bit, 192-bit, and 256-bit keys
- Blocks: 128 bits
- Open (world) competition announced January 97
- 21 candidates submitted
- October 2000: AES=Rijndael
- Standard: FIPS 197, November 2001
- See www.nist.gov/aes for more on AES process

# AES=Rijndael

- Designed by Daemen and Rijmen, Belgium
- Simple design, only byte operations
- S-box, substitutes one byte by another byte
- Iterated cipher

| Key size | 128 | 192 | 256 |
|---|---|---|---|
| Number of rounds | 10 | 12 | 14 |

- Focus on 128-bit key version with 10 rounds/iterations

# AES - iterated cipher, key schedule

- Input: user selected key of 128 bits
- Output: 11 round keys $k_0, k_1, k_2, \ldots, k_{10}$
- $p = c_0$ plaintext
- $c_i = F(k_i, c_{i-1})$
- $c_{10}$ ciphertext
- Details of key-schedule are skipped here.

# AES round Transformation

## State is a Matrix

Arrange the 16 input bytes in a $4 \times 4$ matrix $(a)_i, j$ where

$$a_{i,j} \in \{0,1\}^8$$

| | | | |
|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

# AES round Transformation

| | | | |
|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

## Subfunctions

1. AddRoundKey
2. SubBytes (byte substitution via S-box)
3. ShiftRows
4. MixColumns

# AddRoundKey (bit-wise XOR)

## AddRoundKey

Byte-wise XOR with the round-key-matrix

$$b_{i,j} = a_{i,j} \oplus k_{i,j}$$

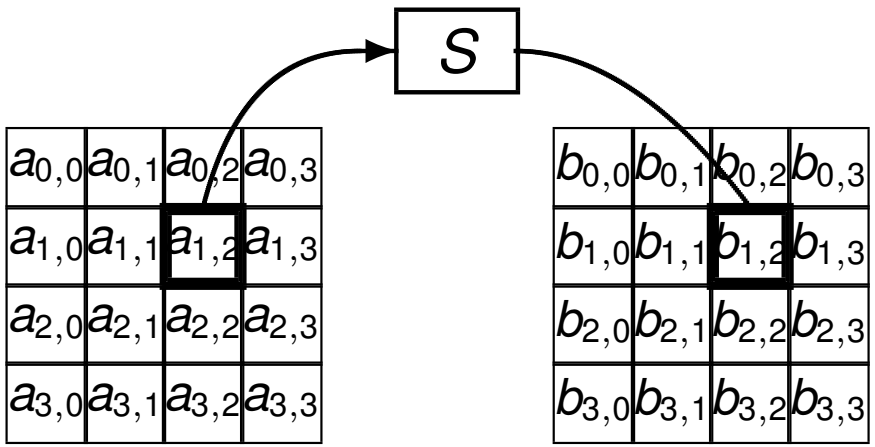
$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

# SubBytes

## SubBytes

Use one single invertible Sbox for all bytes and all rounds

$$\begin{aligned} S : \{0,1\}^8 &\rightarrow \{0,1\}^8 \\ b_{i,j} &= S(a_{i,j}) \end{aligned}$$



# The S-box in AES

$S = ($

```
 99, 124, 119, 123, 242, 107, 111, 197,  48,   1, 103,  43, 254, 215, 171, 118,
202, 130, 201, 125, 250,  89,  71, 240, 173, 212, 162, 175, 156, 164, 114, 192,
183, 253, 147,  38,  54,  63, 247, 204,  52, 165, 229, 241, 113, 216,  49,  21,
  4, 199,  35, 195,  24, 150,   5, 154,   7,  18, 128, 226, 235,  39, 178, 117,
  9, 131,  44,  26,  27, 110,  90, 160,  82,  59, 214, 179,  41, 227,  47, 132,
 83, 209,   0, 237,  32, 252, 177,  91, 106, 203, 190,  57,  74,  76,  88, 207,
208, 239, 170, 251,  67,  77,  51, 133,  69, 249,   2, 127,  80,  60, 159, 168,
 81, 163,  64, 143, 146, 157,  56, 245, 188, 182, 218,  33,  16, 255, 243, 210,
205,  12,  19, 236,  95, 151,  68,  23, 196, 167, 126,  61, 100,  93,  25, 115,
 96, 129,  79, 220,  34,  42, 144, 136,  70, 238, 184,  20, 222,  94,  11, 219,
224,  50,  58,  10,  73,   6,  36,  92, 194, 211, 172,  98, 145, 149, 228, 121,
231, 200,  55, 109, 141, 213,  78, 169, 108,  86, 244, 234, 101, 122, 174,   8,
186, 120,  37,  46,  28, 166, 180, 198, 232, 221, 116,  31,  75, 189, 139, 138,
112,  62, 181, 102,  72,   3, 246,  14,  97,  53,  87, 185, 134, 193,  29, 158,
225, 248, 152,  17, 105, 217, 142, 148, 155,  30, 135, 233, 206,  85,  40, 223,
140, 161, 137,  13, 191, 230,  66, 104,  65, 153,  45,  15, 176,  84, 187,  22
```

$)$

**Example**

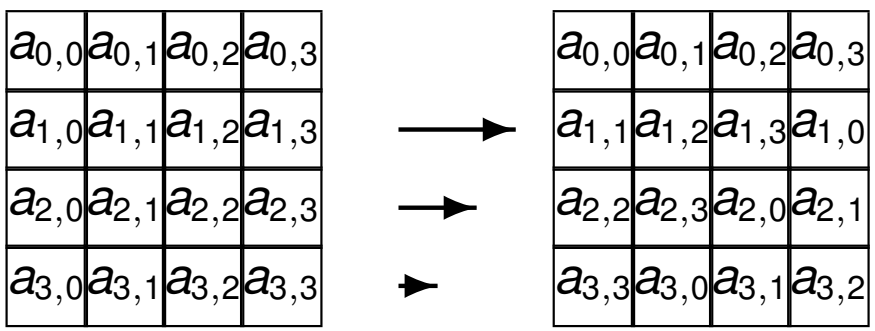
$S(0) = 99,\ S(1) = 124,\ \ldots,\ S(255) = 22.$

# ShiftRows

## ShiftRows

Shift the rows:

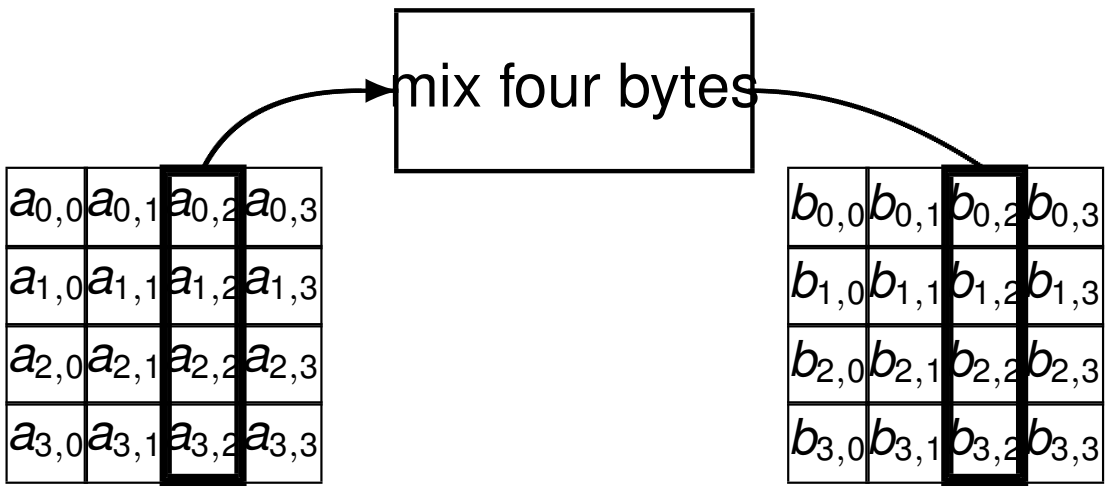
- The first by 0 positions
- The second by 1 positions
- The third by 2 positions
- The fourth by 3 positions



# MixColumns

## MixColumns

Each of four $b_{i,j}$ in a column depends on all four $a_{i,j}$ from same column.



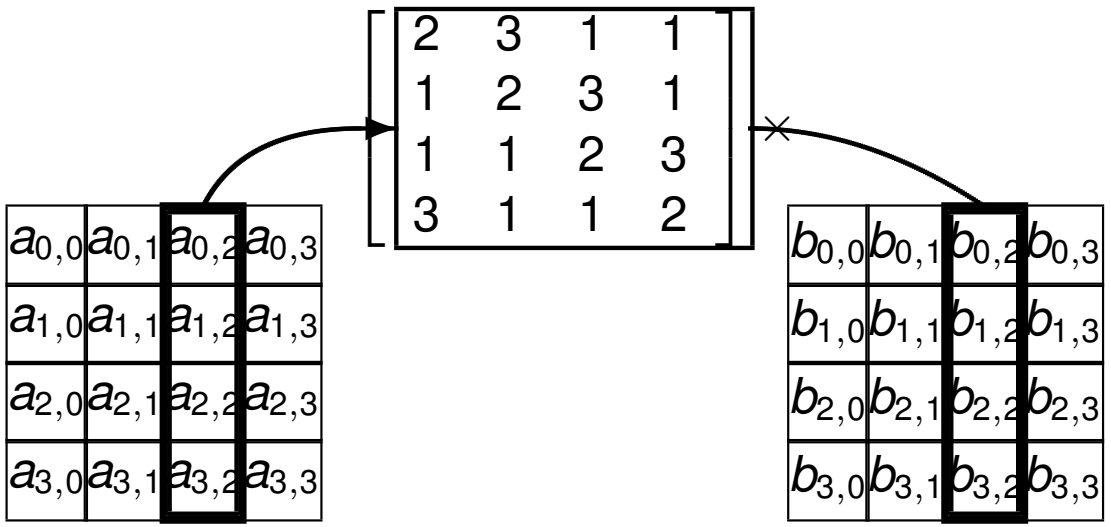
# MixColumns

## MixColumns

Bytes in columns are combined linearly, e.g.

$$b_{0,2} = \{2\} \times a_{0,2} + \{3\} \times a_{1,2} + \{1\} \times a_{2,2} + \{1\} \times a_{3,2}.$$

Multiplication is a special field-multiplication



# AES - 10-round version

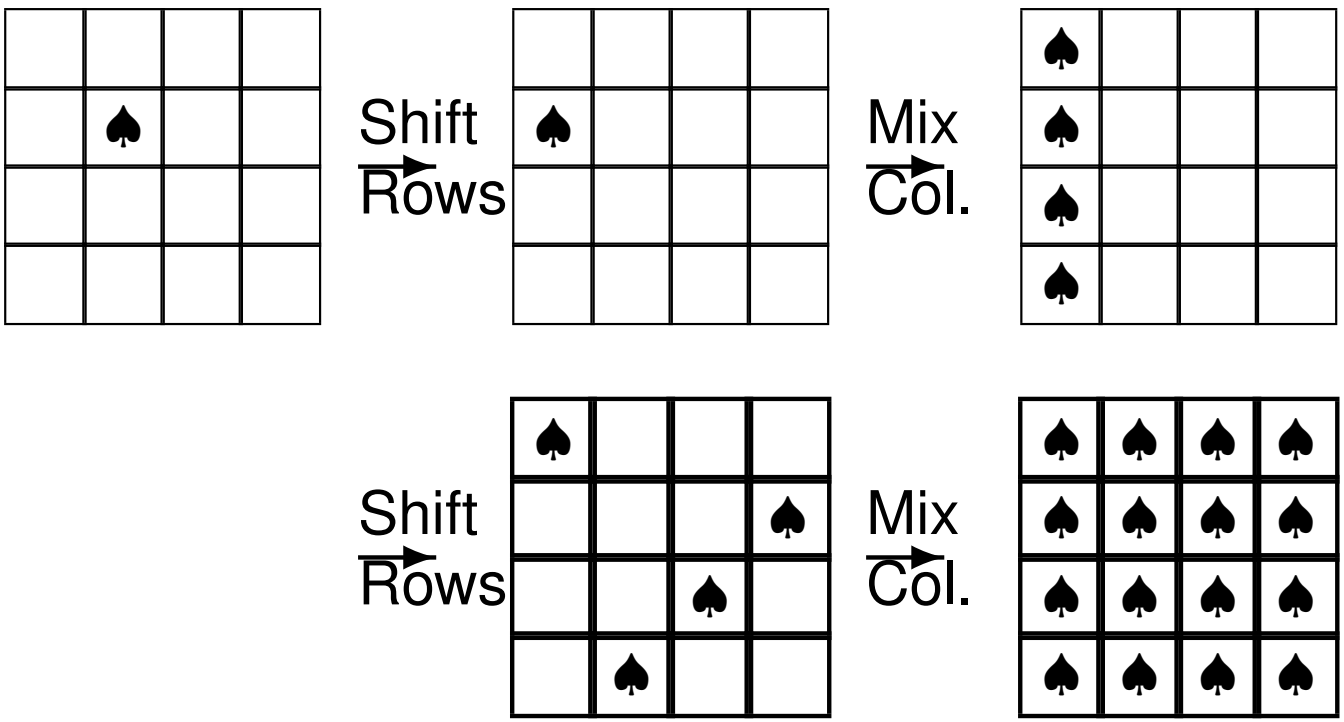
Arrange the 16 input bytes in a $4 \times 4$ matrix

- AddRoundKey
- Do nine times
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- SubBytes
- ShiftRows
- AddRoundKey

# Byte mixing in AES

## Byte Mixing

Each byte after two rounds of encryption depend on all 16 bytes in message.



# Bit mixing in AES

## Bit Mixing

The chosen S-box gives a very fast mixing of bits within each byte.

$$\begin{aligned} S : \mathbb{F}_{2^8} &\rightarrow \mathbb{F}_{2^8} \\ S(x) &= x^{-1} \end{aligned}$$

This mapping has very good cryptographic properties.

# Lightweight Ciphers in Real Life

## Algorithms Used In Real Products

- Keeloq
- DST
- DECT, C2, Mifare,...

What they have in common:

- efficient
- proprietary/not public (violates Kerckhoffs' principle)
- non standard designs
- not good

A lot more out there...

# Keeloq

**Keeloq**

A 32 bit block-cipher with a 64 bit key.

- Developed by Gideon Kuhn (around 1985).
- Sold for 10M$ to Microchip Technology Inc (1995).
- Algorithm for remote door openers: Cars, Garage, ...
- Used by: Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, Volkswagen Group,...

# Keeloq: Overview

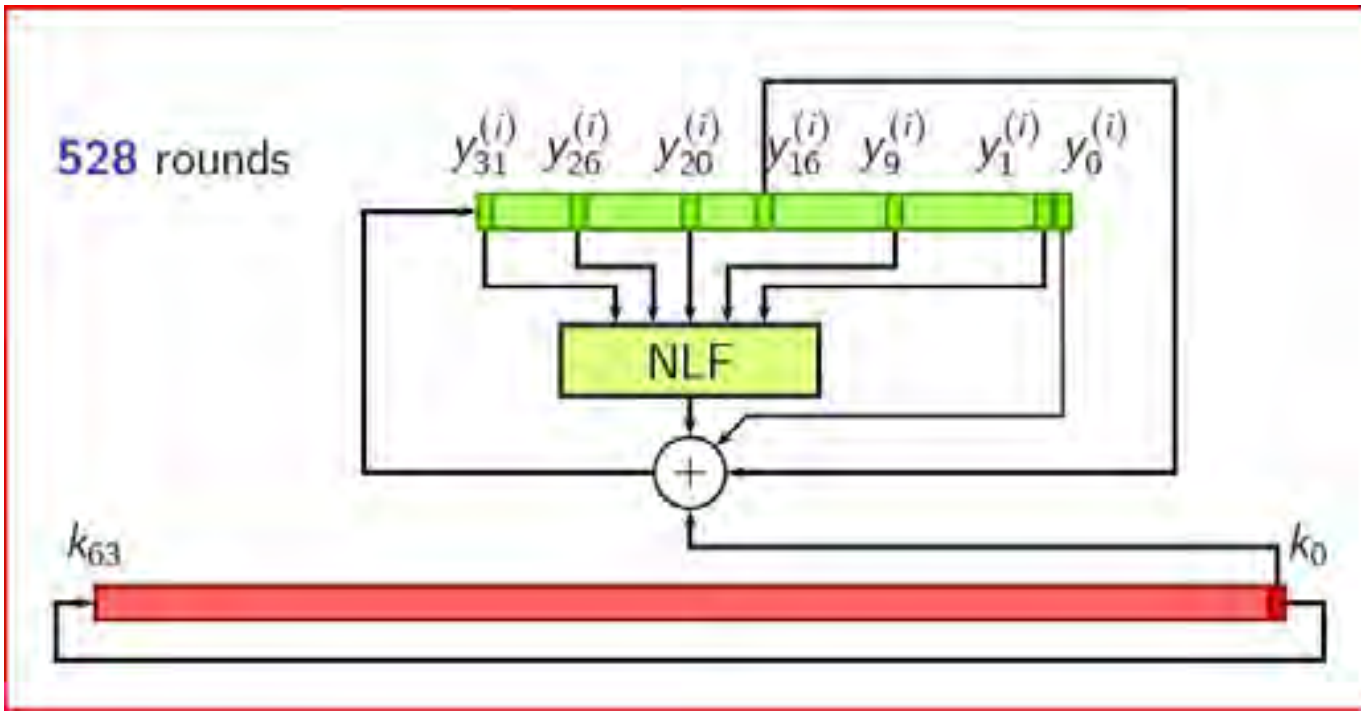



Figure: Overview of Keeloq

# Design Principles of Keeloq

**Keeloq**

Unbalanced Feistel-cipher.

- Many, very simple rounds.
- small block size.
- relatively small key.

# Attacks on Keeloq

Keeloq is broken (Biham, Dunkelman, Indesteege, Keller, Preneel 2008):

- Known plaintext: $2^{16}$ plain-text/cipher-text pairs and 500 CPU days of computation.
- Chosen plaintext: $2^{16}$ plain-text/cipher-text pairs and 200 CPU days of computation.

Main weakness here: Key-scheduling is periodic.

- Side-Channel attack (Eisenbarth, Kasper, Moradi, Paar, Salmasizadeh, Shalmani 2008): 10 encryptions, negligible computation.

Often: The master-key can be found.

**Summary**

Practical attacks with real consequences.

# DST

## DST

A 40 bit block cipher with a 40 bit key.

- Developed by Texas Instruments
- Used in Exxon-Mobil Speedpass payment system (approximately 7 million transponders)
- In vehicle immobilizer systems of Ford, Lincoln, Mercury, Toyota, Nissan.
- following Wikipedia: “one of the most widely-used unbalanced Feistel ciphers in existence”

# DST: Overview

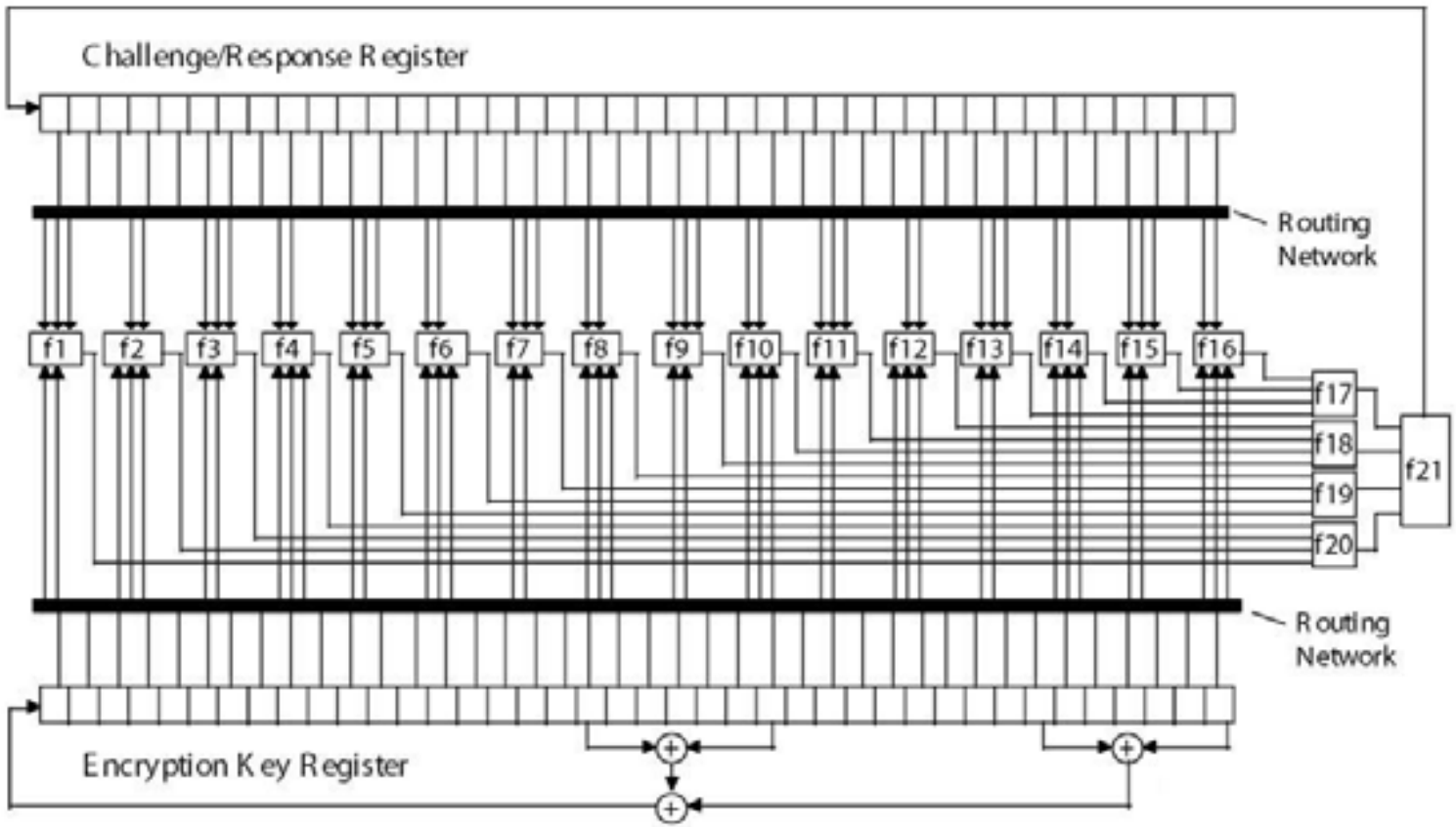



Figure: Overview of DST

# Design Principles of DST

**DST**

Unbalanced Feistel-cipher.

- Many, very simple rounds.
- small block size.
- very small key.
- non-standard key mixing.

# Attacks on DST

No attacks needed!

- Brute Force feasible.
- One a PC: Several weeks
- With specialized hardware (COPACOBANA 10kEUR): 9 min.

Main weakness here: small key

**Question**

Is the design sound?

**Summary**

Practical attacks with real consequences.

# C2

## C2

A 64 bit block cipher with 2048+56 bit key.

- Developed by 4C Entity (IBM, Intel, Panasonic, Toshiba)
- Algorithm for DVD-Audio, DVD-Video, SD-cards
- CPRM (Content Protection for Recordable Media)

# C2: Overview

- 10 Feistel rounds

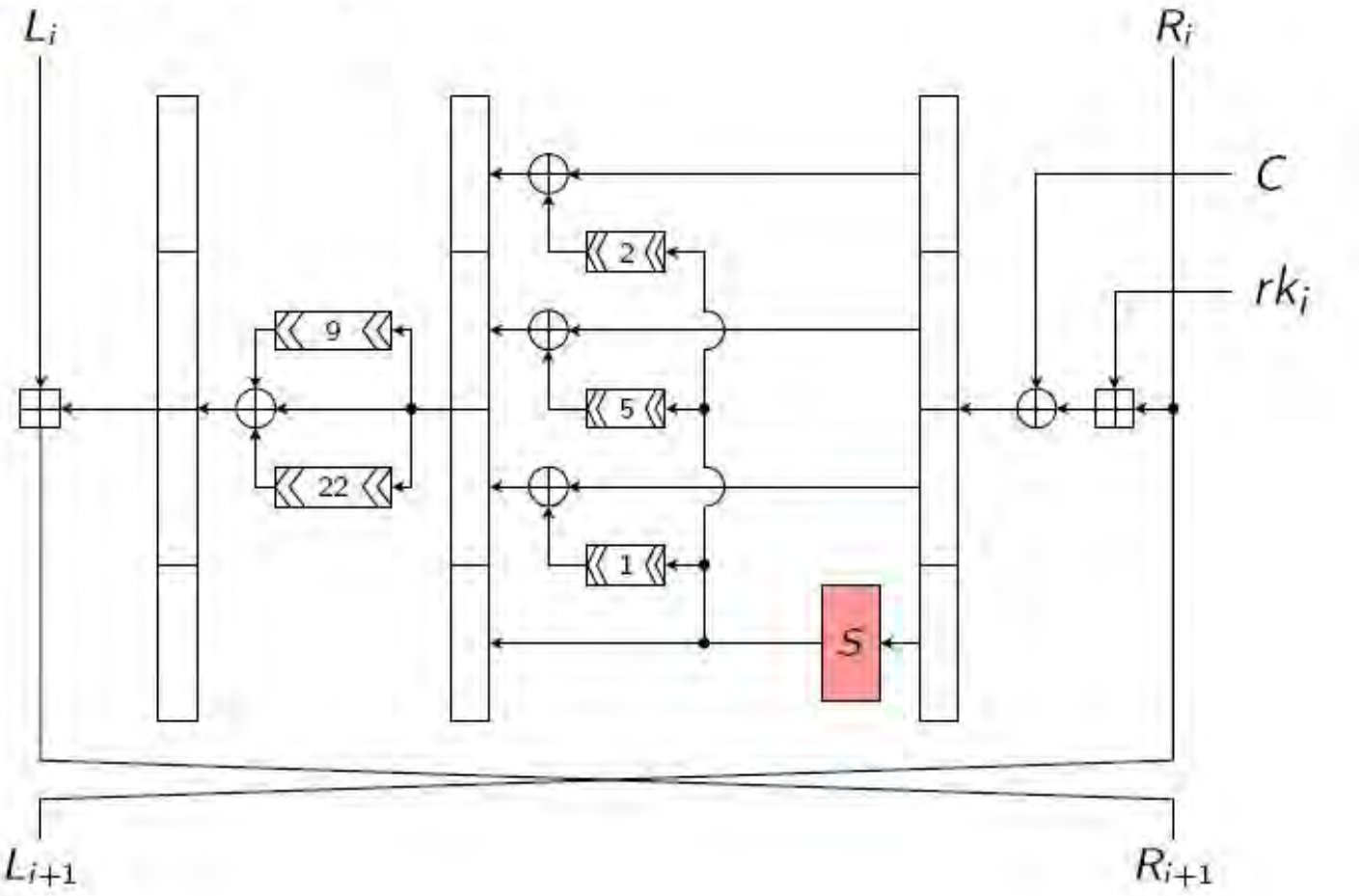



Figure: Overview of C2

# Design Principles of C2

- Optimized for small software footprint
- Few and simple rounds.
- relatively small block size.
- secret S-box

# Attacks on C2

Several Attacks on C2 known:

- Chosen key, chosen plain-text: S-box recovery with $2^{24}$ plaintext/ciphertext pairs, few minutes computation.
- Known Sbox, chosen plain-text: Key recovery with $2^{48}$ plaintext/ciphertext pairs and $2^{48}$ operations.
- Secret Sbox: Sbox and key-recovery with $2^{52}$ plaintext/ciphertext pairs and $2^{52}$ operations.

Main weakness:

- Differential properties
- Key-scheduling makes very limited use of S-box.

**Summary**

Might be practical attacks.

# Why?

**Question**

Why do they do that?

**Answer I**

They do not know better

**Answer II**

They have to.

Often a combination of both.

# How?

Some common design principles:

- Relative small block-size
- Relative small key-size
- Many simple rounds

**We can learn from that!**

We will rediscover (some of) those in the modern lightweight block ciphers

# How?

Some weaknesses:

- Overly simplified key scheduling
- Non-standard components

# How?

Some weaknesses:

- Overly simplified key scheduling
- Non-standard components

**We can learn from that!?**

We will rediscover (some of) those in the modern lightweight block ciphers

# Why?

**Question**

Why do they do that?

**Answer II**

They have to.

We need

- secure
- well analyzed
- public

ciphers for highly resource constrained devices.

# General Design Philosophy

## Guidelines/Goals

- Efficiency: Here mainly area
- Simplicity
- Security

# Design Considerations: Hardware

**Hardware**

What do things cost in hardware?

**Suggestion**

Make it an interdisciplinary project!

# Cost Overview

## Question

What should/should not be used?

Rule of Thumb:

- NOT: 0.5 GE
- NOR: 1 GE
- AND: 1.33 GE
- OR: 1.33
- XOR: 2.67

Registers/Flipflops: $6 - 12$ GE per bit!

# Design Decisions I

## Question

Block size/ Key size?

Storage (FF) is expensive in hardware.

- Block size of 128 is too much.
- We do not have to keep things secret forever.

## Decision

Relative Small Block Size: 32,48 or 64
Key size: 80 bit often enough

# Design Decisions

**Question**

Feistel vs. SP-Network?

# Feistel Cipher

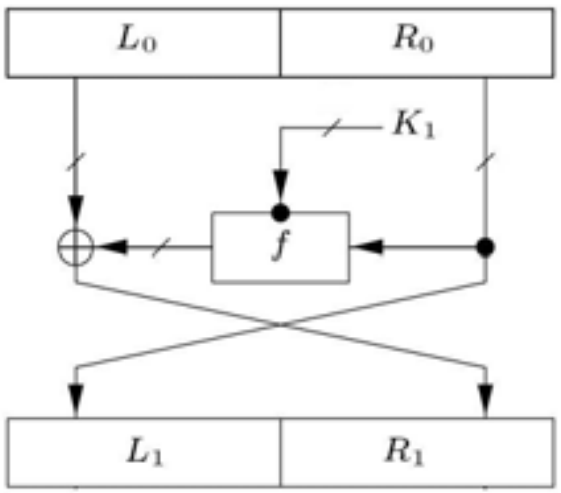



Figure: Feistel Cipher (DES)

# SP-Network

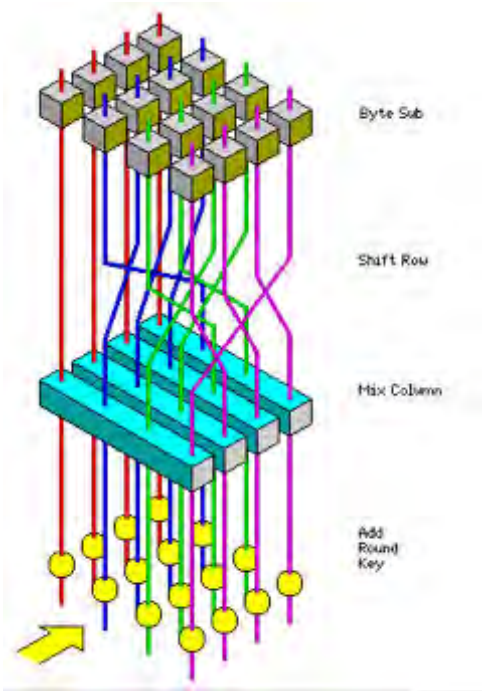



Figure: SP-Network (AES)

# Design Decisions

## Question

Feistel vs. SP-Network?

Pro-Feistel:

- Potentially Reduced complexity.
- (Strongly) unbalanced Feistel.
- Decryption can be almost free.

Pro-SP:

- Often: Encryption only.
- Less rounds/Easier to analyze?

## Decision

Both reasonable.

# SP-Network

## SP-Network

We have to design

- S-Layer
- P-Layer
- Key-scheduling

Here we focus on the S-Layer and the P-Layer.

# Design Issues

## Design Issues

The S-Layer has to maximize nonlinearity.
It has to be cheap.

The S-Layer consist of a number of Sboxes executed in parallel

$$S_i : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$$

In hardware realized as Boolean functions.

# Design Issues

**Question**

Different Sboxes vs. all Sboxes the same?

A serialized implementation becomes smaller if all Sboxes are the same.

**Decision**

Only one Sbox.

# Design Issues

**Question**

What size of Sbox?

In general: The bigger the Sbox the more expensive it is in hardware.

# Sbox Costs

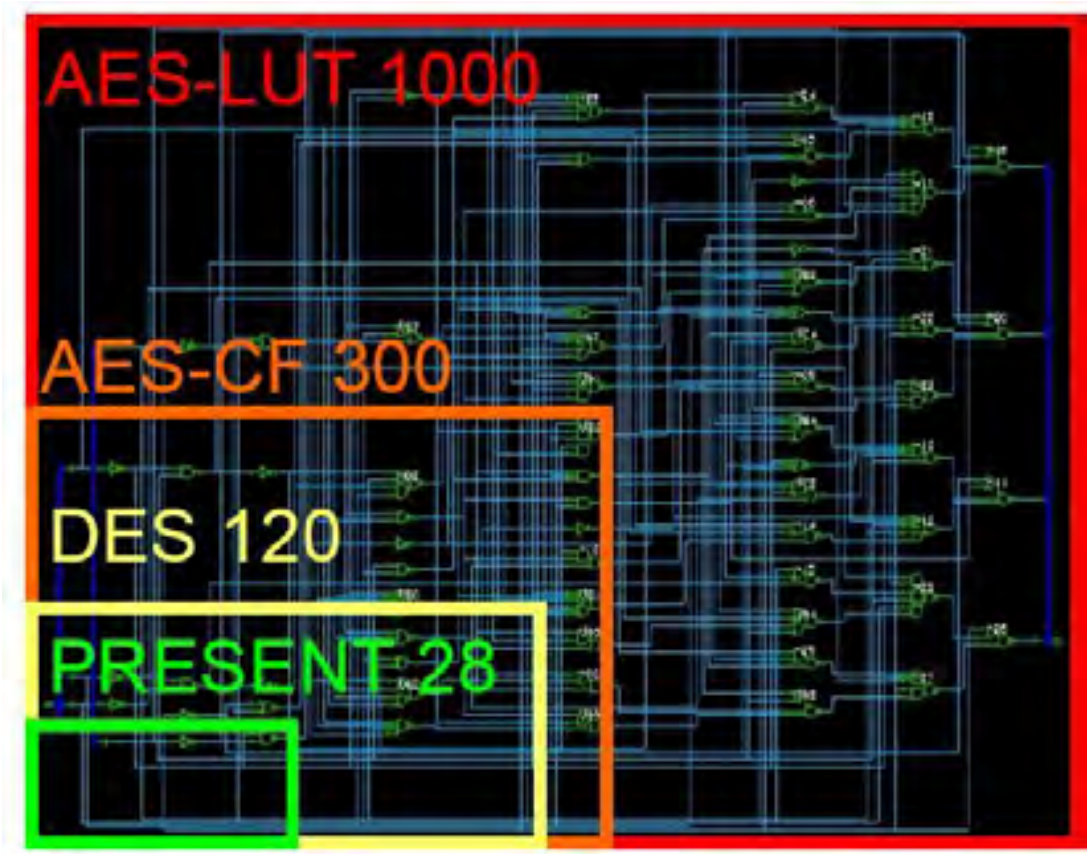



Figure: Comparison of Sboxes

# P-Layer

## Design Issues

The P-Layer has to maximize diffusion.
It has to be cheap.

- Many modern ciphers: MDS codes (great diffusion!)
- DES: Bit permutation (no cost!)

## Design Decision

- Use less diffusion per round
- Use more rounds

# Feistel Cipher

## Feistel Cipher

We have to design a function

$$F : \mathbb{F}_2^n \to \mathbb{F}_2^m$$

- Inspired by practice: Make $m$ small!
- (Highly) unbalanced Feistel cipher.
- Mix with $m$ key bits.

# How far can you go?

## Memory

Given a block-size and a key-size the (minimal) memory requirements are fixed.

## Focus on Area

Minimize the overhead to this.

- PRESENT: 80 percent memory
- KATAN: $\approx$ 90 percent memory

Even doing nothing is not a lot cheaper!

# How far can you go?

Even doing nothing is not a lot cheaper!

## Good or Bad?

- In terms of area: Good
- In terms of battery: Bad

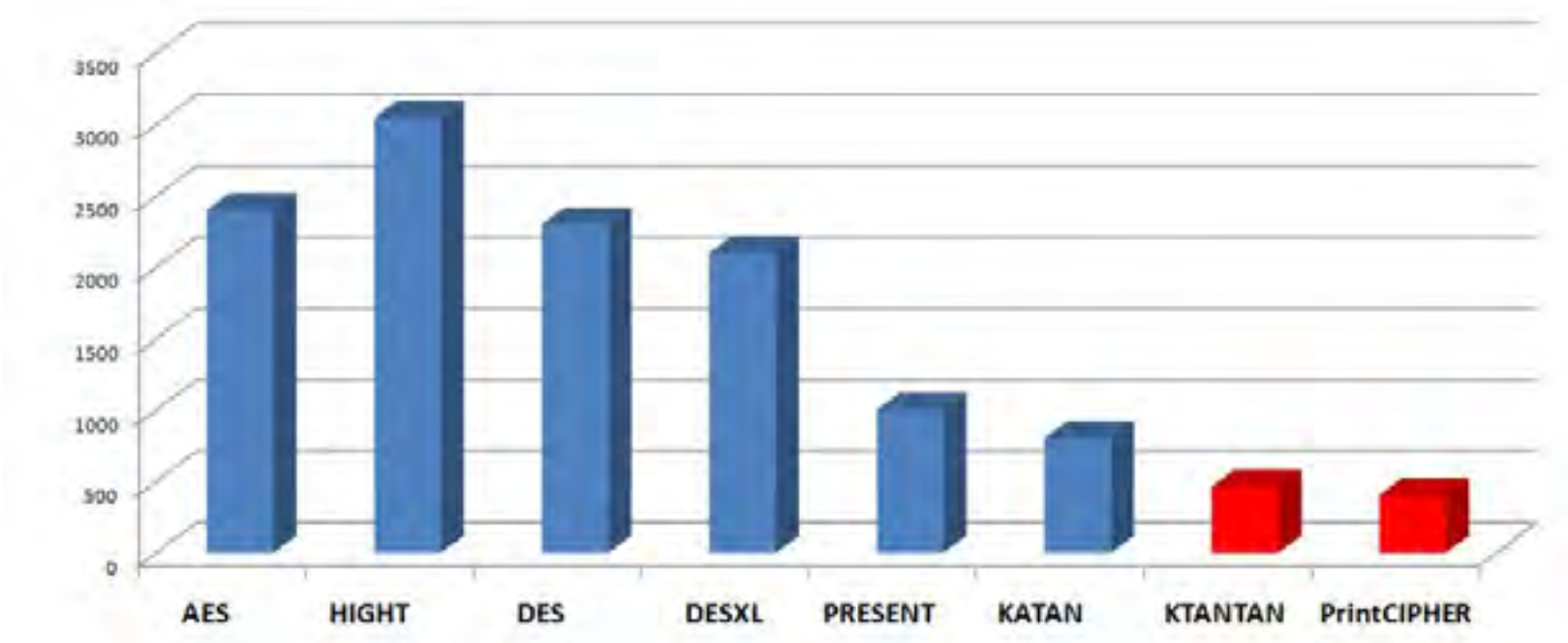
# Examples

## Modern Lightweight block ciphers

- SEA
- DESL
- PRESENT
- KATAN/ KTANTAN
- HIGHT
- PrintCIPHER

A lot more out there...

# A comparison: (To be taken with care)




A fair comparison is difficult

- Many dimensions
- Depends on the technology

# First Example: PRESENT

**PRESENT (CHES 2007)**

A 64 bit block cipher with 80/128 bit key and 31 rounds.

Developed by RUB/DTU/ORANGE

- SP-network
- 4 bit Sbox
- Bit permutation as P-layer

# PRESENT: Overview

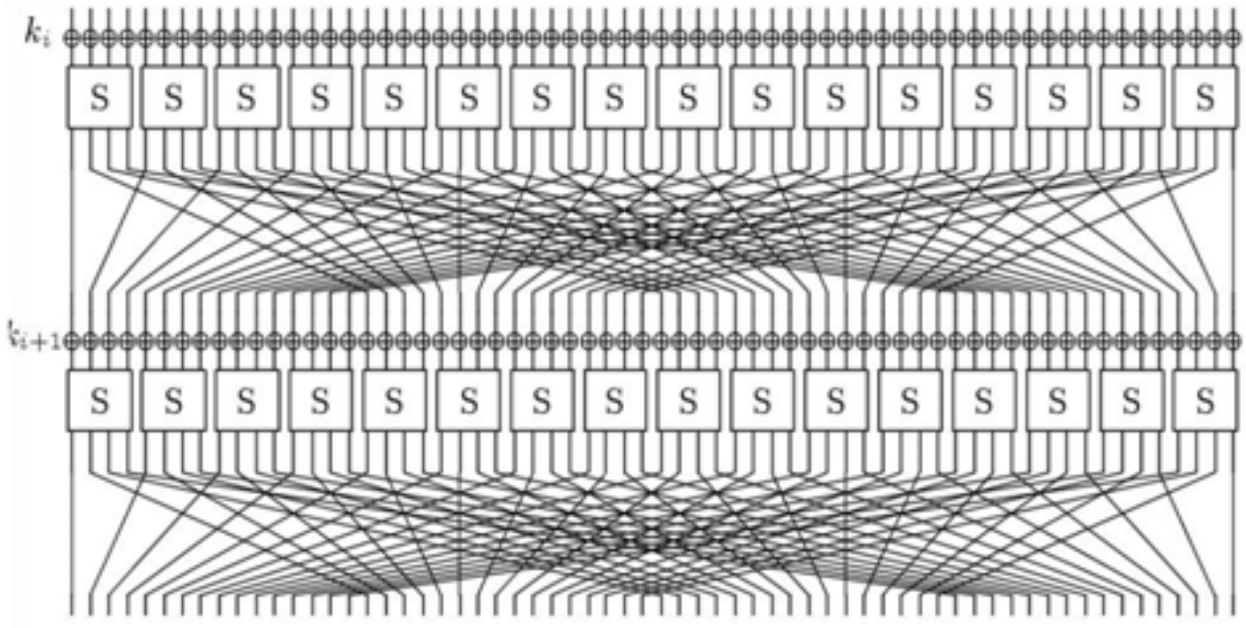



Figure: Overview of PRESENT

# Security of PRESENT

Security Results (a result of simplicity):

**Theorem**

*Any differential characteristic over* 5 *rounds involves at least* 10 *active Sboxes.*

**Theorem**

*Any linear trail over* 4 *rounds has an absolute bias less than* $2^{-7}$.

# Known Attacks on PRESENT

| Rounds | Attack | Complexity |
|---|---|---|
| 16 | DC | $2^{64}$ texts |
| 17 | RKR | $2^{63}$ texts |
| 24 | SSA | $\geq 2^{63}$ texts |
| 26 | LH | $2^{64}$ texts |
| 26 | MLC | $2^{64}$ texts |

# Second Example: KATAN

**KATAN (CHES 2009)**

A 32/48/64 bit block cipher with 80 bit key and 254 rounds.

Developed by KUL

- A (kind of) Feistel-cipher
- Highly unbalanced
- Inspired by Trivium
- Very simple non-linear function

# KATAN: Overview

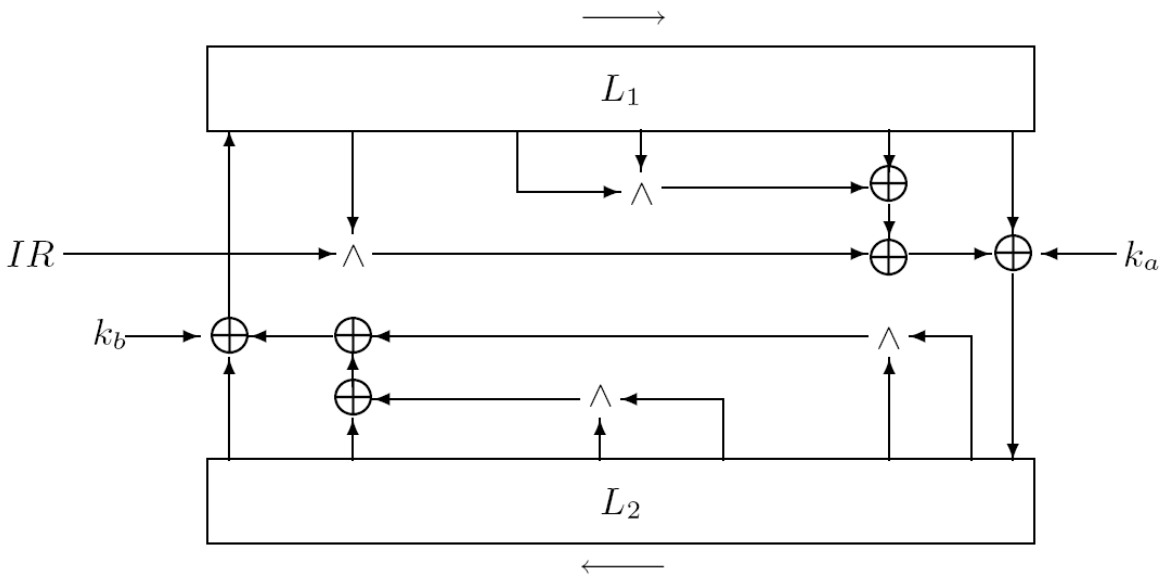



Figure: Overview of KATAN

# Security of KATAN

Security Results (a result of simplicity):

**Theorem**

*For an n-bit block size, no differential characteristic with probability greater than $2^{-n}$ exists for* 128 *rounds.*

**Theorem**

*For an n-bit block size, no linear approximation with bias greater than $2^{-n/2}$ exists for* 128 *rounds.*

# Known Attacks on KATAN

| Rounds | Attack | Complexity |
|---|---|---|
| 78 | Conditional-DC | $2^{22}$ texts |
| 115 | Multi-DC | $2^{32}$ texts |

# Third Example: LED

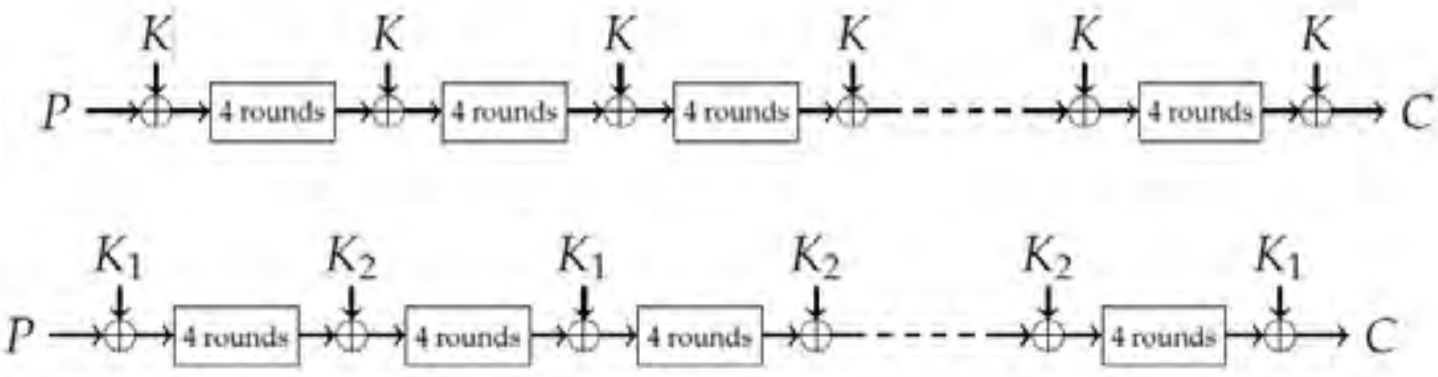
**LED (CHES 2011)**

A 64 bit block cipher with $64 - 128$ bit key and 32/48 rounds.

Developed by NTU and Orange Labs

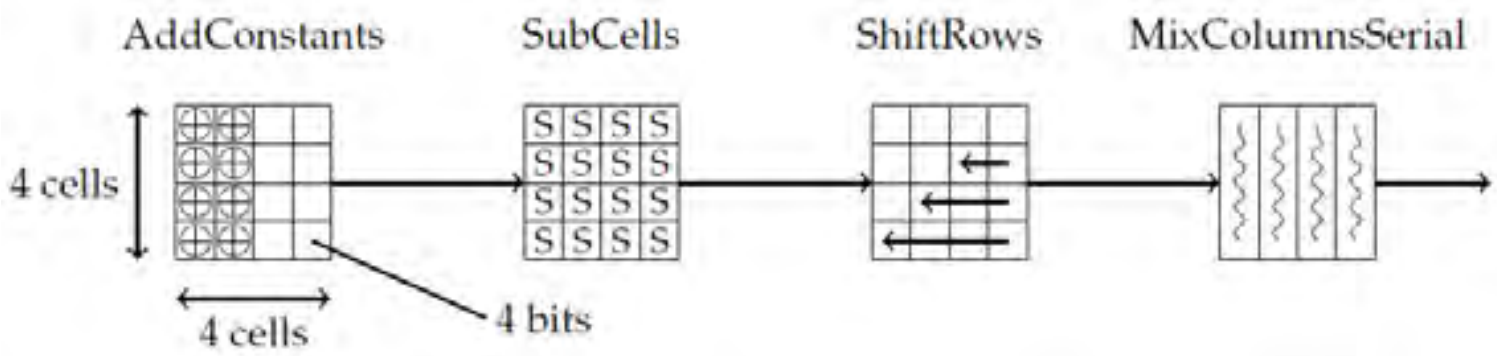
- A SP-network
- Inspired by AES
- Nice tweak to Mix Columns

# LED: Overview



# LED: Round Function

Very AES inspired:




Nice Trick – Hardware friendly MDS Matrix:

$$(B)^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{pmatrix}^4 = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}$$

Very hardware friendly (but slower).

# Security of LED

Security Results (a result of simplicity and similarity to AES):

**Theorem**

- *In single-key model: 25 active Sboxes for 4 rounds of LED*
- *25 active Sboxes for 8 rounds of LED*

Strong results, large security margin.

# Known Attacks on LED

| Rounds | Attack | Complexity |
|---|---|---|
| 15 | Chosen related key (LED-64) | $2^{16}$ texts |
| 27 | Chosen related key (LED-128) | $2^{16}$ texts |

# Remember: How far can you go?

**Memory**

Given a block-size and a key-size the (minimal) memory requirements are fixed.

But maybe the key is fixed...

**Fixed Key**

A fixed key saves a lot of GE!

To make optimal use of this, we need a (very) simple key-scheduling

- KTANTAN
- PrintCIPHER

# Fourth Example: KTANTAN

## KTANTAN (CHES 2009)

A 32/48/64 bit block cipher with 80 bit key and 254 rounds.

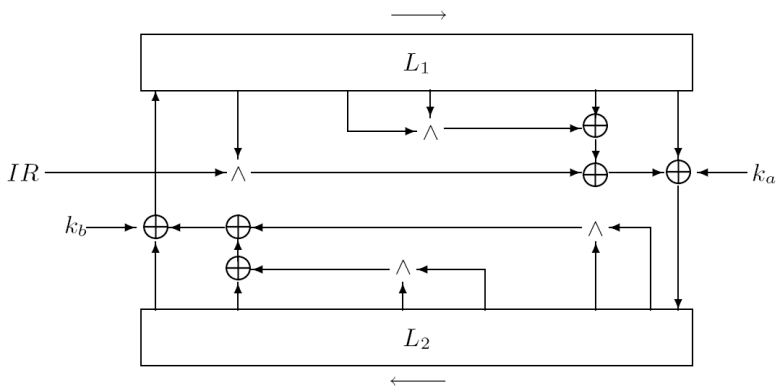



Figure: Overview of KTANTAN

Can you see the difference?

# Fourth Example: KTANTAN

Can you see the difference? No, it is in the key-scheduling

- Round-key-bits selected from the master-key
- Very efficient in hardware

**Generalized Meet-In-The-Middle Attack (Bogdanov-Rechberger+Improvements)**

Selection not well distributed. KTANTAN can be broken in $\approx 2^{75}$.

Can be fixed with little overhead.

# Fifth Example: PrintCIPHER

## PrintCIPHER (CHES 2010)

A 48/96 bit block cipher with 80/160 bit key and 48/96 rounds.

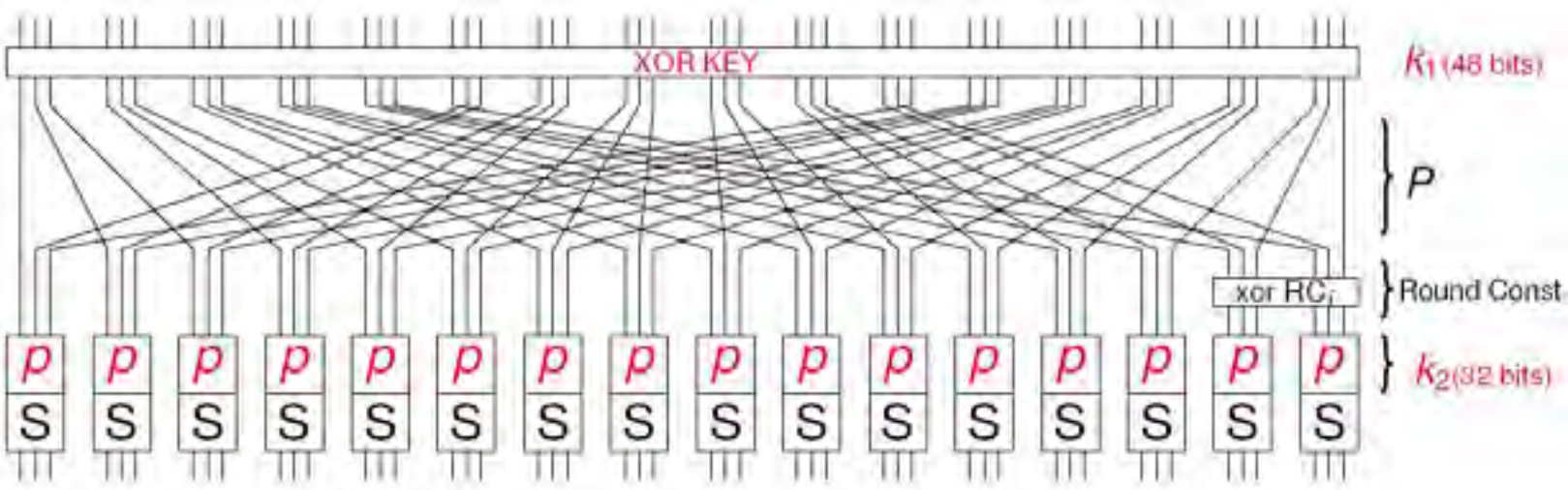



Figure: Overview of PrintCIPHER

# Fifth Example: PrintCIPHER

Again, very simple key-scheduling

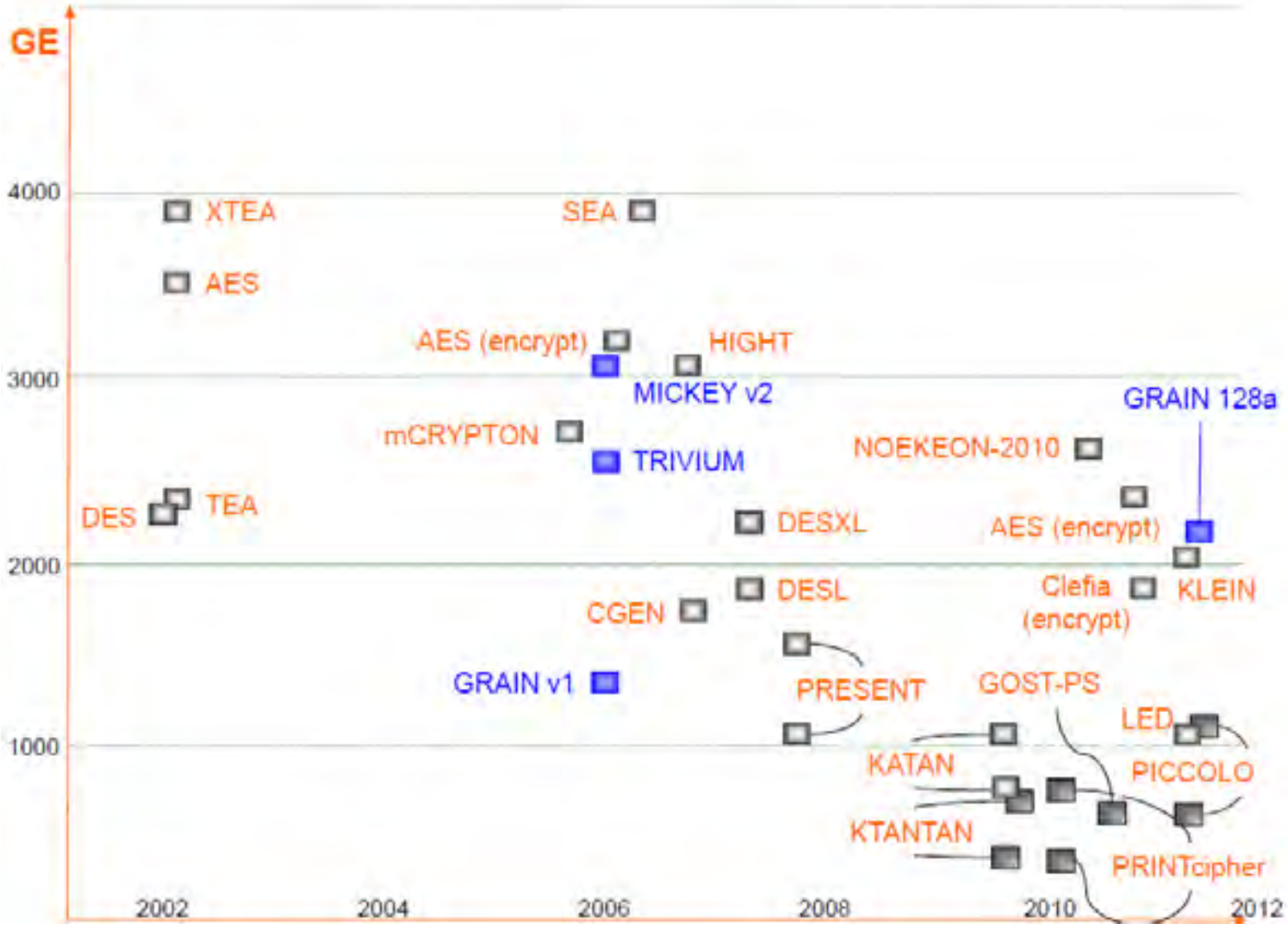
- All round-keys are the same.

**Invariant Subspace Attack (CRYPTO 2011)**

$2^{51}$ (resp. $2^{102}$) weak keys. For those: Distinguisher for PrintCIPHER using a few texts.

Can be fixed with little overhead.

# Overview: As Time Goes By



# Outline

# Origin

## Abdelraheem et al.'12

Invariant Subspace Attack presented at CRYPTO 2012.

## Idea

Make use of a

- weak keys
- that keep a subspace invariant

# Introduction

## PRINTCIPHER

Lightweight SPN block cipher proposed at CHES 2010.

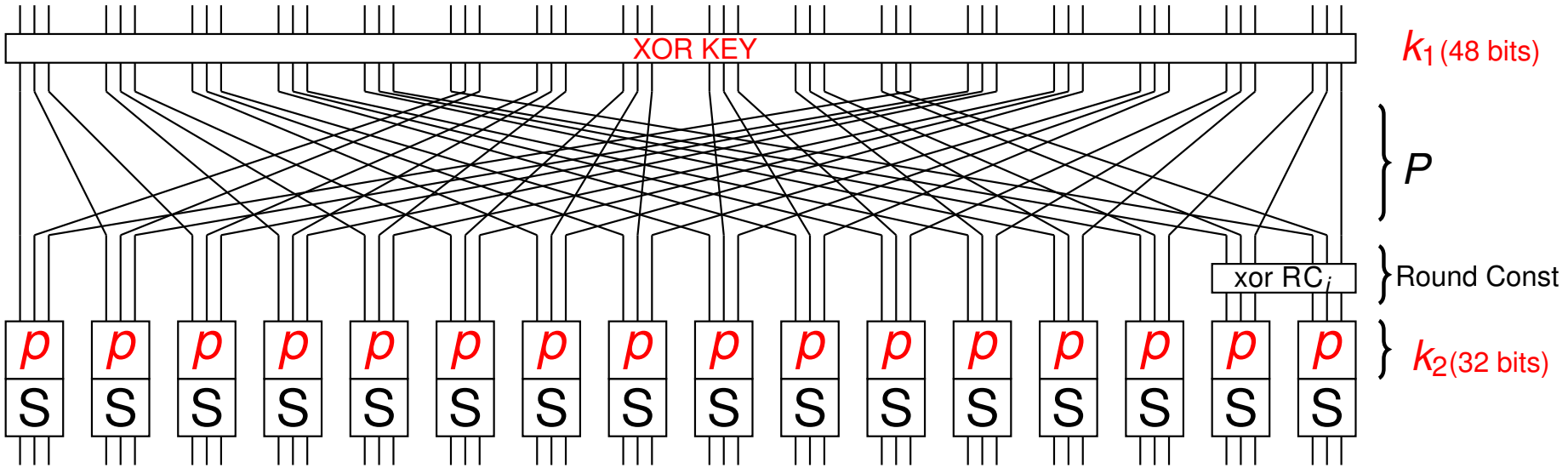
Idea: Take advantage of a fixed key.

## Claim

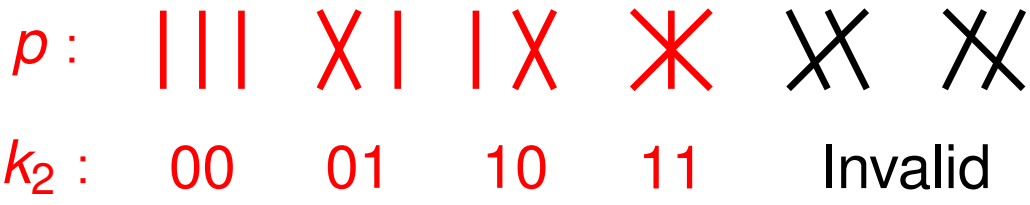
Secure against known attacks.

So far: Attacks on reduced-round variants.

# One round of PRINTCIPHER-48




- 48-bits block size, 48 rounds that use the same 80-bit key.
- Each two bits of $k_2$ permute 3 state bits in a certain way.
- Only 4 out of 6 possible permutations are allowed:



# Simplify Things

In this presentation: A simpler variant of PRINTCIPHER.

- Block size 24
- Fix the permutation key
- Modified Sbox

# Sbox Property

Modified Sbox:

$$S(000) = 000 \qquad S(001) = 001$$
$$S(010) = 010 \qquad S(100) = 100$$

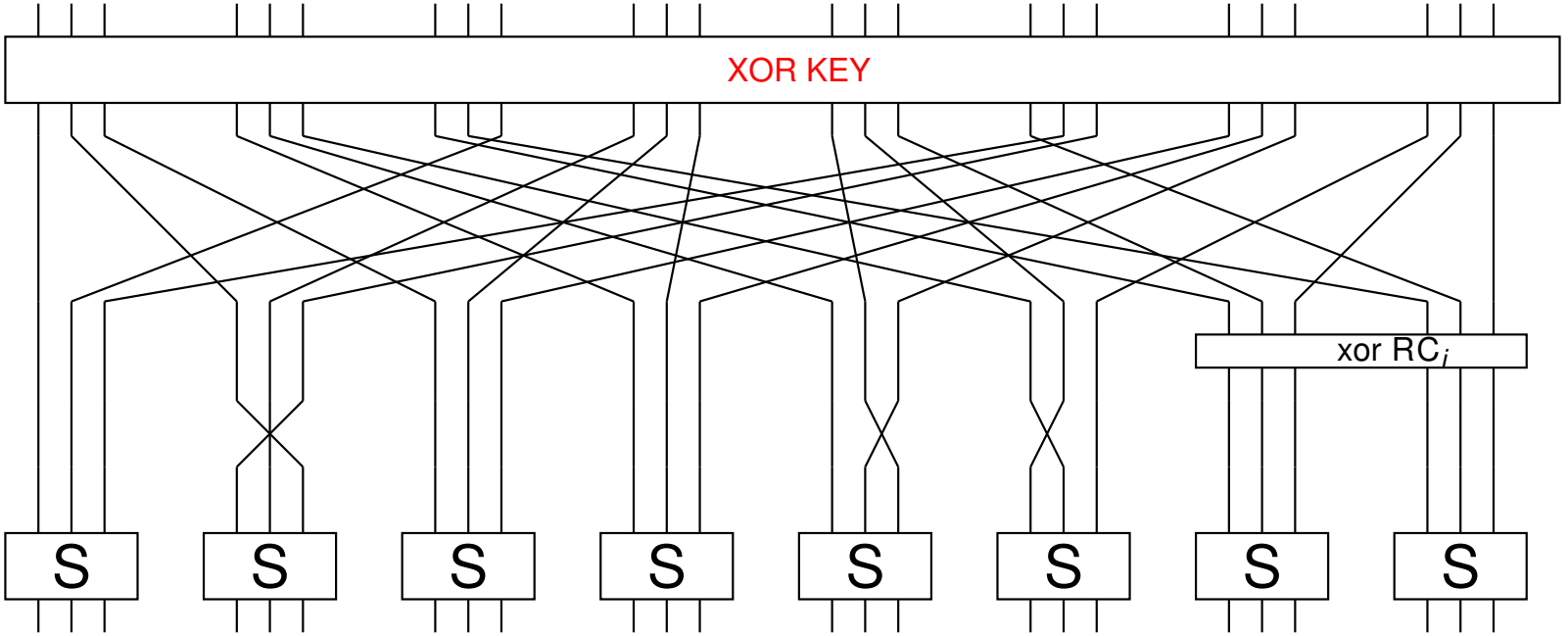
Can be written as:

$$\begin{aligned} S(00*) &= 00* \\ S(0*0) &= 0*0 \\ S(*00) &= *00 \end{aligned}$$

**Remark**

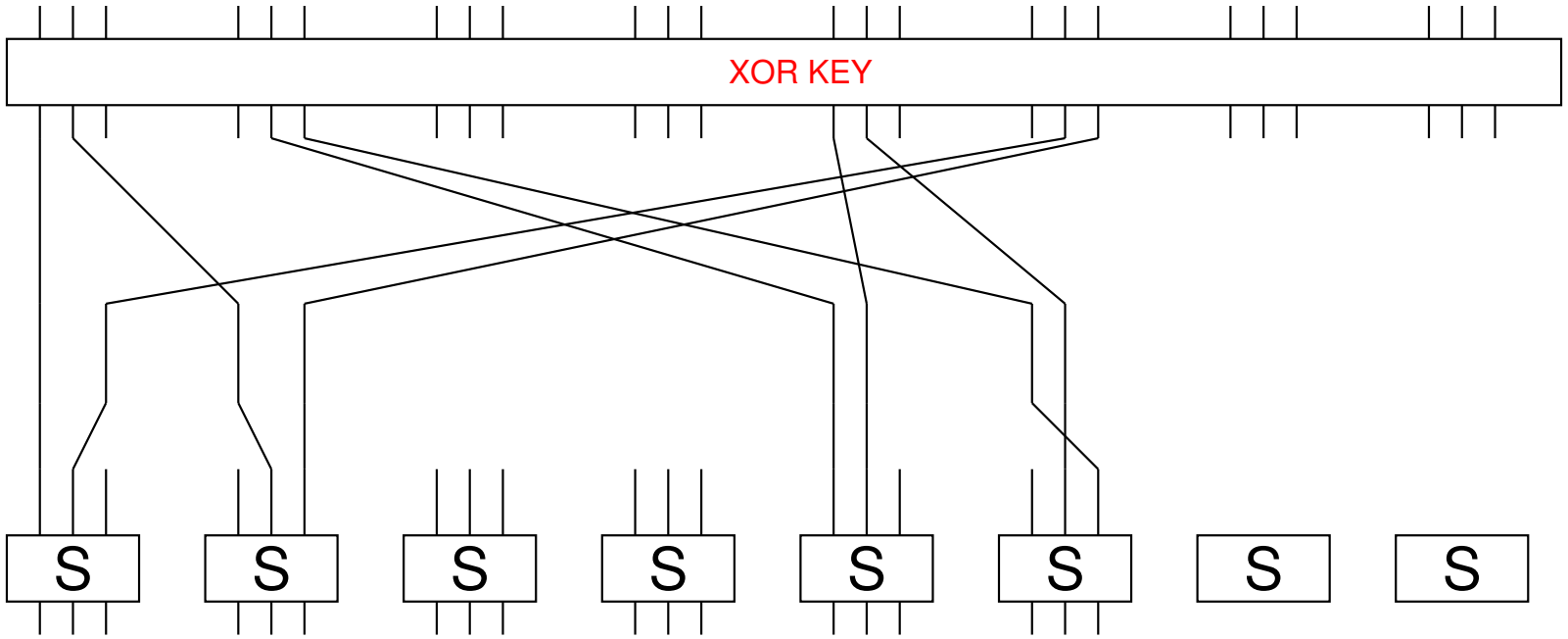
The original Sbox fulfils something similar.

# Simplified Version




$$S(00*) = 00*$$
$$S(0*0) = 0*0$$
$$S(*00) = *00$$

# Let's Focus




## Invariant Subspace for $P$

Set of highlighted bits is mapped onto itself.

# What about $S$

An Invariant Subspace alone is not a problem!

**Question**

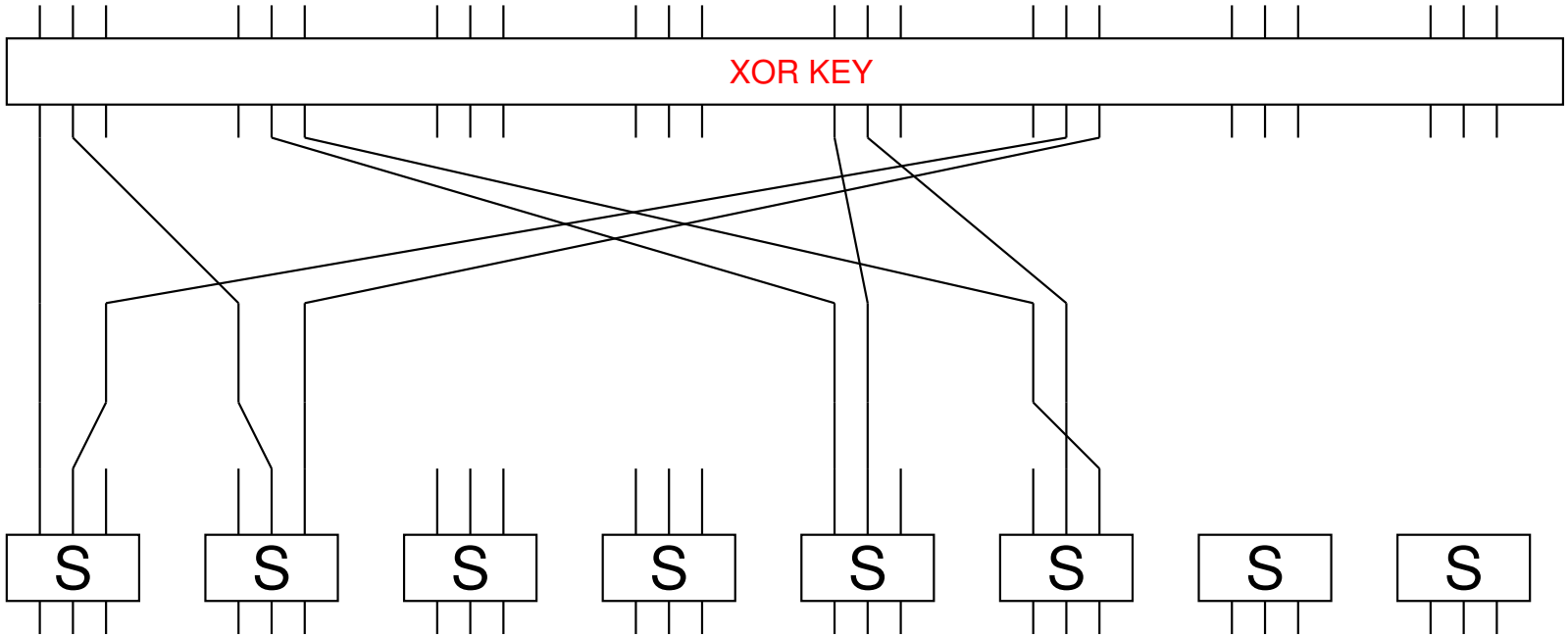
What about the $S$-layer?

For this: we fix some bits

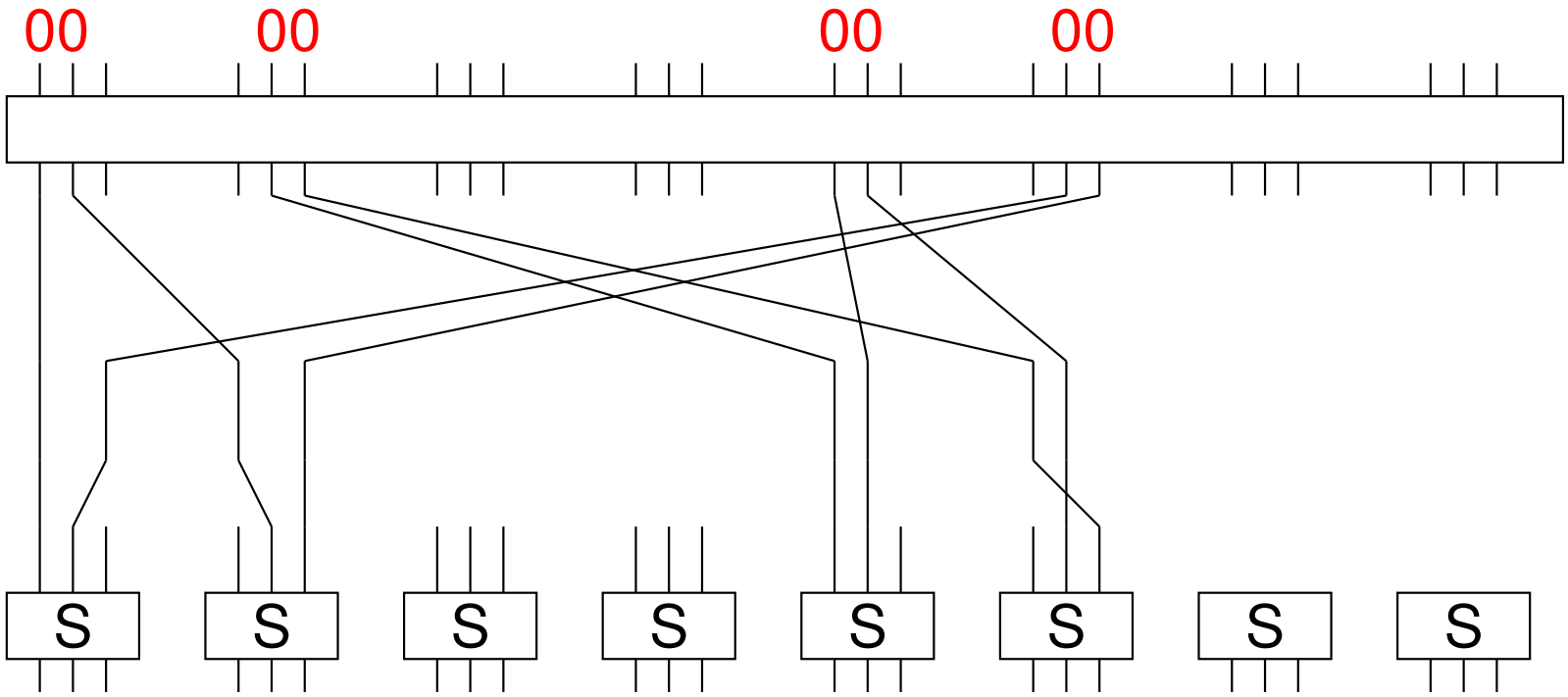
- in the plaintext
- in the (XOR)-key

$\Rightarrow$ The attack does not work for all keys.

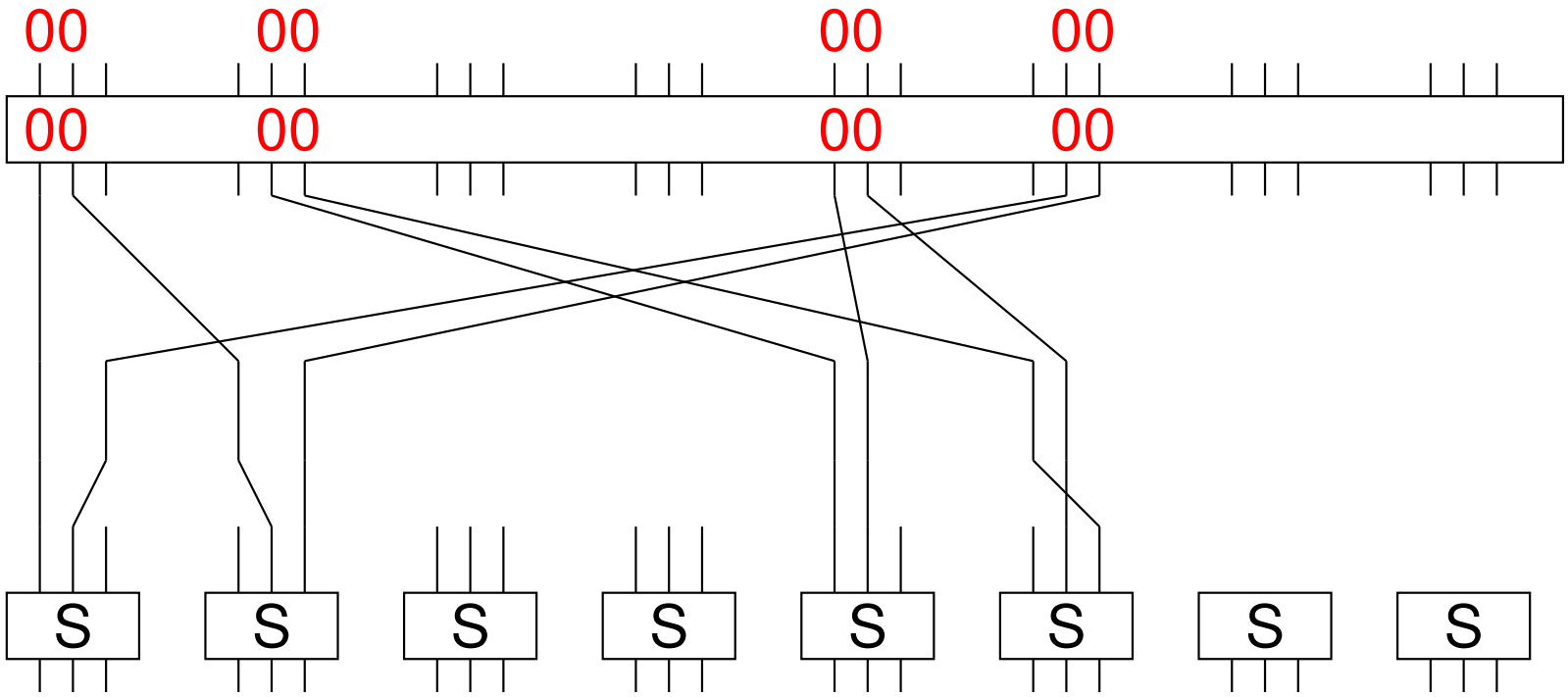
# Simplified Version



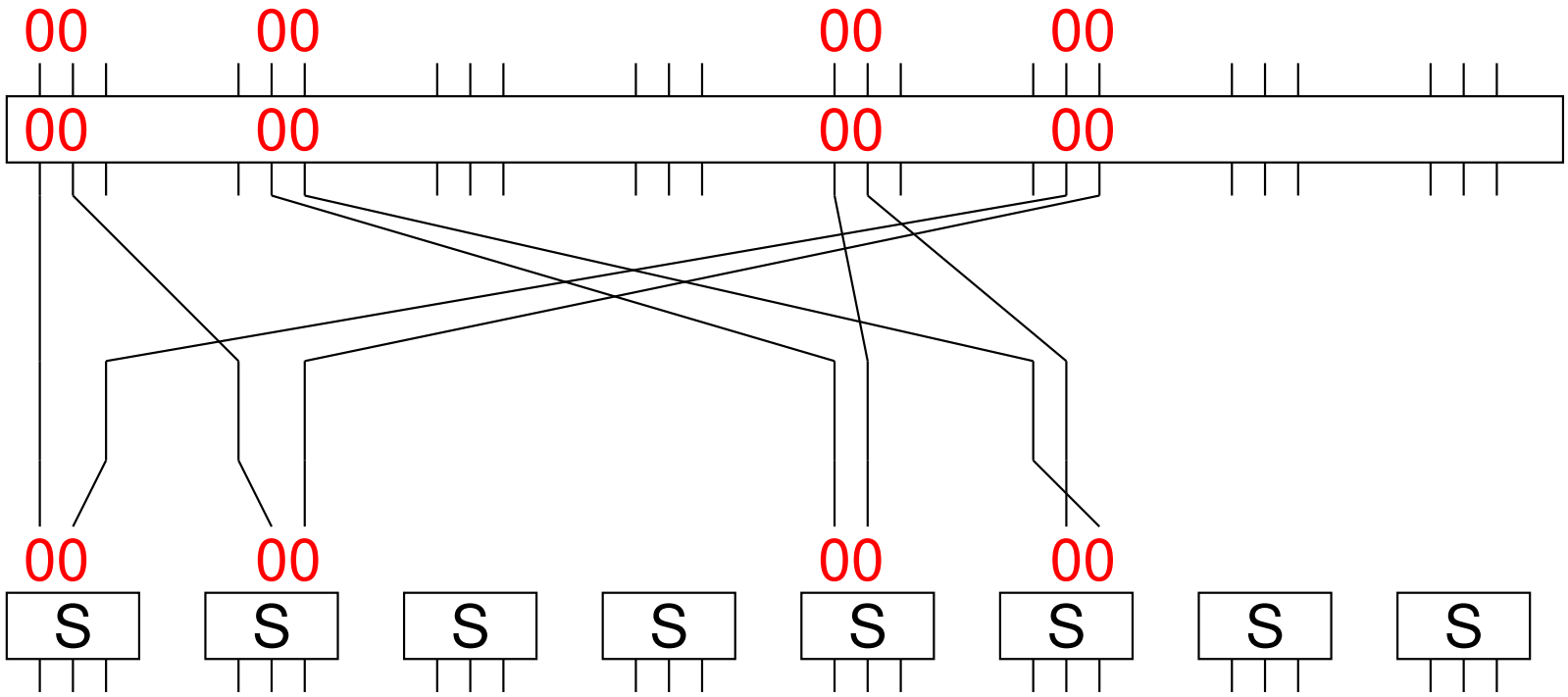
# Simplified Version



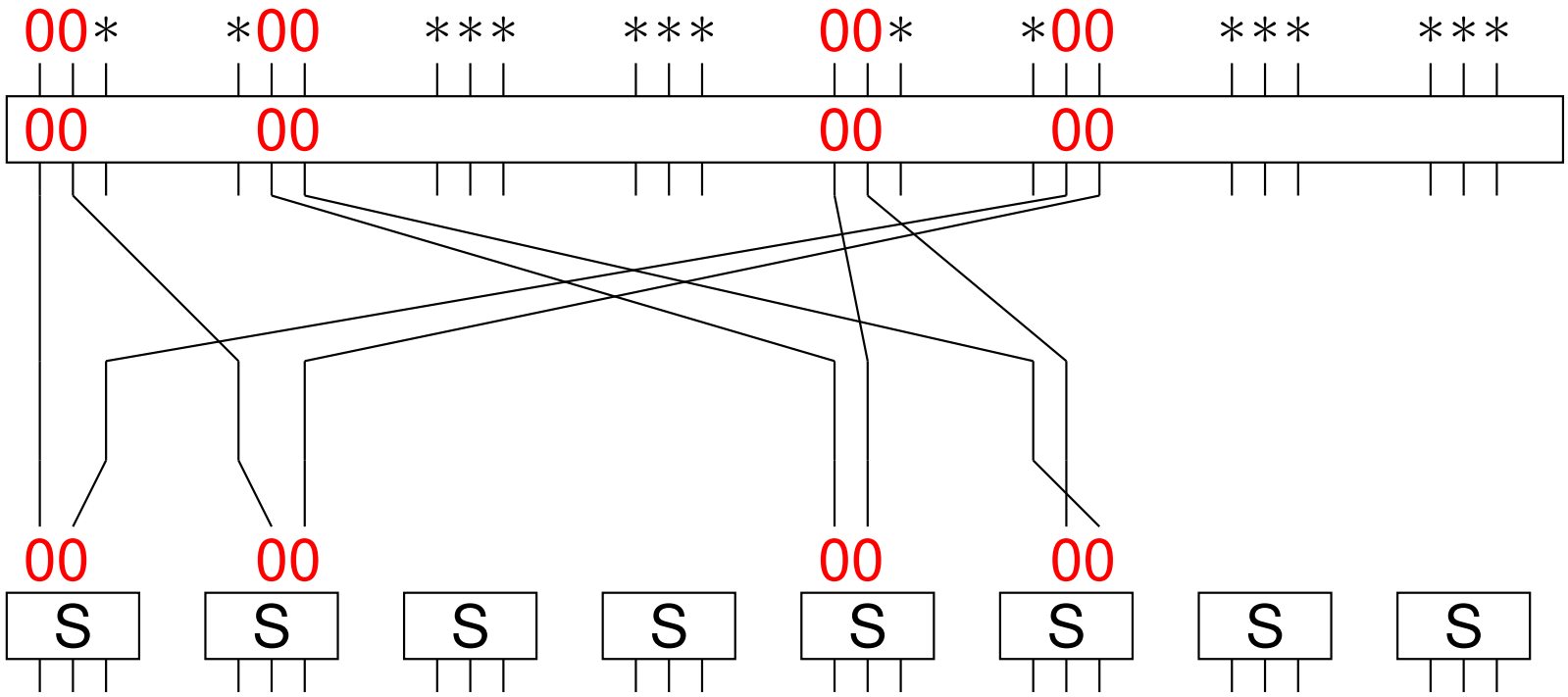
# Simplified Version



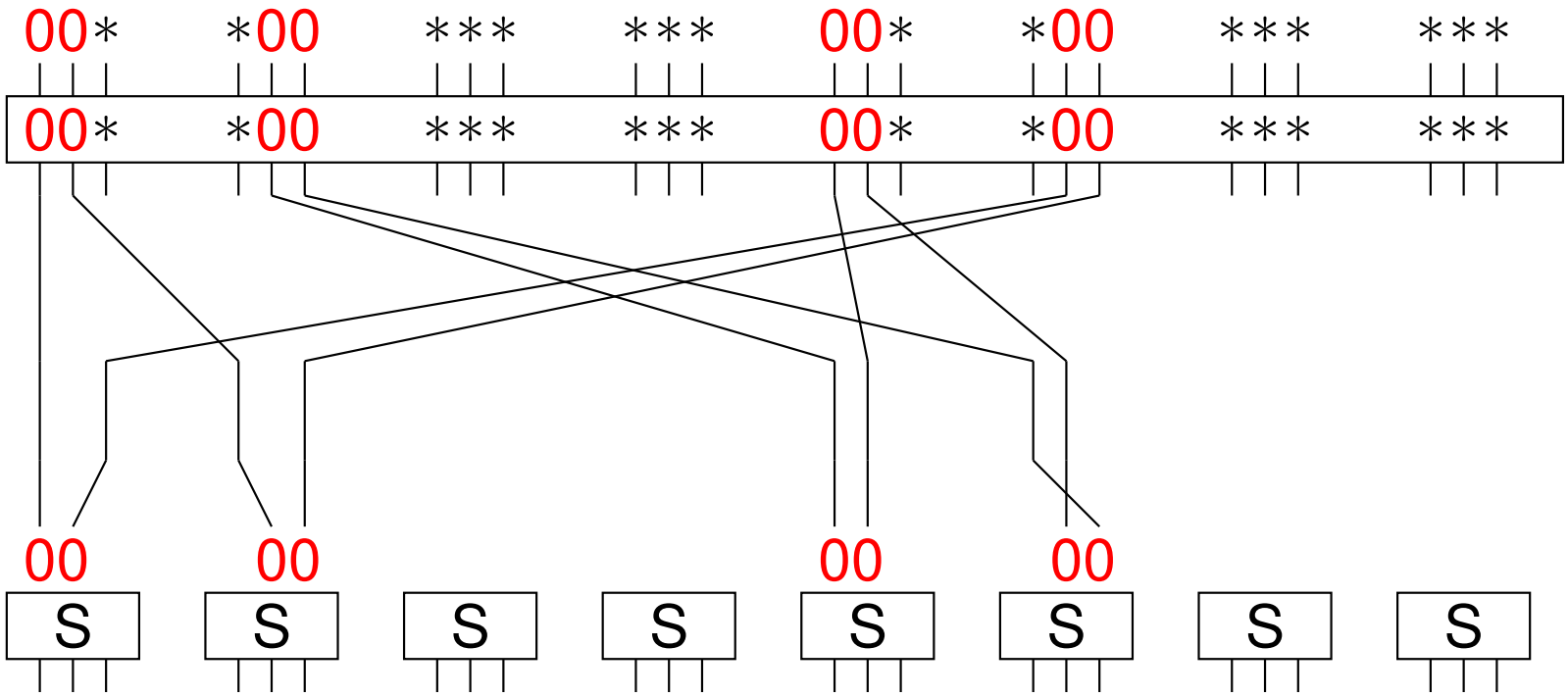
# Simplified Version



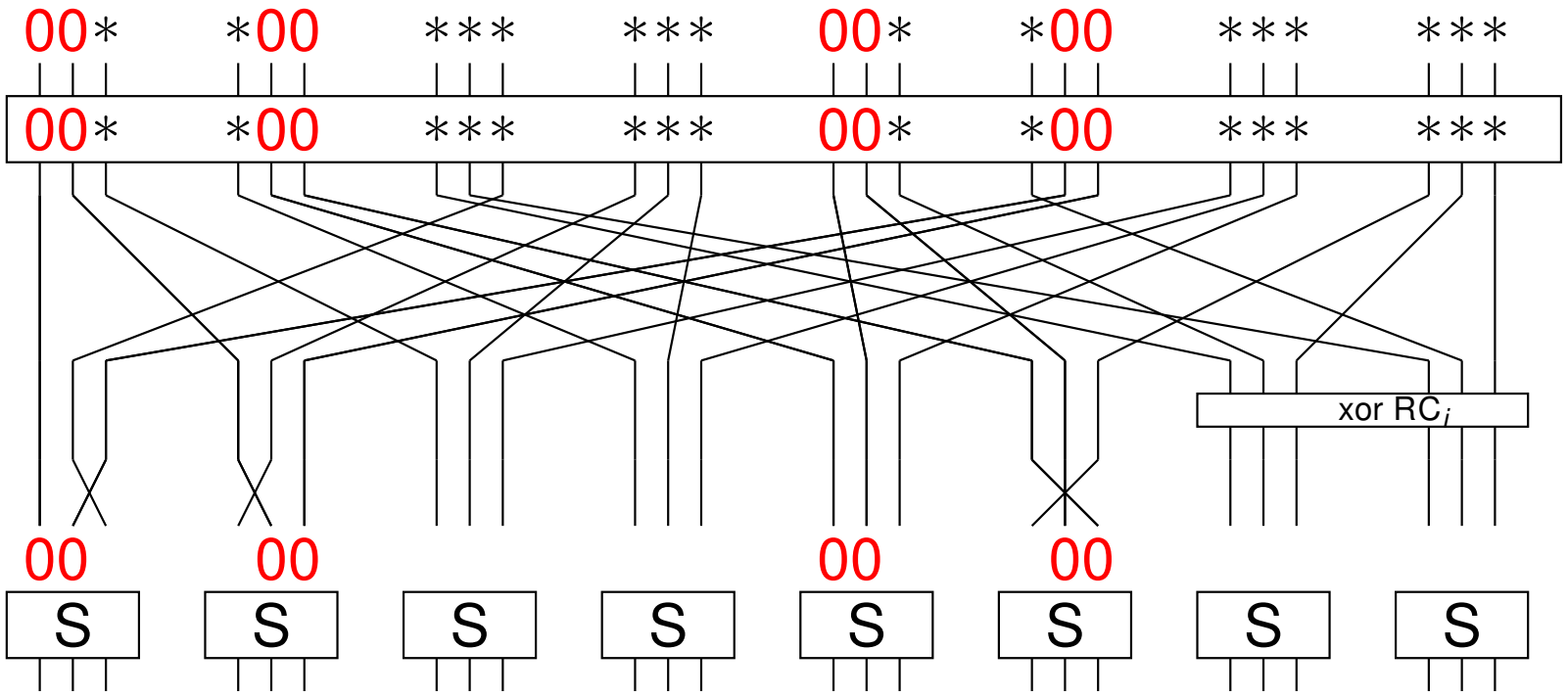
# Simplified Version



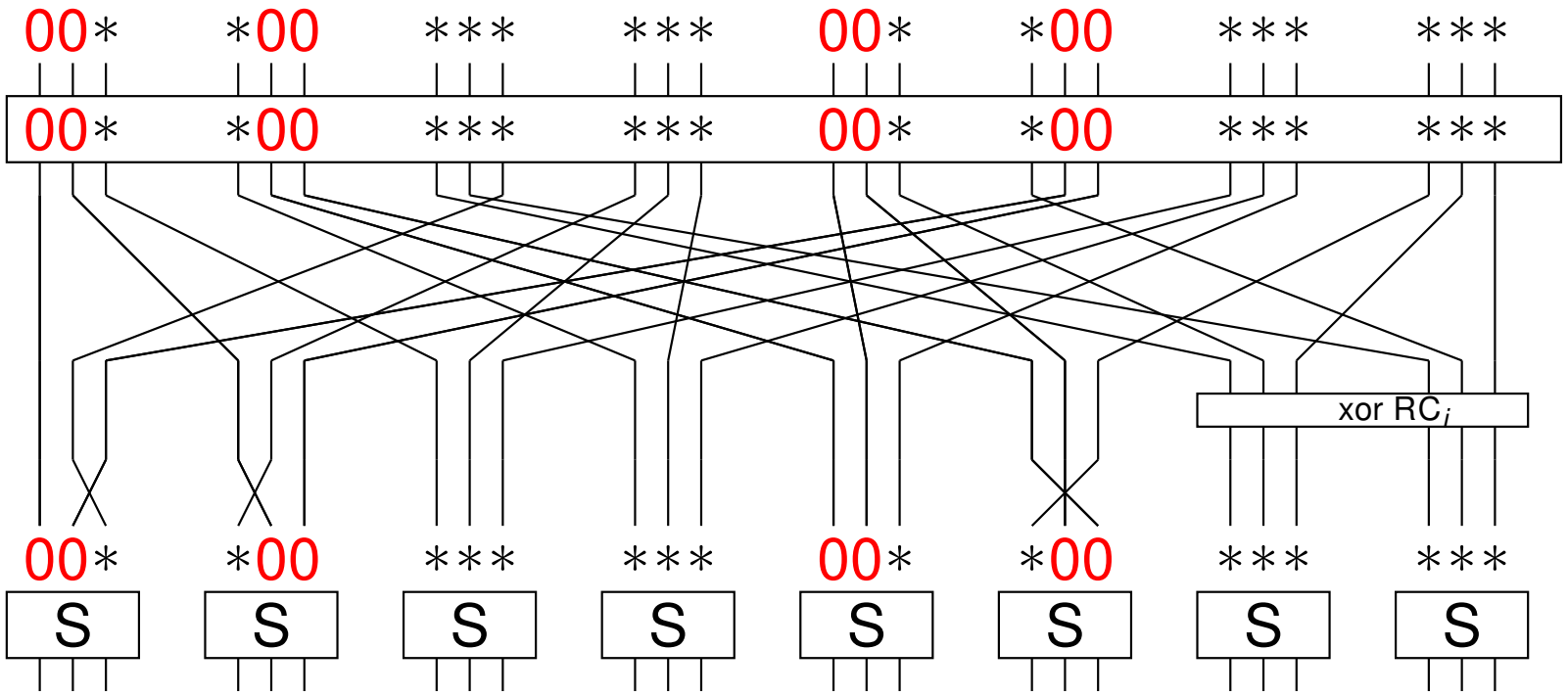
# Simplified Version



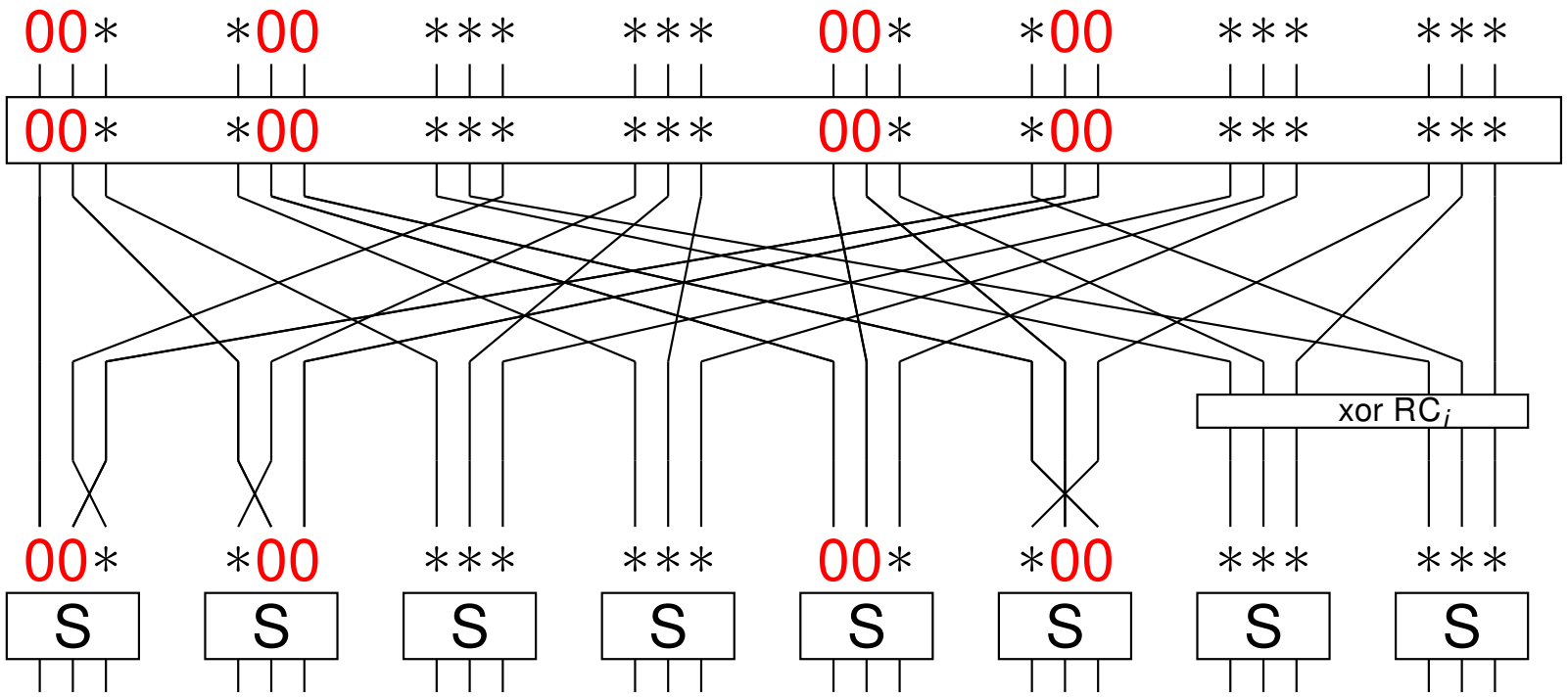
# Simplified Version



# Simplified Version

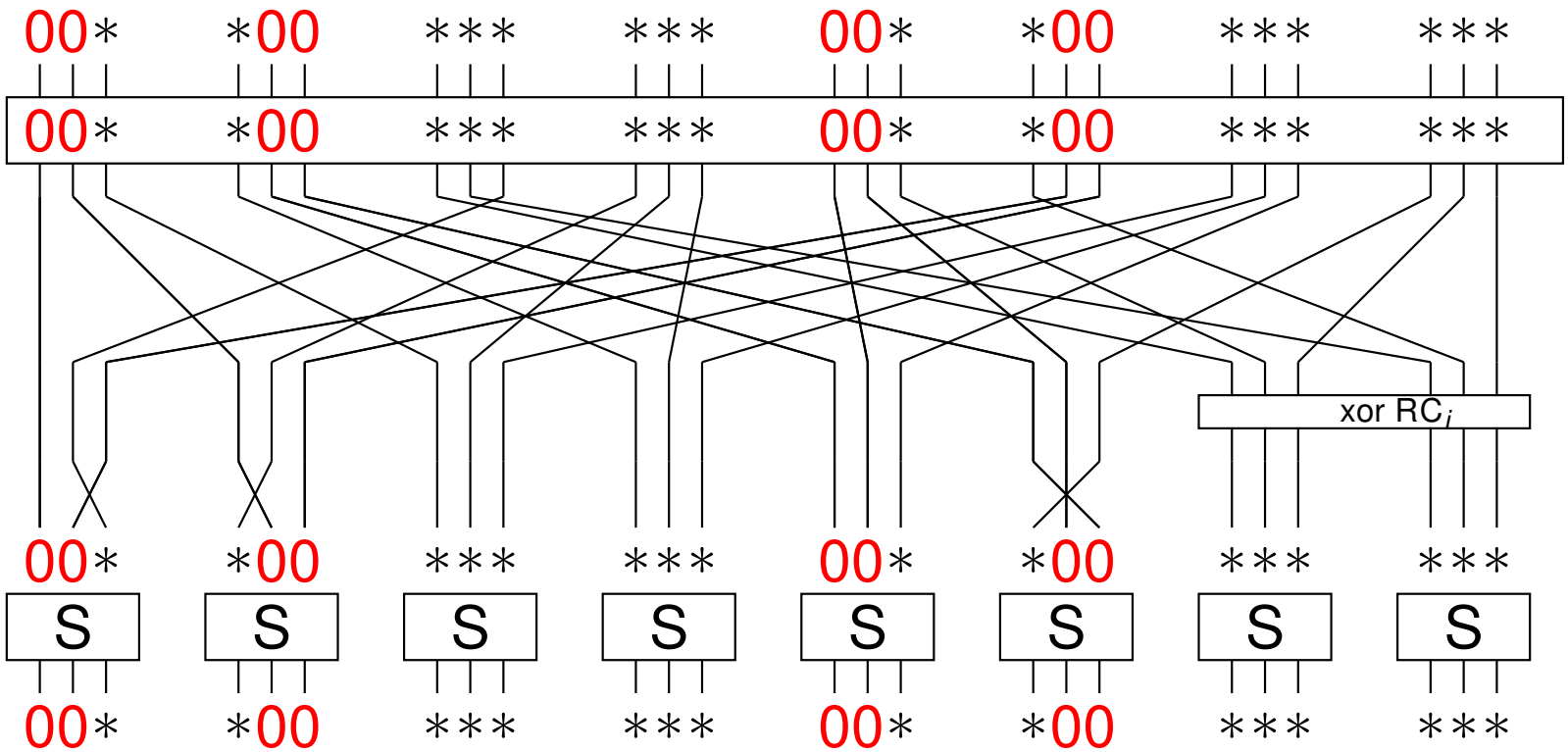


# Simplified Version



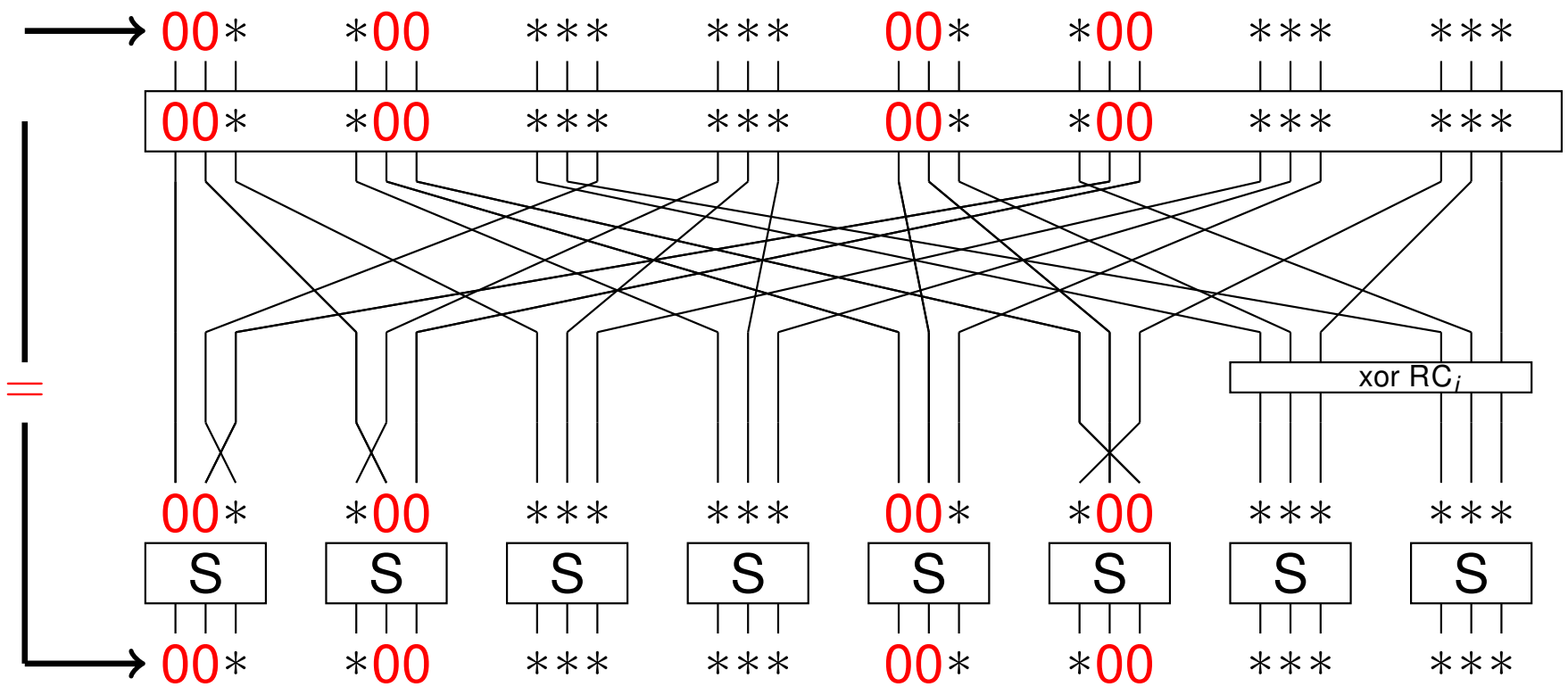

$$S(00*) = 00* \quad S(0*0) = 0*0 \quad S(*00) = *00$$

# Simplified Version




$$S(00*) = 00* \quad S(0*0) = 0*0 \quad S(*00) = *00$$

# Simplified Version




$$S(00*) = 00* \quad S(0*0) = 0*0 \quad S(*00) = *00$$

# An Iterative One-Round Distinguisher

If certain key bits are zero:

**Distinguisher**

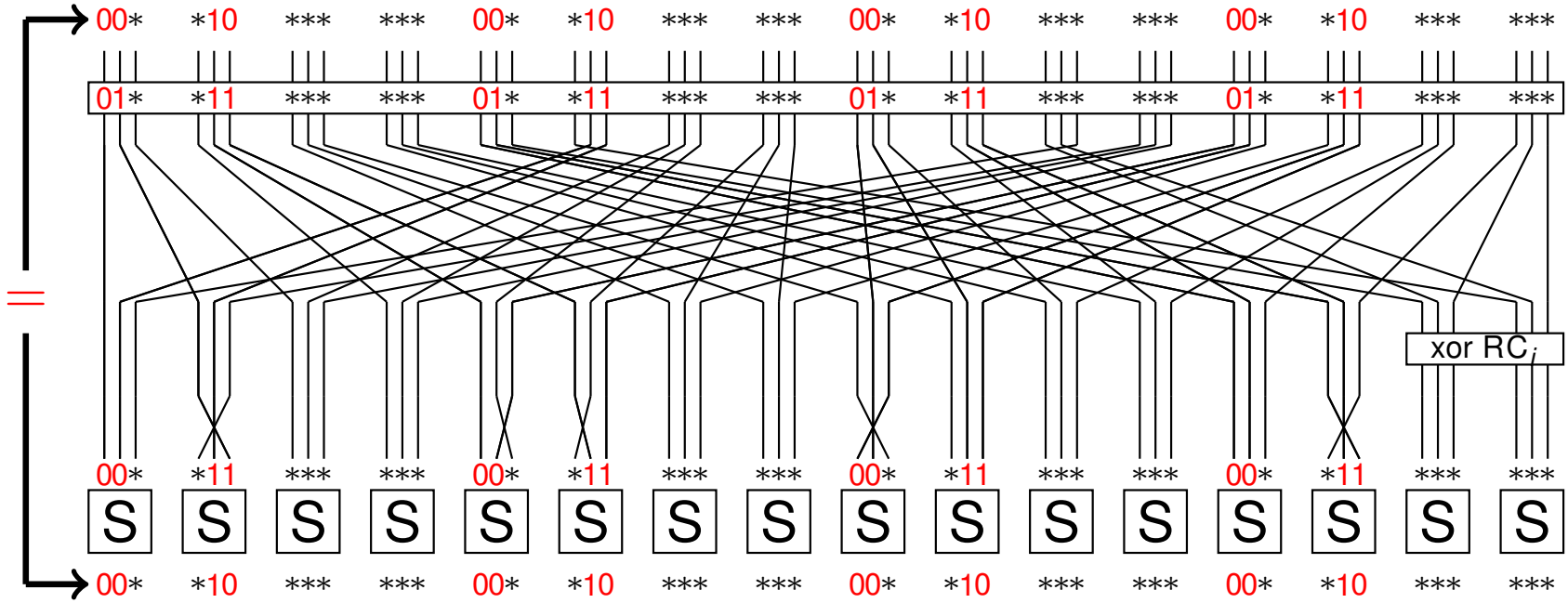
Zero bits in the plaintext $\Rightarrow$ zero bits in the ciphertext.

Some Remarks:

- Round-constant does not help
- Works for the whole cipher

Let's look at PRINTCIPHER-48

# The Attack on PRINTCIPHER-48




$$S(00*) = 00*$$
$$S(1*0) = 1*1$$
$$S(*11) = *10$$

# PRINTCIPHER-48 Attack

## Summary

- Prob 1 distinguisher for full cipher
- $2^{50}$ out of $2^{80}$ keys weak.
- Similar for PRINTCIPHER-96

Abstraction:

$$R(U \oplus d) = U \oplus c$$

If $k \in U \oplus (d \oplus c)$

$$R_k(U \oplus d) = U \oplus d$$

Thus an invariant subspace

# The Probability of A Characteristic

Given a $r$-round differential characteristic

$$\alpha \xrightarrow{p} \alpha \xrightarrow{p} \cdots \xrightarrow{p} \alpha$$

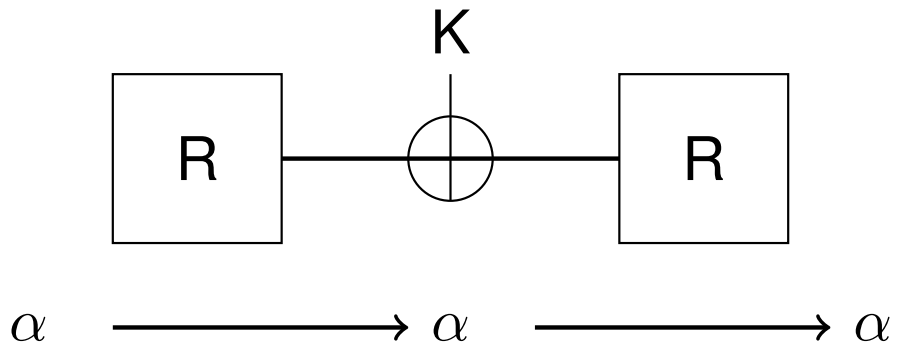
**Theorem**

*Given independent round keys the average probability is $p^r$*

**Hypothesis of Stochastic Equivalence**

All keys behave similarly.

# Two Round Characteristics




$$A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$$

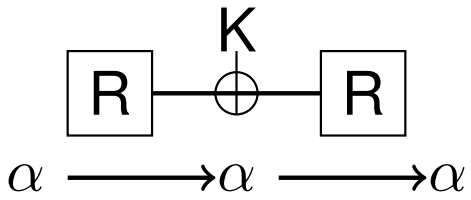
"$A$ is the set of good pairs"

## Two Rounds, fixed Key

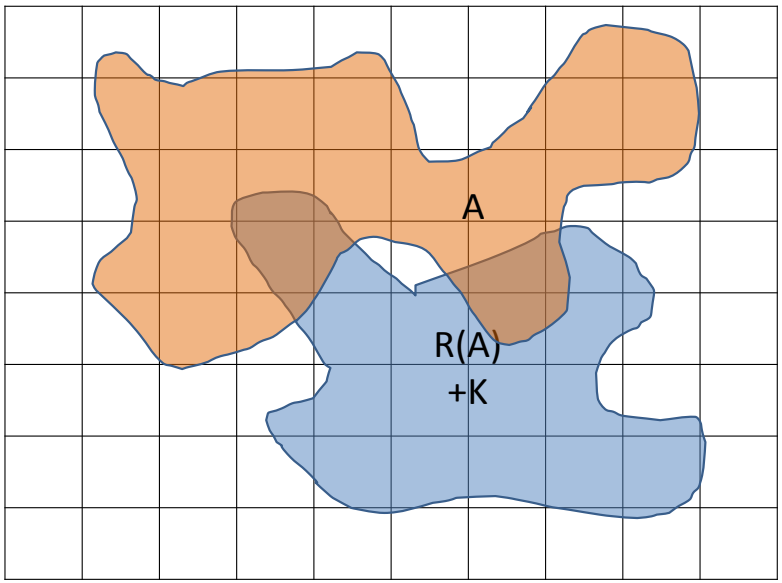
Probability of the characteristic for a key $K$:

$$\frac{|(R(A) \oplus K) \bigcap A|}{2^n}$$

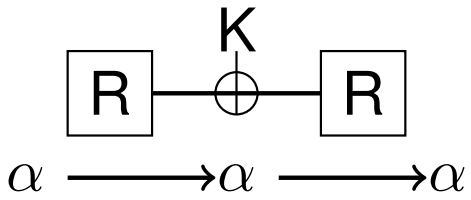
# Two Rounds, fixed Key



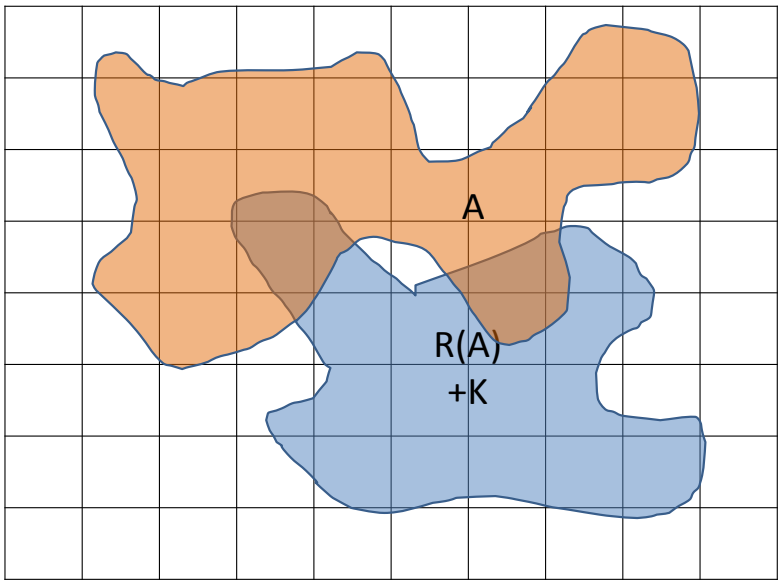

Good Pairs: $A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$
Probability (scaled): $|(R(A) \oplus K) \bigcap A|$

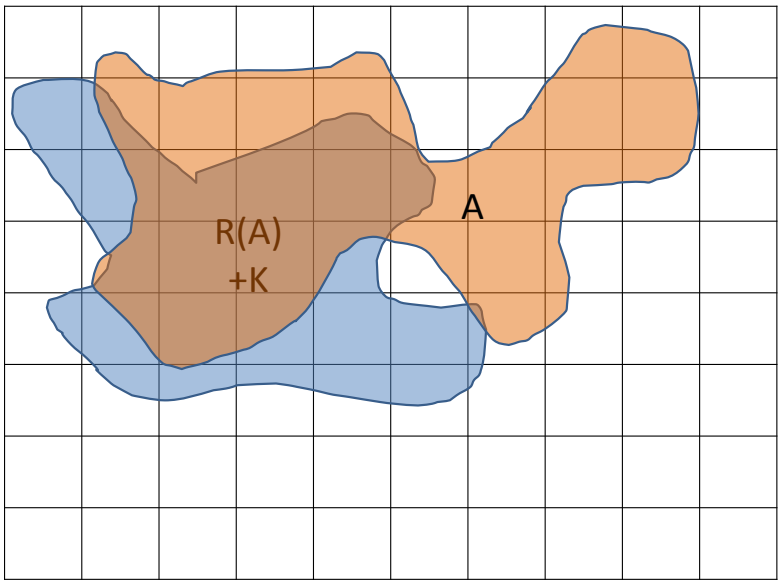


# Two Rounds, fixed Key

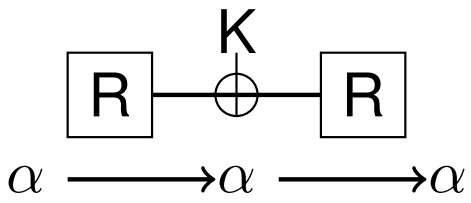



Good Pairs: $A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$
Probability (scaled): $|(R(A) \oplus K) \bigcap A|$



# Back To PRINTCIPHER-48




Good Pairs: $A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$

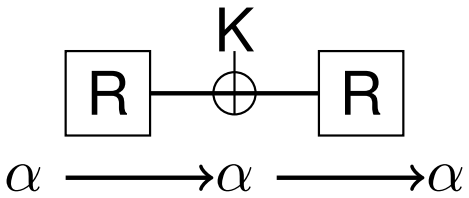
## Observations for special $\alpha$

- $A$ is an affine subspace $U \oplus d$
- $U$ is invariant under $R$
- $\Rightarrow \; R(A) = U \oplus c$

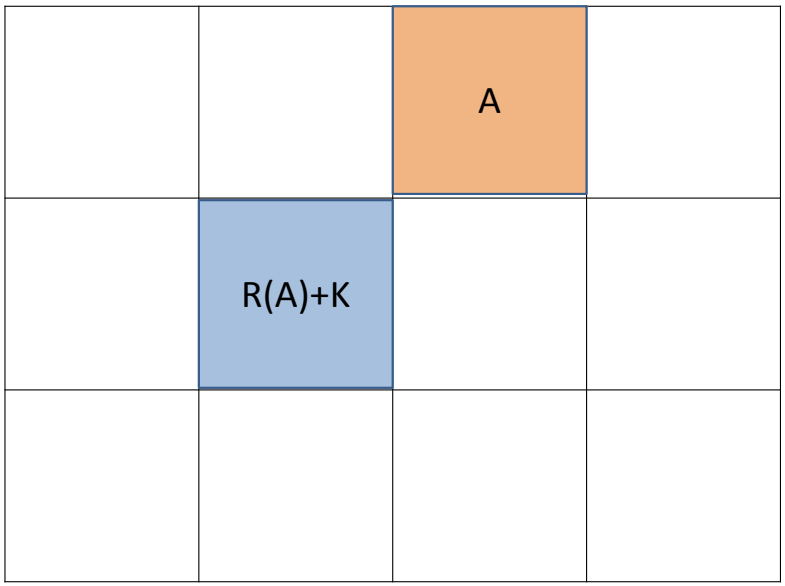
Probability (scaled):

$$\left|(R(A) \oplus K) \bigcap A\right| = \left|(U \oplus c \oplus K) \bigcap (U \oplus d)\right|$$

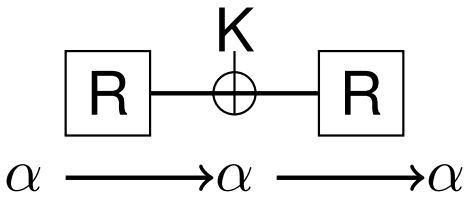
# Two Rounds, fixed Key: PRINTCIPHER-48



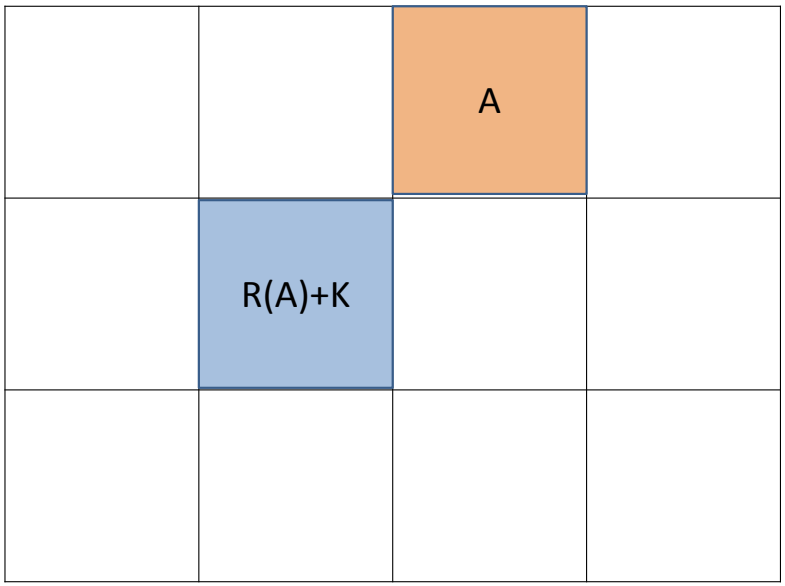

Good Pairs: $A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$
Probability (scaled): $|(R(A) \oplus K) \bigcap A|$

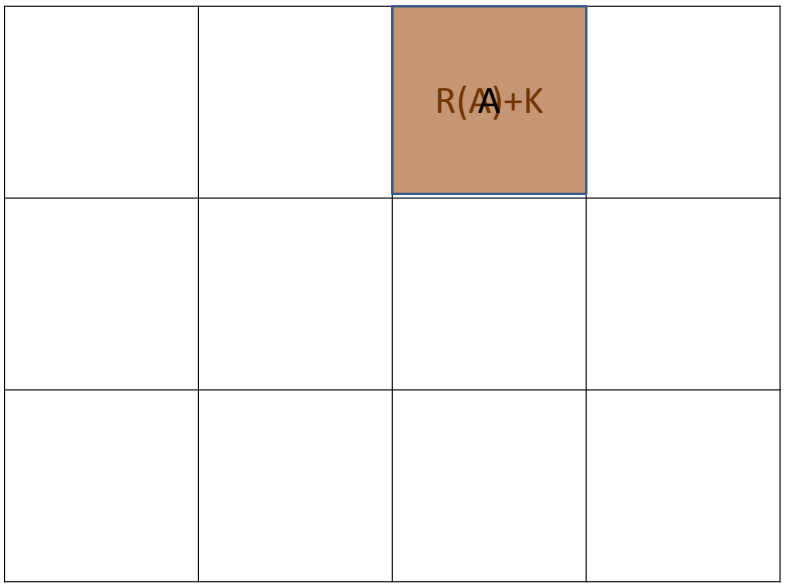


# Two Rounds, fixed Key: PRINTCIPHER-48




Good Pairs: $A := \{x \mid R(x) \oplus R(x \oplus \alpha) = \alpha\}$
Probability (scaled): $|(R(A) \oplus K) \bigcap A|$



# PRINTCIPHER-48

There exist a $r$-round differential characteristic

$$\alpha \rightarrow \alpha \rightarrow \cdots \rightarrow \alpha$$

such that

$$p_k = \begin{cases} 2^{-16} & \text{if} \quad k \text{ is weak} \\ 0 & \text{if} \quad k \text{ is not weak} \end{cases}$$

## Remarks

- Probabilities do not multiply
- Keys behave very differently

# Strongly Biases Linear Approximations

It can be shown:

**Theorem (PRINTCIPHER-48)**

*Given a weak-key there exist linear approximations with correlation $\geq 2^{-17}$ for any number of rounds.*

Not too hard to see but surprising!

# Linear Cryptanalysis: The Idea

**Idea (Matsui '93)**

Approximate a linear combination of plaintext bits with a linear combination of ciphertext bits (and key bits).

Example:

$$p_0 + p_{31} = c_0 + c_{14} + k_1$$

with probability

$$\frac{1}{2} + \epsilon \quad \epsilon \text{ is the bias}$$

More convenient:

$$\text{Correlation} := 2\epsilon$$

# Linear Trails vs. Linear Hulls

A *linear trail U* consists of

- An input mask $\alpha$
- An output mask $\beta$
- Intermediate masks for every rounds.

$$U = (\alpha, \ldots, \beta)$$

Easy to compute the correlation $C_U$ for a given trail.

# Linear Trails vs. Linear Hulls

A *linear trail U* consists of

- An input mask $\alpha$
- An output mask $\beta$
- Intermediate masks for every rounds.

$$U = (\alpha, \ldots, \beta)$$

Easy to compute the correlation $C_U$ for a given trail.

**Theorem (Correlation of Linear Approximations)**

$$C(\alpha, \beta) = \sum_{\substack{U \\ U_0=\alpha, U_n=\beta}} (-1)^{\langle U,K \rangle} |C_U|$$

# Linear Hulls

## Theorem (Correlation of Linear Approximations)

$$C = \sum_{U} (-1)^{\langle U,K \rangle} |C_U|$$

Only the signs are key depended.

## Design Strategy

- Show that $|C_U|$ is small for all trails.
- Assume "good" behavior of signs

This is what we did for PRINTCIPHER!

# Surprising?

**Theorem (PRINTCIPHER-48)**

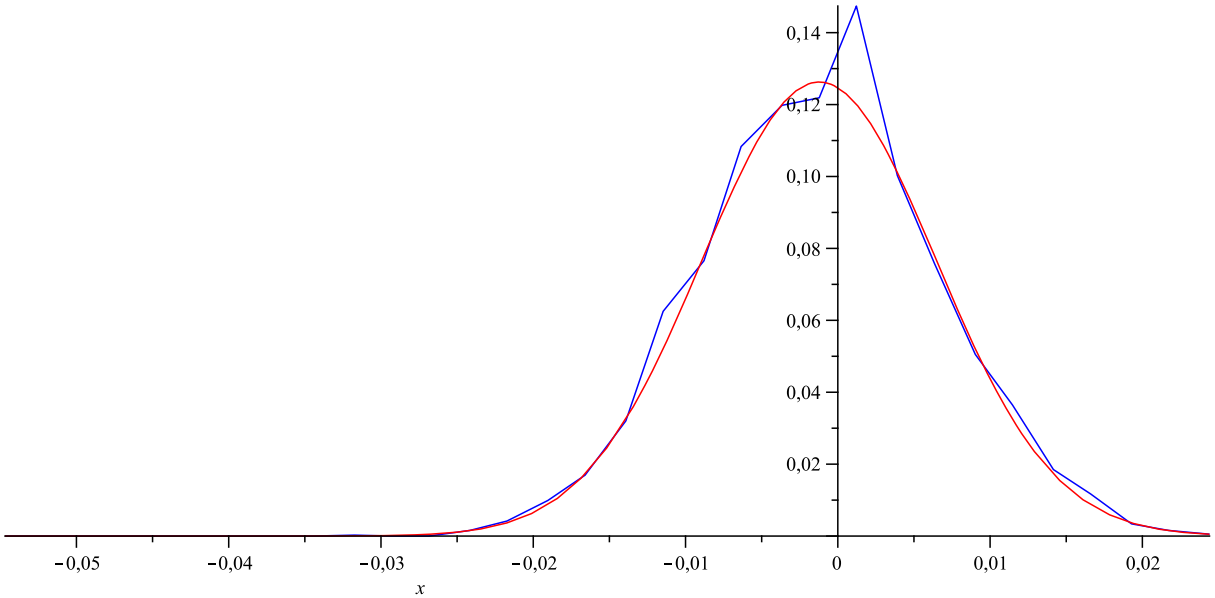
*Given a weak-key there exist linear approximations with bias* $\geq 2^{-17}$ *for any number of rounds.*

**Theorem (Correlation of Linear Approximations)**

$$C = \sum_{U} (-1)^{\langle U,K \rangle} |C_U|$$

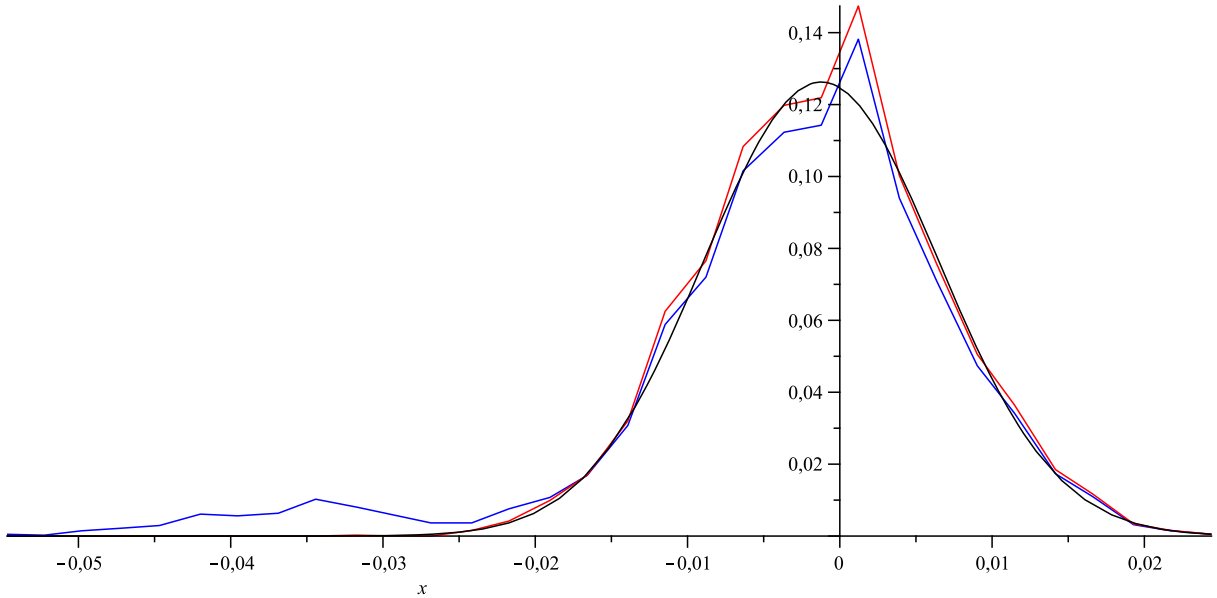
$\Rightarrow$ Signs do not behave nicely!

# The Role of Key-Scheduling: PrintCIPHER




- PrintCipher-12
- Non-weak keys

# The Role of Key-Scheduling: PrintCIPHER




- PrintCipher-12
- weak and non-weak keys

# The Role of Key-Scheduling: PrintCIPHER

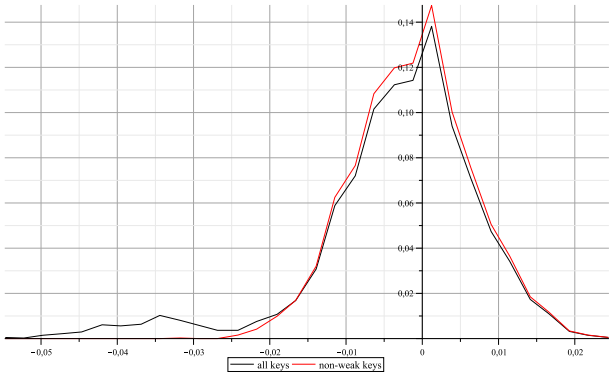



Figure: PrintCipher-24

Conclusions:

- Somewhat normal-distribution
- except for weak-keys
- More rounds do not help.

# Why is that?

**Theorem**

*Invariant Subspace*

$$\Rightarrow$$

*Sub-matrix of the correlation matrix has a eigenvector with eigenvalue* 1*.*

Consequence:

- This matrix has a non-zero limit.
- Trail clustering for any number of rounds.

# Conclusion

## Summary: Invariant Subspace Attack

- Weak keys for full PRINTCIPHER-48 and PRINTCIPHER-96
- Strange behavior of differential characteristics
- Similar observation for linear attacks

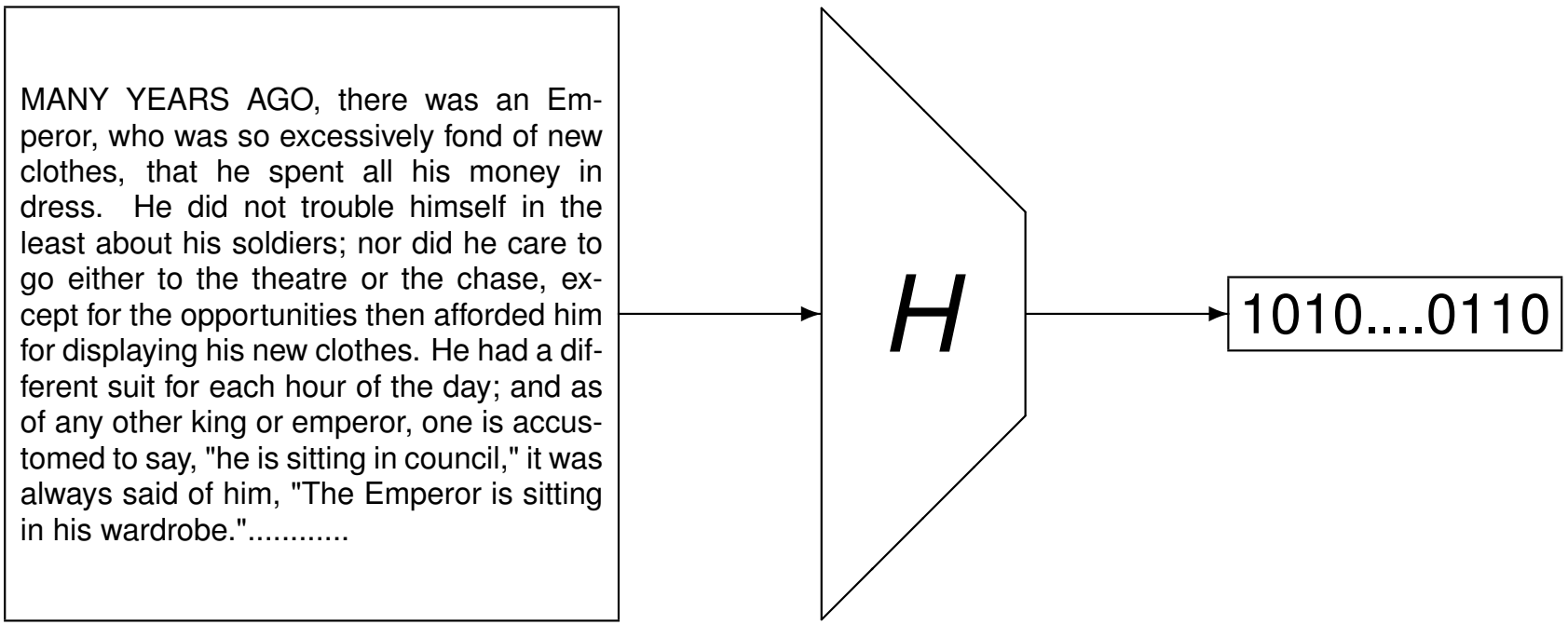
## Note

A symmetry is a special case of an invariant subspace

- Hard to prove the non-existence
- Unlikely to exist
- Especially with a more complex key scheduling

# Outline

# Definition - hash function




Any input lenght $\mapsto$ fixed output length

# Definition - more

## Definition (Hashfunction)

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

- Any input length.
- Fixed output length $n$, e.g. $n = 160$.
- no secret parameters.
- given $x$, easy/efficient to compute $H(x)$

# Cryptographic hash functions

**Remark**

Hash functions play a crucial role in cryptography

Some Applications:

- Used in digital signatures.
- Used for password security.
- Used in many cryptograhic protocols.

# Cryptographic hash functions

**Question**

What should a hash function provide?

Depends on the application:

- Should appear to be one-to-one in practice
- Should appear to be totally random (Random Oracle).
- Should be hard to invert.
- Should be hard to find collisions.
- ...

Sometimes it is not even clear.

# Random Oracle

$H : \{0,1\}^* \rightarrow \{0,1\}^n$ totally random.

## Random Oracle Model

Think about the hash function as an oracle.

- The oracle has a list of already queried values $x_i$ and the responses $y_i$,that is $(x_i, y_i)$.
- Given a query $x$ the oracle checks if the value is in the list.
- If yes: send the same answer again.
- If no: choose a random answer, update the list.

# Random Oracle

$H : \{0,1\}^* \rightarrow \{0,1\}^n$ totally random.

## Properties of a Random Oracle

- The perfect hash function.
- Is good enough for any application.
- Makes proofs possible/simpler.
- Does not exist.

# Properties

$H : \{0,1\}^* \rightarrow \{0,1\}^n$, for fixed value of $n$

**Definition (Preimage resistance)**

Given $H(x)$, hard to find $x'$, s.t. $H(x) = H(x')$

**Definition (2nd preimage resistance)**

Given $x$, hard to find $x' \neq x$, s.t. $H(x) = H(x')$

**Definition (Collision resistance)**

Hard to find $x$ and $x'$, such that $x \neq x'$ and $H(x) = H(x')$.

# Brute Force Attacks

## Generic Attacks

An attack is called generic if it can be applied independent of the details of a hashfunction/cipher

- One can never avoid these attacks
- Security goal: There are not better attacks.

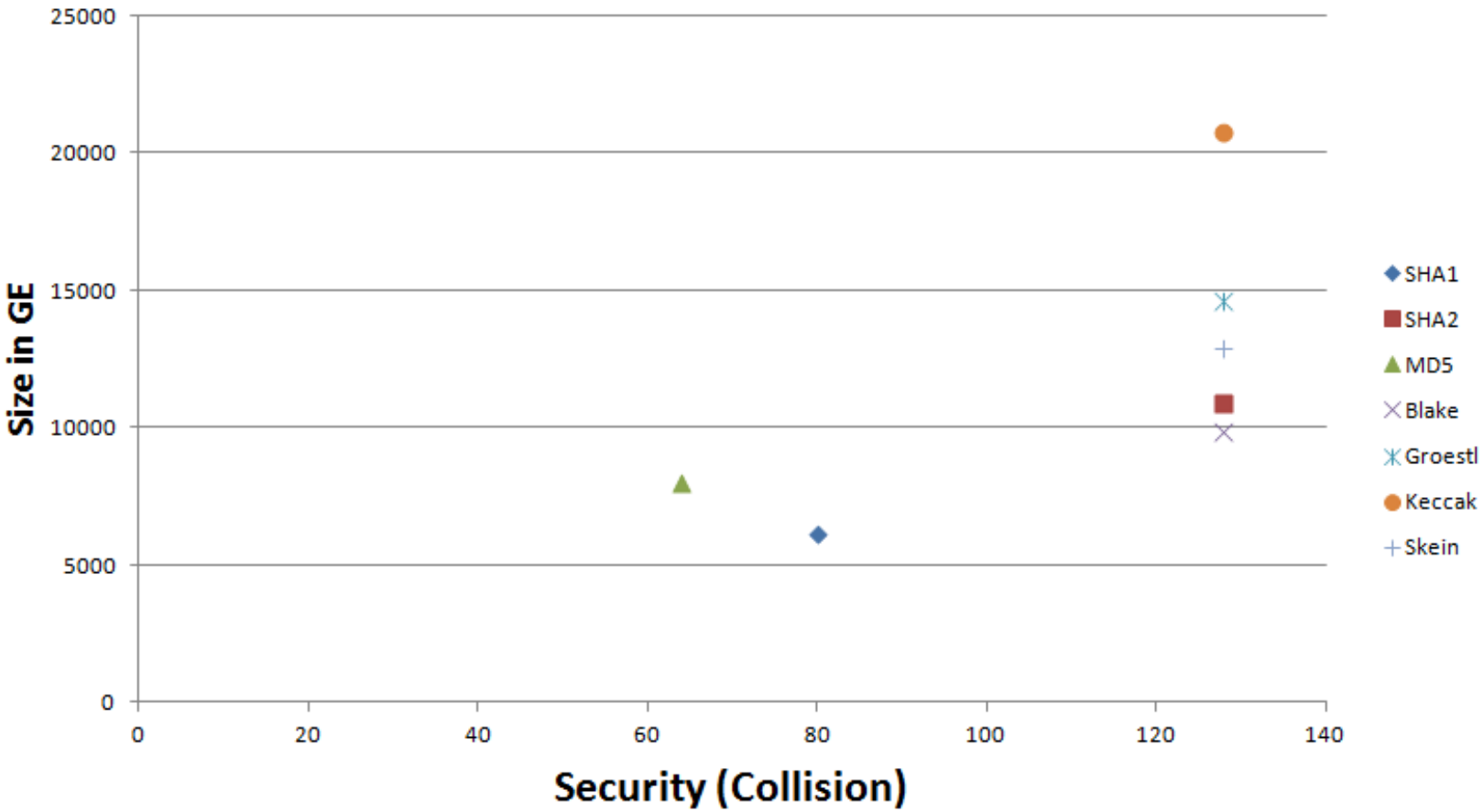
# Cryptographic hash functions - generic attacks

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

| attack | rough complexity |
|---|---|
| collision | $\sqrt{2^n} = 2^{n/2}$ |
| 2nd preimage | $2^n$ |
| preimage | $2^n$ |

- Today: $n \geq 128$ is recommended
- Security Goal: no better attacks than generic attacks

# Standard Hash Functions in HW




## Conclusion

Too Big!

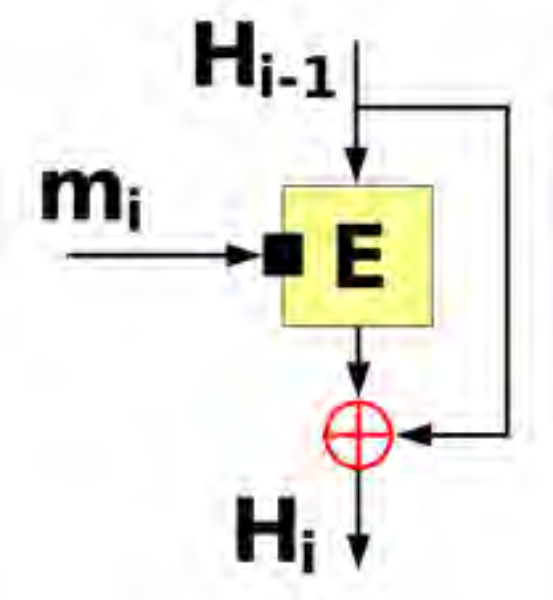
# Lightweight Hash-Function Design (I)

## DM-Construction

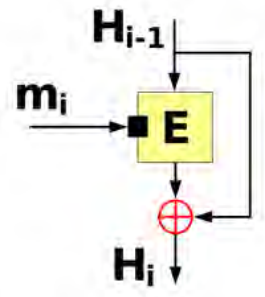
Use a block cipher

$$H_i = E_{m_i}(H_i)$$

$$H_0 = IV \quad H_k = \text{output}$$



# Lightweight Hash-Function Design (II)




Not very hardware friendly:

- Store $H_i$ (feed forward)
- Store state
- Store message

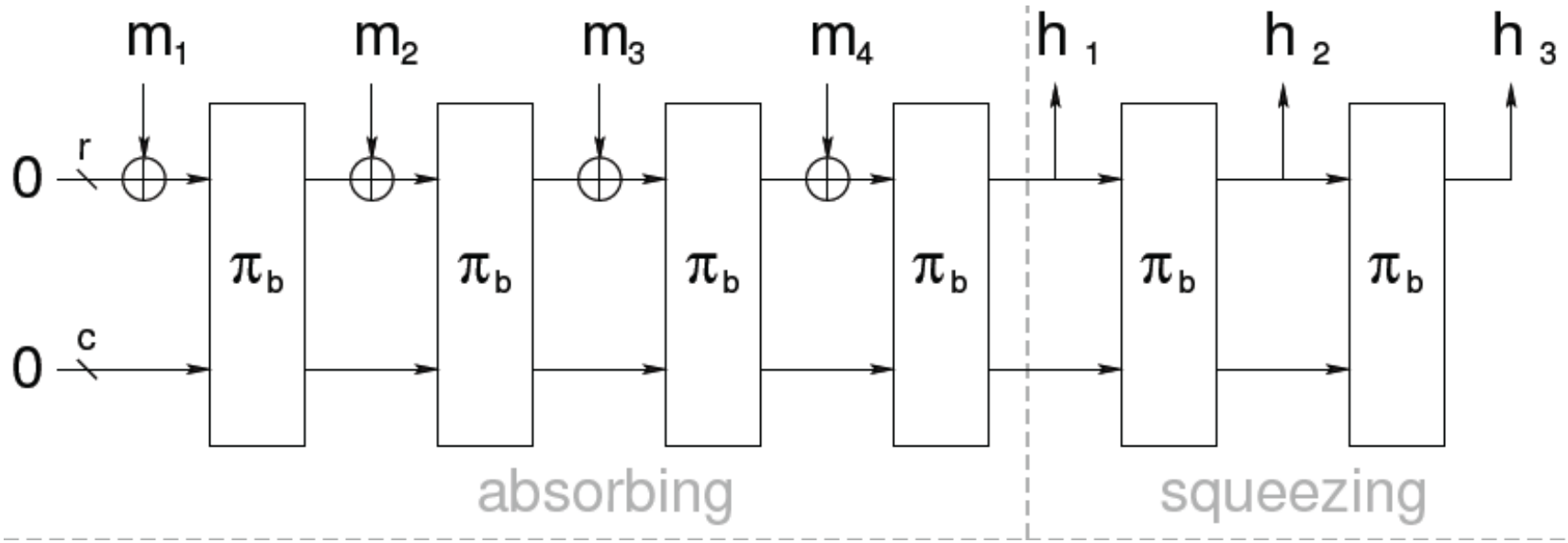
$2n + k$ bits storage!

# Lightweight Hash-Function Design (III)

## Lesson Learnt

From block cipher: Minimize state!

More Promising: Sponge Construction (Bertoni et al. 2007)




$c + r$ bits storage!

# Lightweight Hash-Functions: Examples

Three Examples (Sponge Based)

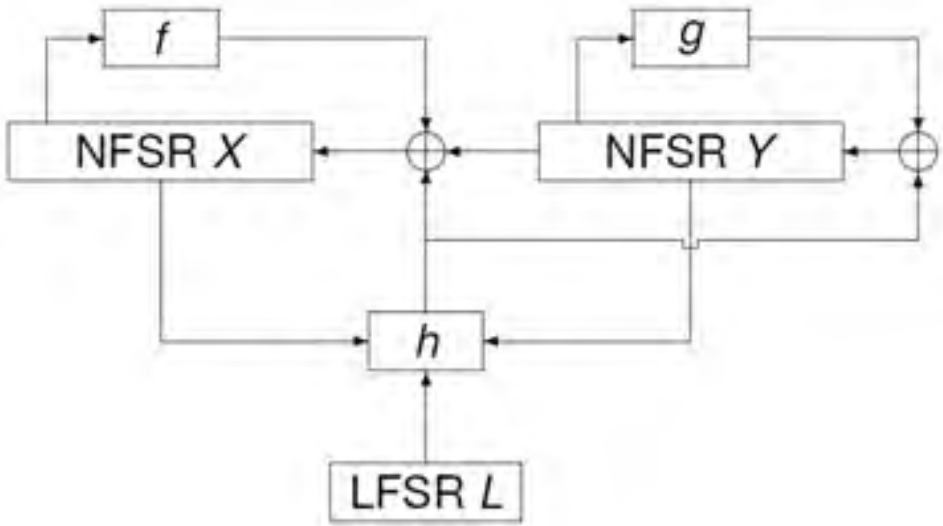
- Quark
- PHOTON
- Spongent

# Quark

## Quark

Designed by Aumasson et al. (CHES 2010)

- First real lightweight hash function
- Grain/Katan inspired

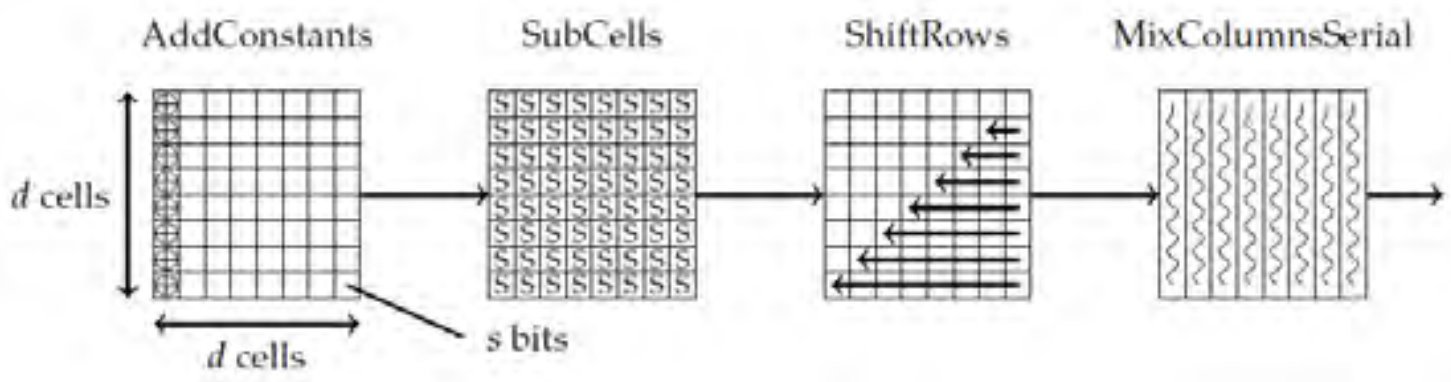


# PHOTON

## PHOTON

Designed by Guo, Peyrin, Poschmann (CRYPTO 2011)

- Nice observations on trade-offs
- AES inspired
- Similar to LED (LED is PHOTON inspired)

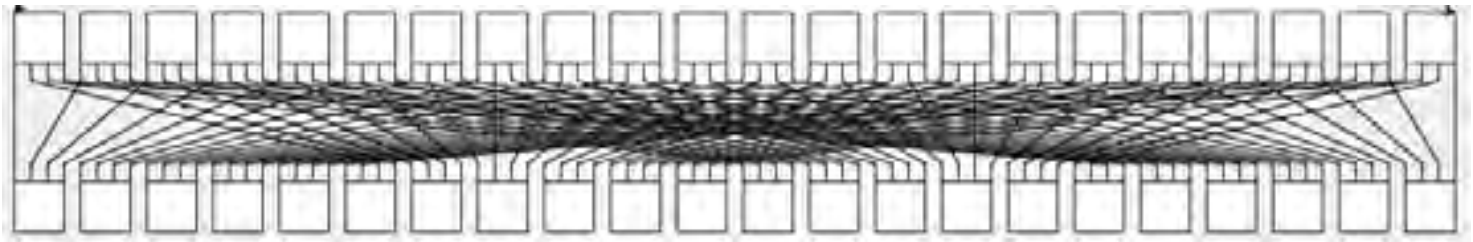


# SPONGENT

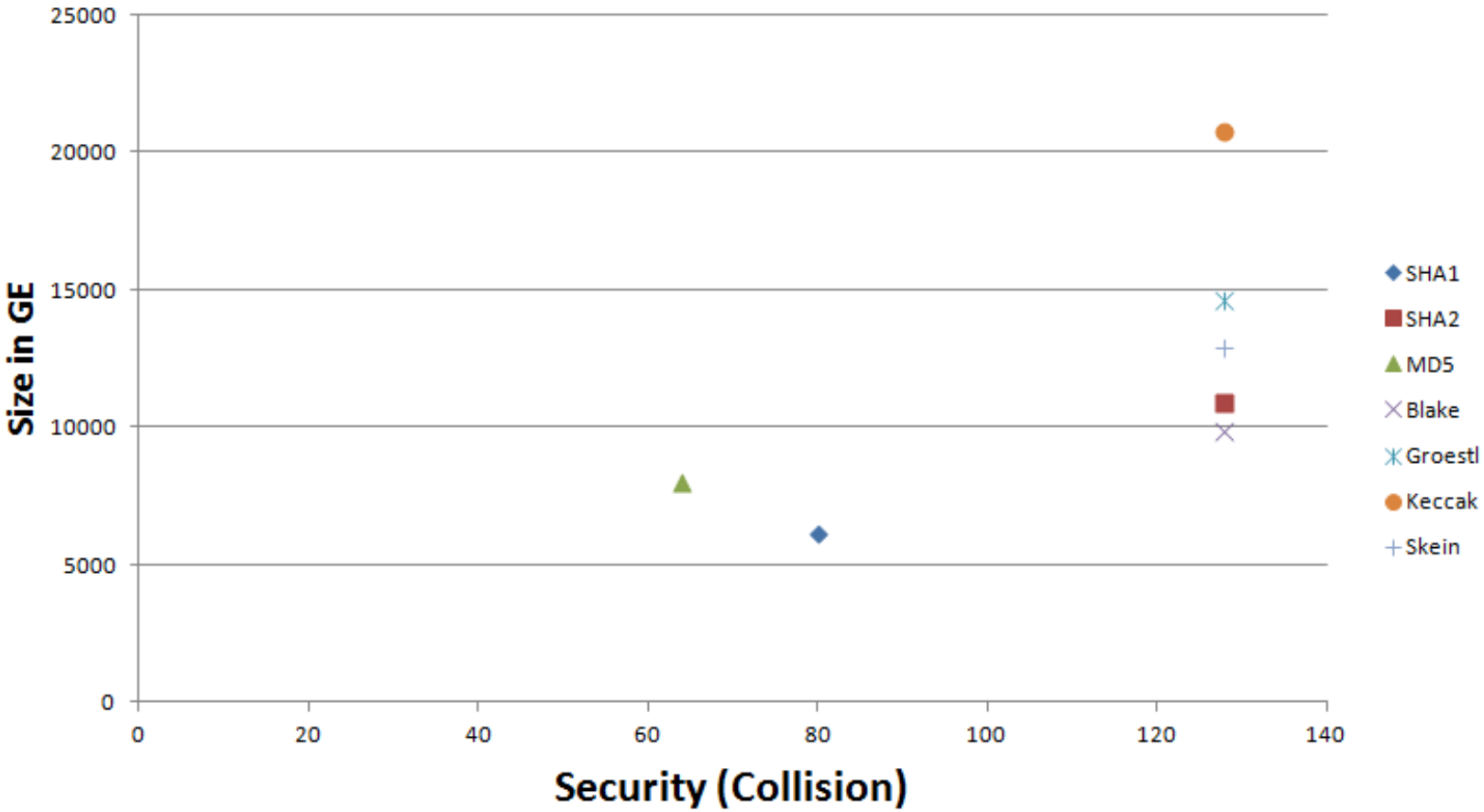
## SPONGENT

Designed by Bogdanov et al. (CHES 2011)

- PRESENT inspired
- smallest so far !?



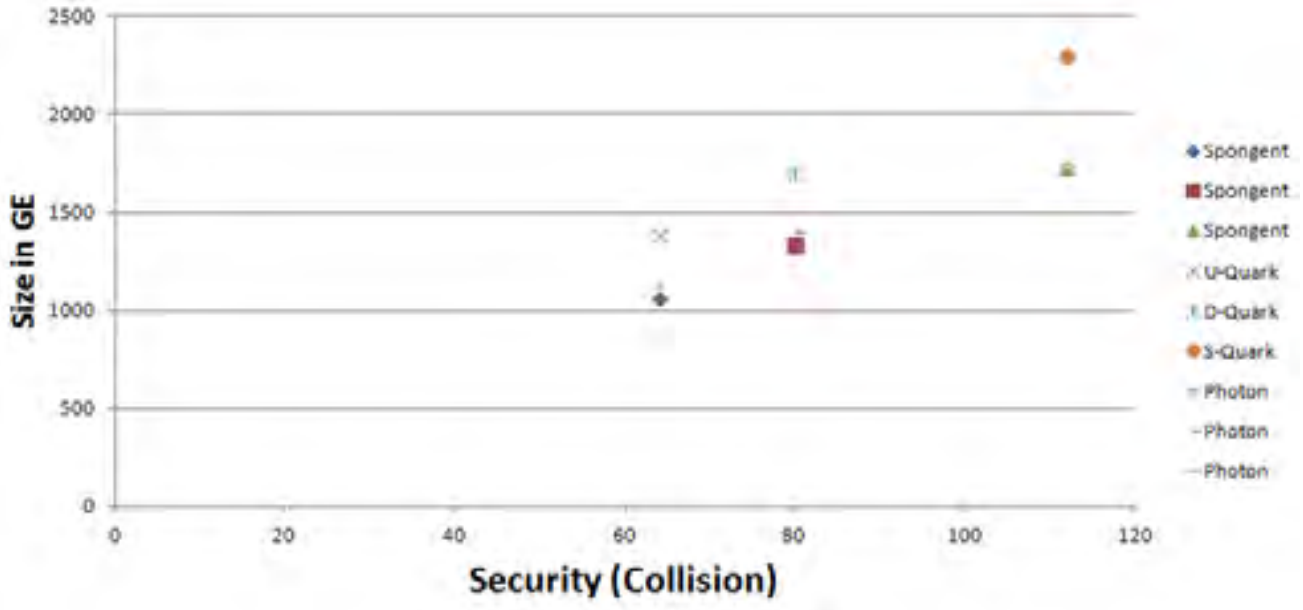
# Recall: Standard Hash Functions in HW




## Question

Did it get smaller?

# Lightweight Hash Functions in HW




## Careful: A fair comparison is difficult

- Technology depended
- Speed is ignored here
- Sponge allows many tradeoffs

# Outline

# Conclusions

## Challenging research area

- Calls for new ideas
- Many interesting proposals available
- Inter-disciplinary research
- Chance to be applied
- Key-scheduling design non-trivial

# The Future of Lightweight Cryptography (I)

- Optimize with respect to other criteria
  - Latency
  - Throughput
  - Energy efficiency
  - Code size
  - ...
- A specific combination

**Domain Specific Block Ciphers**

Design Ciphers for a specific combination of performance criteria.

Better: Design a small set of parameterizable cipher meeting many possible sets of criteria.

# The Future of Lightweight Cryptography (II)

Other promising topics include

- include side channel resistance in design
- Unbalanced primitives

# Fin

Thanks a lot!