UNIVERSITÄT PASSAU

Attacks on Cryptographic Hardware

Why strong crypto is not enough

Philipp Jovanovic

Department of Informatics and Mathematics
University of Passau

January 28, 2013

Outline

1. The Hitchhiker's Guide to Implementation Attacks
2. Crypto, Block Ciphers and Everything
3. Mostly Harmful
4. And Another Thing ...
5. So Long, and Thanks for (all) the Keys

Outline

UNIVERSITÄT PASSAU

Definition

Definition


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!

Definition

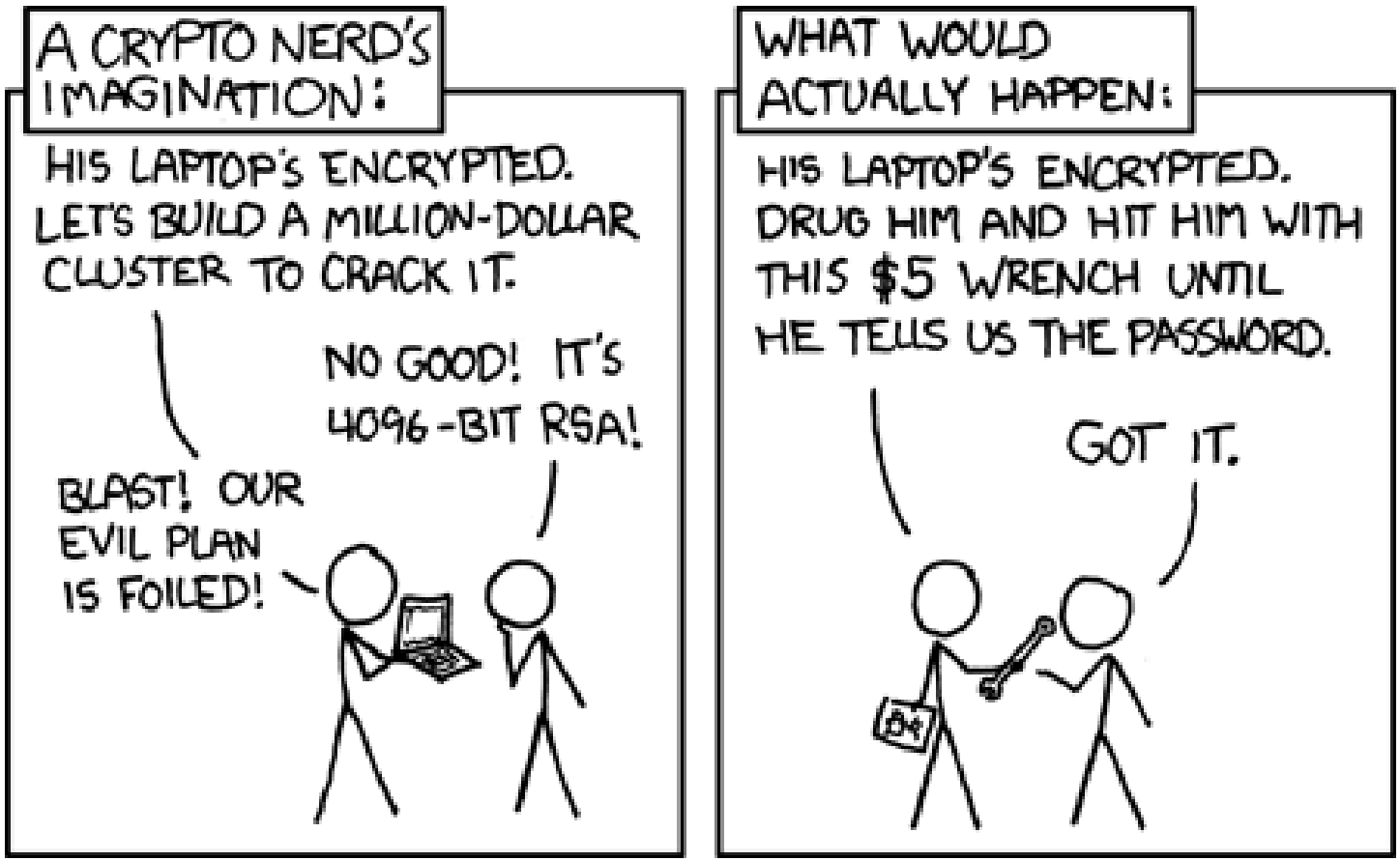


Figure: http://xkcd.com/538

Cryptology

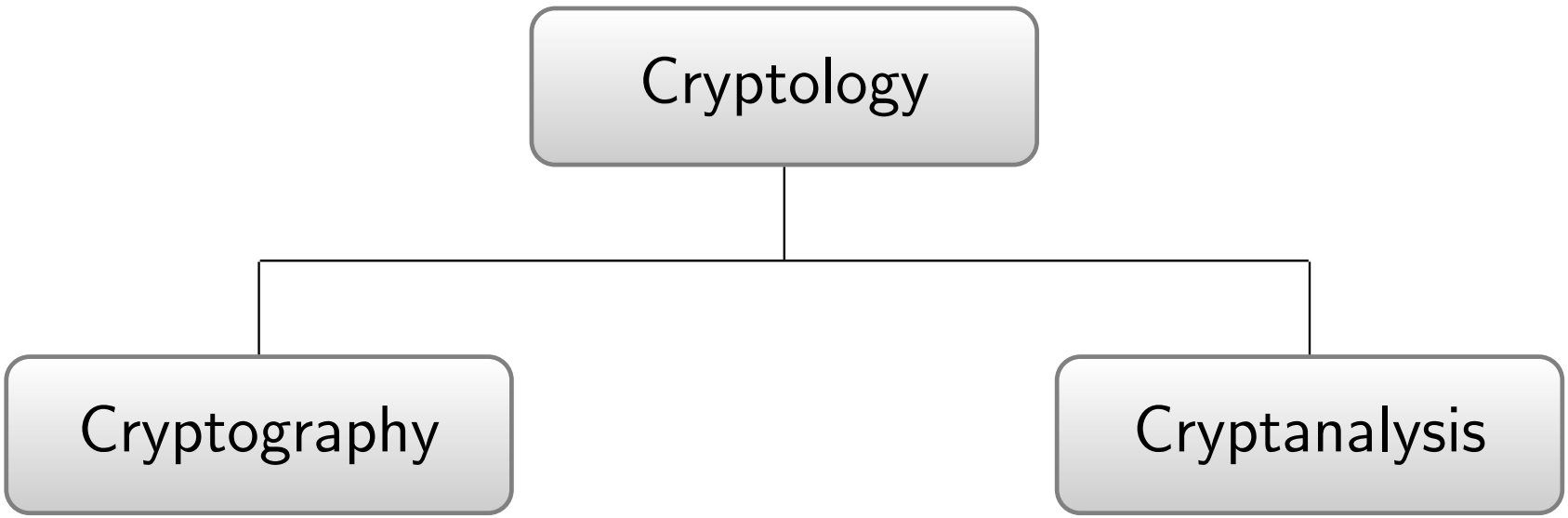


Figure: Overview of cryptology.

Fields of Cryptanalysis

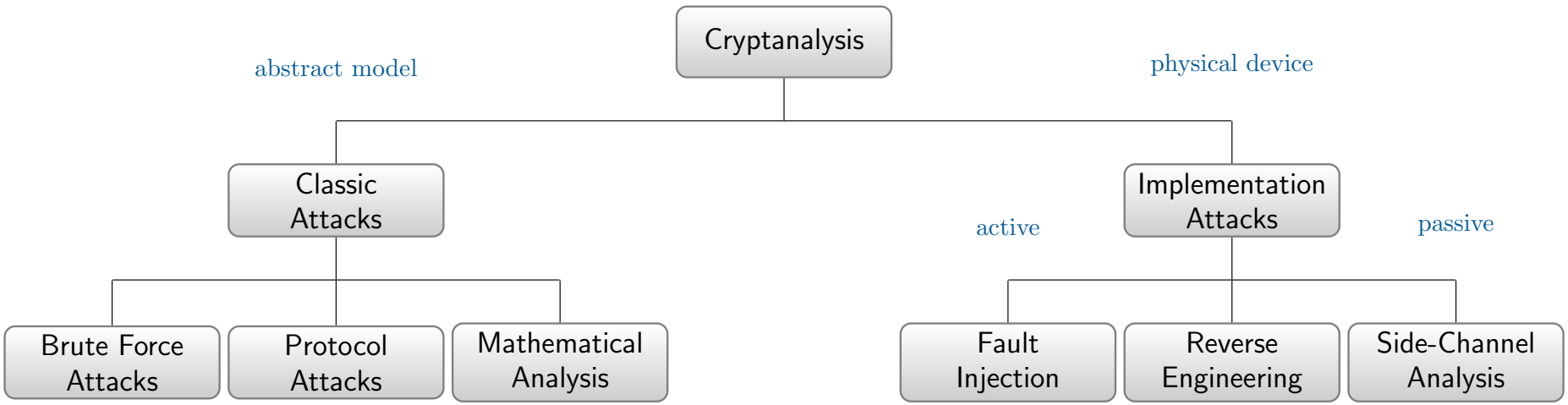


Figure: Overview of the different fields of cryptanalysis.

Implementation Attacks

UNIVERSITÄT PASSAU

Definition

- An *implementation attack* is a technique where additional informations gained from the physical implementation of a cryptosystem are used during cryptanalysis.
- In other words: An attacker is not trageting directly the encryption (or decryption) map, but rather it's implementation.

UNIVERSITÄT PASSAU

Definition

- An *implementation attack* is a technique where additional informations gained from the physical implementation of a cryptosystem are used during cryptanalysis.
- In other words: An attacker is not trageting directly the encryption (or decryption) map, but rather it's implementation.

Motivation

Why shoud we bother?

- Many fast growing fields for embedded applications: RFID, sensor networks, “Internet of Things” ...
- Areas of interest e.g. public transportation, communication, healthcare, car industry, banking sector, military.
- Drastic increase in the importance of hardware security and the demand for secure chips.
- Hardware implementing cryptographic functions (including smartcards) often show severe vulnerabilities.

Motivation

UNIVERSITÄT PASSAU

Why shoud we bother?

- Many fast growing fields for embedded applications: RFID, sensor networks, “Internet of Things” ...
- Areas of interest e.g. public transportation, communication, healthcare, car industry, banking sector, military.
- Drastic increase in the importance of hardware security and the demand for secure chips.
- Hardware implementing cryptographic functions (including smartcards) often show severe vulnerabilities.

Motivation

UNIVERSITÄT PASSAU

Why shoud we bother?

- Many fast growing fields for embedded applications: RFID, sensor networks, “Internet of Things” ...
- Areas of interest e.g. public transportation, communication, healthcare, car industry, banking sector, military.
- Drastic increase in the importance of hardware security and the demand for secure chips.
- Hardware implementing cryptographic functions (including smartcards) often show severe vulnerabilities.

Motivation

UNIVERSITÄT PASSAU

Why shoud we bother?

- Many fast growing fields for embedded applications: RFID, sensor networks, “Internet of Things” ...
- Areas of interest e.g. public transportation, communication, healthcare, car industry, banking sector, military.
- Drastic increase in the importance of hardware security and the demand for secure chips.
- Hardware implementing cryptographic functions (including smartcards) often show severe vulnerabilities.

UNIVERSITÄT PASSAU

"Milking the Digital Cash Cow."

- Presented at the *29th Chaos Communication Congress* by the crypto group of the Ruhr-University Bochum.
- Extract secret keys from NXP's *Mifare DESfire MF3ICD40 smartcards* by exploiting electro-magnetic information leakage.
- Those cards are used e.g. for the university's "Mensacards".
- After secret key extraction: Change stored amount of money to an arbitrary value.

Further details:

- `events.ccc.de/congress/2012/Fahrplan/events/5393.en.html`
- `www.hgi.rub.de/hgi/news/articles/milkingdigitalcashcow`
- Recordings of the talk are on youtube.

Cold Boot Attacks

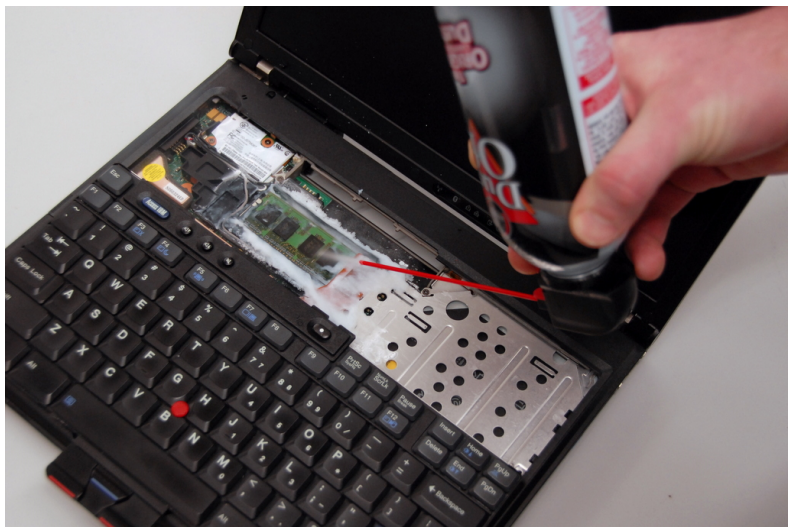


Figure: Freezing memory.

- Reconstruct full disk encryption keys from a memory dump extracted from frozen RAM.
- Further details: `https://citp.princeton.edu/research/memory`

Observation

- Power consumption of a circuit varies according to the activity of its individual components.
- Measurements of the power consumption contain informations about operations being performed and the data being processed.

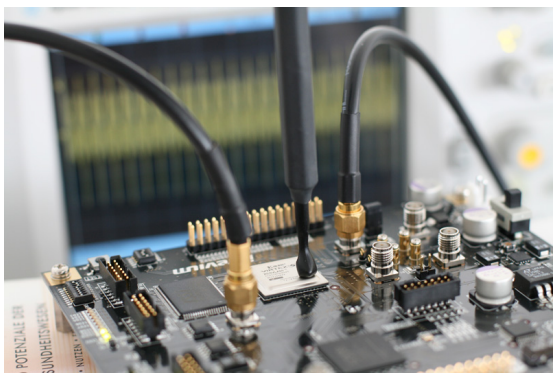


Figure: Power analysis setup.

Idea: Exploit (minimal) *statistic significances* appearing in the collected power traces to extract the secret key from a device.

Power Analysis

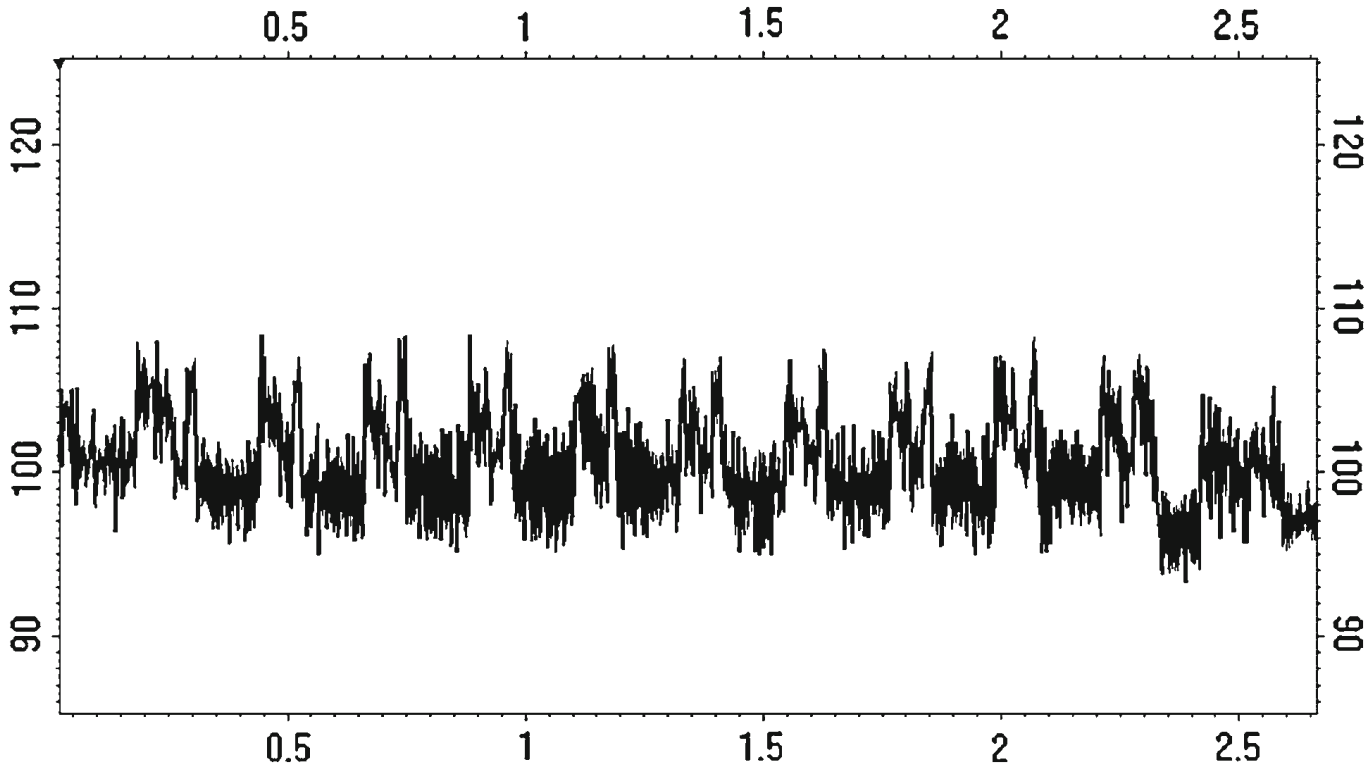


Figure: Power trace from a smart card performing an AES-128 encryption, with ten rounds clearly visible.

UNIVERSITÄT PASSAU

Some characteristics

- Attacks are practical, non-invasive and highly effective.
- Applicable even against complex and noisy systems where cryptographic computations account only for a small fraction of the overall power consumption.

UNIVERSITÄT PASSAU

Some characteristics

- Attacks are practical, non-invasive and highly effective.
- Applicable even against complex and noisy systems where cryptographic computations account only for a small fraction of the overall power consumption.

Characteristics

- Introduced in 2001 / 2002.
- By injecting faults into the electronical circuit, the attacker tries to find out someting about the stored secret.
- Lead to powerful new attack techniques and chip manufacturers were forced to rethink their designs.

Characteristics

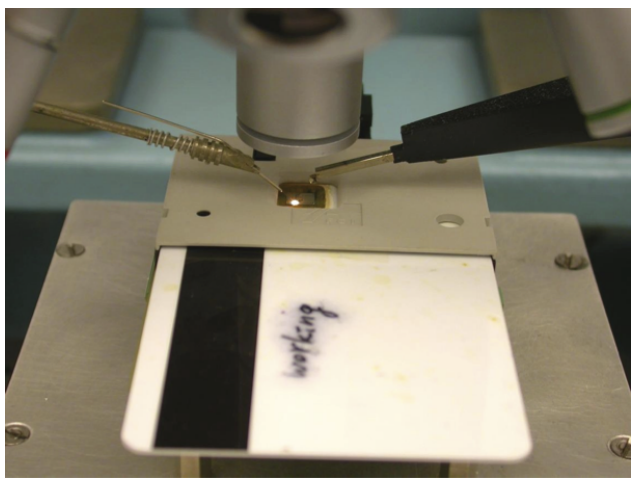
- Introduced in 2001 / 2002.
- By injecting faults into the electronical circuit, the attacker tries to find out someting about the stored secret.
- Lead to powerful new attack techniques and chip manufacturers were forced to rethink their designs.

Characteristics

- Introduced in 2001 / 2002.
- By injecting faults into the electronical circuit, the attacker tries to find out someting about the stored secret.
- Lead to powerful new attack techniques and chip manufacturers were forced to rethink their designs.

Techniques to induce faults

- Manipulation of the power-supply voltage to cause miscalculations.
- Manipulation of the circuit's clock.
- Parasitic charge-carrier generation by a laser beam.



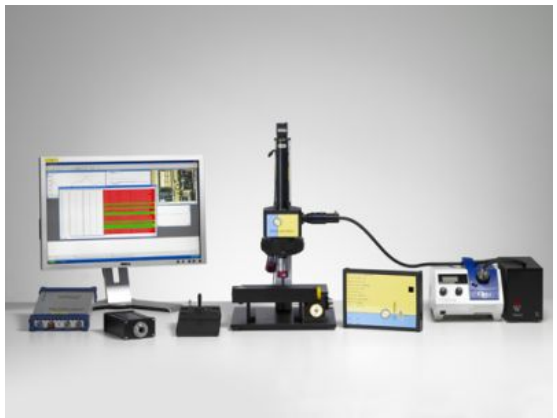


Figure: www.riscure.com

Related: Glitch attacks to break pay-tv chips or gaming consoles (PS3, XBOX).

Other types

- Timing Analysis.
- Reverse Engineering: Extraction and cryptanalysis of proprietary encryption algorithms (e.g. A5/1 stream cipher, which was used in the GSM cellular telephone standard).
- Microprobing.
- Imaging.
- Photonic Analysis: Measure photons that are emitted during the computations of a circuit and thereby extract secret informations.

Other types

- Timing Analysis.
- Reverse Engineering: Extraction and cryptanalysis of proprietary encryption algorithms (e.g. A5/1 stream cipher, which was used in the GSM cellular telephone standard).
- Microprobing.
- Imaging.
- Photonic Analysis: Measure photons that are emitted during the computations of a circuit and thereby extract secret informations.

Other types

- Timing Analysis.
- Reverse Engineering: Extraction and cryptanalysis of proprietary encryption algorithms (e.g. A5/1 stream cipher, which was used in the GSM cellular telephone standard).
- Microprobing.
- Imaging.
- Photonic Analysis: Measure photons that are emitted during the computations of a circuit and thereby extract secret informations.

Other types

- Timing Analysis.
- Reverse Engineering: Extraction and cryptanalysis of proprietary encryption algorithms (e.g. A5/1 stream cipher, which was used in the GSM cellular telephone standard).
- Microprobing.
- Imaging.
- Photonic Analysis: Measure photons that are emitted during the computations of a circuit and thereby extract secret informations.

UNIVERSITÄT PASSAU

Other types

- Timing Analysis.
- Reverse Engineering: Extraction and cryptanalysis of proprietary encryption algorithms (e.g. A5/1 stream cipher, which was used in the GSM cellular telephone standard).
- Microprobing.
- Imaging.
- Photonic Analysis: Measure photons that are emitted during the computations of a circuit and thereby extract secret informations.

Outline

Warning

The next slides may contain traces of mathematics.

Warning

UNIVERSITÄT PASSAU

The next slides may contain traces of mathematics.

Don’t Panic

UNIVERSITÄT PASSAU

Definition

Given a block size of n_1 bits and a key size of n_2 bits a block cipher is specified by an encryption function

$$E : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^{n_1}, (m,k) \mapsto c$$

and a decryption function

$$D : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^{n_1}, (c,k) \mapsto m$$

such that

$$D_k(E_k(m)) = m$$

for all plaintext messages $m \in \{0,1\}^{n_1}$ and all keys $k \in \{0,1\}^{n_2}$.

UNIVERSITÄT PASSAU

Common designs

- Feistel Network
- Addition-Rotation-XOR Network (ARX)
- Substitution Permutation Network (SPN)

Overview

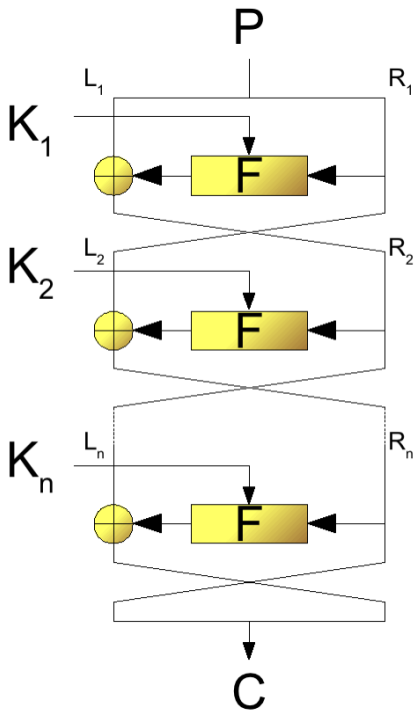


Figure: Structure of Feistel Ciphers.

Basic procedure:

- State block splitted into: L_i, R_i
- $L_{i+1} = R_i$
- $R_{i+1} = L_i \oplus F(k_i, R_i)$

Prominent examples: DES, GOST, Blowfish, Twofish, FEAL, ...

Overview

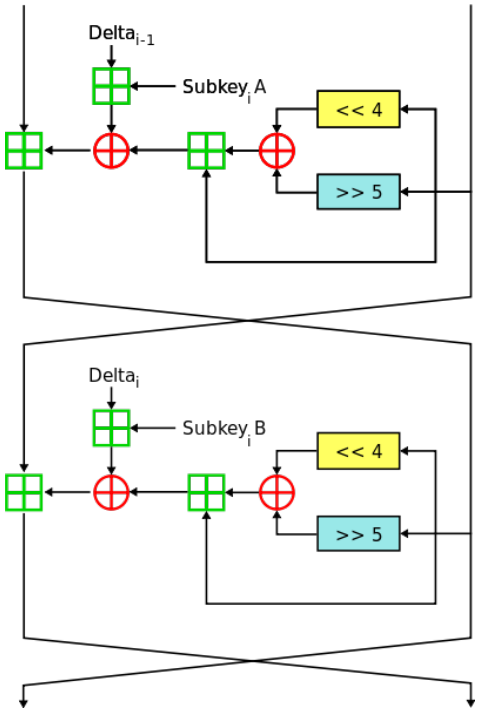


Figure: Example ARX Cipher: XTEA

Operations:

- **A**ddition (mod 2^n): $\boxplus$
- **R**otation: $\lll r$
- **X**OR: $\oplus$

Prominent examples: RC{4,5,6}, TEA, XTEA, Threefish, ...

Overview

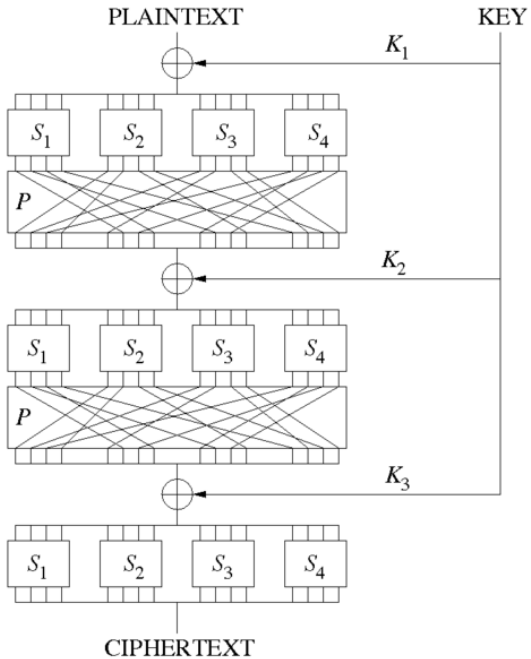


Figure: Substitution Permutation Network

Operations:

- Substitution: SBox application.
- Permutation.

Prominent examples: AES, PRESENT, KHAZAD, LED, PRINCE ...

The LED Block Cipher (∗)

UNIVERSITÄT PASSAU

General features

- Uses a 64-bit state and either a 64-bit or a 128-bit key, no key schedule.
- Number of encryption rounds: 32 or 48.
- Based on a Substitution-Permutation-Network (SPN).
- Design oriented on AES and PRESENT.
- Small silicone footprint: Especially designed for resource-constrained environments like smart cards, sensor networks etc.

(∗) J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.

The LED Block Cipher (∗)

UNIVERSITÄT PASSAU

General features

- Uses a 64-bit state and either a 64-bit or a 128-bit key, no key schedule.
- Number of encryption rounds: 32 or 48.
- Based on a Substitution-Permutation-Network (SPN).
- Design oriented on AES and PRESENT.
- Small silicone footprint: Especially designed for resource-constrained environments like smart cards, sensor networks etc.

(∗) J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.

The LED Block Cipher (∗)

UNIVERSITÄT PASSAU

General features

- Uses a 64-bit state and either a 64-bit or a 128-bit key, no key schedule.
- Number of encryption rounds: 32 or 48.
- Based on a Substitution-Permutation-Network (SPN).
- Design oriented on AES and PRESENT.
- Small silicone footprint: Especially designed for resource-constrained environments like smart cards, sensor networks etc.

(∗) J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.

The LED Block Cipher (∗)

UNIVERSITÄT PASSAU

General features

- Uses a 64-bit state and either a 64-bit or a 128-bit key, no key schedule.
- Number of encryption rounds: 32 or 48.
- Based on a Substitution-Permutation-Network (SPN).
- Design oriented on AES and PRESENT.
- Small silicone footprint: Especially designed for resource-constrained environments like smart cards, sensor networks etc.

(∗) J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.

The LED Block Cipher (∗)

UNIVERSITÄT PASSAU

General features

- Uses a 64-bit state and either a 64-bit or a 128-bit key, no key schedule.
- Number of encryption rounds: 32 or 48.
- Based on a Substitution-Permutation-Network (SPN).
- Design oriented on AES and PRESENT.
- Small silicone footprint: Especially designed for resource-constrained environments like smart cards, sensor networks etc.

(∗) J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, The LED Block Cipher, In: B. Preneel and T. Takagi (eds.) *CHES* 2011, LNCS, vol. **6917**, Springer, Heidelberg 2011, pp. 326–341.

Remark

All computations are done over the finite field:

$$\mathbb{F}_{16} = \mathbb{F}_2[x]/\langle f \rangle, \text{ with } f = x^4 + x + 1$$

Notation

In the following, we represent an element $g \in \mathbb{F}_{16}$, with $g = c_3x^3 + c_2x^2 + c_1x + c_0$ and $c_i \in \mathbb{F}_2$, by

$$\phi : g \longmapsto c_3 \parallel c_2 \parallel c_1 \parallel c_0$$

where $\parallel$ denotes the concatenation of bits.

Example

The polynomial $x^3 + x + 1$ has the coefficient vector $(1, 0, 1, 1)$ and is therefore mapped to the bit string `1011`. Hexadecimal shortform: `1011` = `B`.

UNIVERSITÄT PASSAU

Remark

All computations are done over the finite field:

$$\mathbb{F}_{16} = \mathbb{F}_2[x]/\langle f \rangle, \text{ with } f = x^4 + x + 1$$

Notation

In the following, we represent an element $g \in \mathbb{F}_{16}$, with $g = c_3x^3 + c_2x^2 + c_1x + c_0$ and $c_i \in \mathbb{F}_2$, by

$$\phi : g \longmapsto c_3 \parallel c_2 \parallel c_1 \parallel c_0$$

where $\parallel$ denotes the concatenation of bits.

Example

The polynomial $x^3 + x + 1$ has the coefficient vector $(1, 0, 1, 1)$ and is therefore mapped to the bit string `1011`. Hexadecimal shortform: `1011` = `B`.

UNIVERSITÄT PASSAU

Remark

All computations are done over the finite field:

$$\mathbb{F}_{16} = \mathbb{F}_2[x]/\langle f \rangle, \text{ with } f = x^4 + x + 1$$

Notation

In the following, we represent an element $g \in \mathbb{F}_{16}$, with $g = c_3x^3 + c_2x^2 + c_1x + c_0$ and $c_i \in \mathbb{F}_2$, by

$$\phi : g \longmapsto c_3 \parallel c_2 \parallel c_1 \parallel c_0$$

where $\parallel$ denotes the concatenation of bits.

Example

The polynomial $x^3 + x + 1$ has the coefficient vector $(1, 0, 1, 1)$ and is therefore mapped to the bit string `1011`. Hexadecimal shortform: `1011` = `B`.

The state

The 64-bit state s is considered as a concatenation of 16 4-bit strings $s_1 \parallel s_2 \parallel \cdots \parallel s_{15} \parallel s_{16}$ (with $s_i \in \mathbb{F}_{16}$) and conceptually arranged in a 4×4 matrix as written below.

$$s = \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ s_5 & s_6 & s_7 & s_8 \\ s_9 & s_{10} & s_{11} & s_{12} \\ s_{13} & s_{14} & s_{15} & s_{16} \end{pmatrix}$$

Remark

The 64-bit plaintext unit and the 64-/128-bit key are handled in the same way as the state above.

Layout

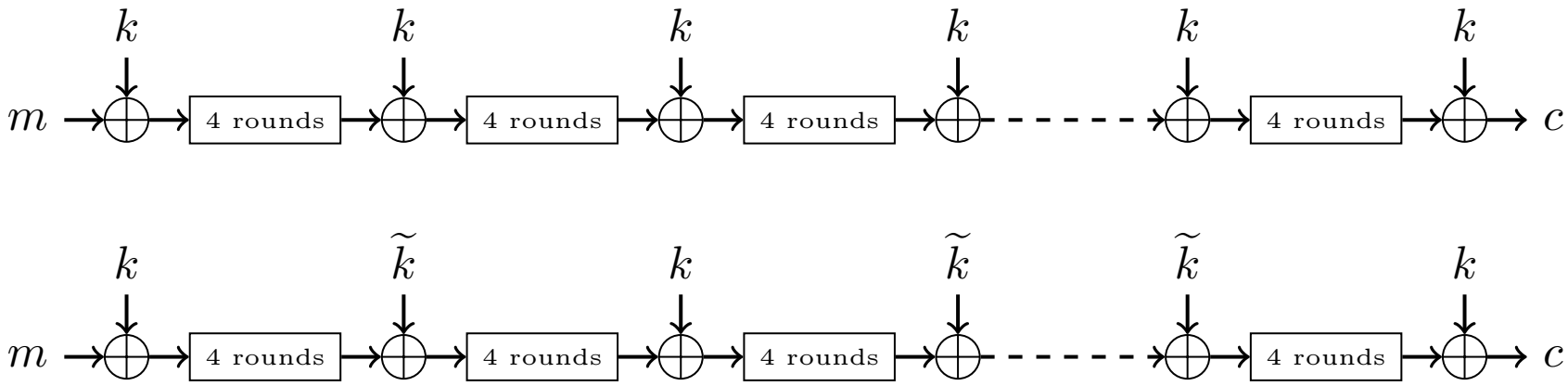


Figure: LED key usage: 64-bit key (top) and 128-bit key (bottom).

The LED Block Cipher

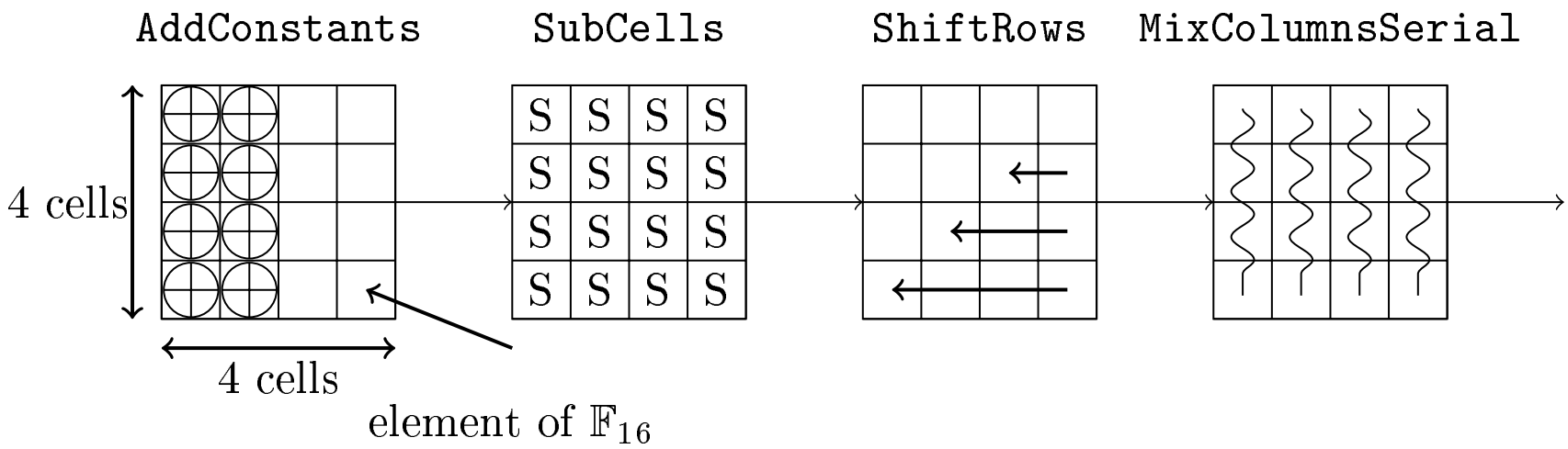


Figure: An overview of a single round of LED.

AddConstants

A round constant is a seq. of six bits $b_5 \,||\, b_4 \,||\, b_3 \,||\, b_2 \,||\, b_1 \,||\, b_0$. Divide it into $\mathtt{x} = b_5 \,||\, b_4 \,||\, b_3$ and $\mathtt{y} = b_2 \,||\, b_1 \,||\, b_0$, form the matrix

$$\begin{pmatrix} 0 & \mathtt{x} & 0 & 0 \\ 1 & \mathtt{y} & 0 & 0 \\ 2 & \mathtt{x} & 0 & 0 \\ 3 & \mathtt{y} & 0 & 0 \end{pmatrix}$$

and add it to the state.

Rounds	Constants
01 – 12	01,03,07,0F,1F,3E,3D,3B,37,2F,1E,3C
13 – 24	39,33,27,0E,1D,3A,35,2B,16,2C,18,30
25 – 36	21,02,05,0B,17,2E,1C,38,31,23,06,0D
37 – 48	1B,36,2D,1A,34,29,12,24,08,11,22,04

Table: The LED round constants.

SubCells

Replace each entry x of the state matrix by the element $S[x]$ from the SBox given by the table below.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table: The LED SBox.

ShiftRows

For $i = 1, 2, 3, 4$, the i-th row of the state matrix is shifted cyclically to the left by $i - 1$ positions.

$$\begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ s_5 & s_6 & s_7 & s_8 \\ s_9 & s_{10} & s_{11} & s_{12} \\ s_{13} & s_{14} & s_{15} & s_{16} \end{pmatrix} \longmapsto \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ s_6 & s_7 & s_8 & s_5 \\ s_{11} & s_{12} & s_9 & s_{10} \\ s_{16} & s_{13} & s_{14} & s_{15} \end{pmatrix}$$

MixColumns

Each column v of the state matrix is replaced by the product $M \cdot v$, where M is the matrix

$$M = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ \mathtt{B} & \mathtt{E} & \mathtt{A} & 9 \\ 2 & 2 & \mathtt{F} & \mathtt{B} \end{pmatrix}$$

Outline

Recall

Aim: By injecting faults into the circuit during it's computations, the attacker wants to gather enough informations to reconstruct somehow the chip's stored secret.

In our case

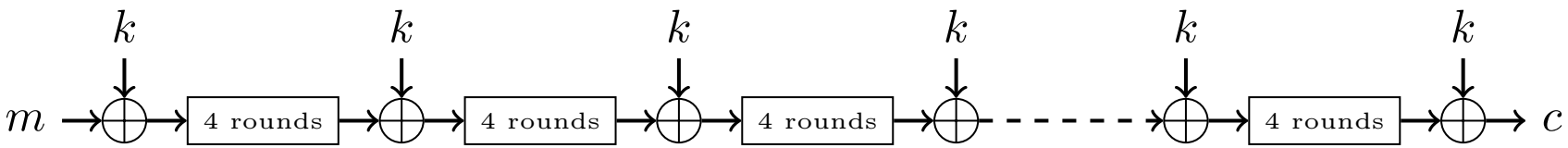


Figure: LED-64 key usage.

$\Rightarrow$ Reconstruction of k.

The Attack Model

What are the attacker's capabilities?

- Known plaintext attack: We assume that the attacker is able to generate an arbitrary number of plaintext, (faulty) ciphertext triples (m, c, c').
- Kerckhoffs Principle or "The enemy knows the system": The design of the cipher is known to the adversary. (No security by obscurity)

The Fault Model

Requirements

- Temporal resolution: Fault injection timing is controllable *quite precisely*, i.e. injection after a specific operation of the cipher.
- Spatial resolution: Injection effects a *single, random nibble* (4-bit value) of the whole state. The affected nibble itself is either **known** (Model 1) or **unknown** (Model 2).
- Effects: State nibble is changed to a *random and unknown 4-bit value*.

Note: We will focus on Model 1 in this talk.

A Fault Attack on LED-64

UNIVERSITÄT PASSAU

Outline of the attack

1. Simulate fault injection: Add a *random, unknown fault value* to the *first entry* (or any other) of the state matrix at the beginning of the *30-th round*.
2. Construct fault equations: Model the relation between correct and faulty ciphertext.
3. Filter key candidates: Evaluation of the fault equations.
4. Brute-force search: On the remaining keys.

UNIVERSITÄT PASSAU

Outline of the attack

1. Simulate fault injection: Add a *random, unknown fault value* to the *first entry* (or any other) of the state matrix at the beginning of the *30-th round*.
2. Construct fault equations: Model the relation between correct and faulty ciphertext.
3. Filter key candidates: Evaluation of the fault equations.
4. Brute-force search: On the remaining keys.

UNIVERSITÄT PASSAU

Outline of the attack

1. Simulate fault injection: Add a *random, unknown fault value* to the *first entry* (or any other) of the state matrix at the beginning of the *30-th round*.
2. Construct fault equations: Model the relation between correct and faulty ciphertext.
3. Filter key candidates: Evaluation of the fault equations.
4. Brute-force search: On the remaining keys.

UNIVERSITÄT PASSAU

Outline of the attack

1. Simulate fault injection: Add a *random, unknown fault value* to the *first entry* (or any other) of the state matrix at the beginning of the *30-th round*.
2. Construct fault equations: Model the relation between correct and faulty ciphertext.
3. Filter key candidates: Evaluation of the fault equations.
4. Brute-force search: On the remaining keys.

A Fault Attack on LED-64

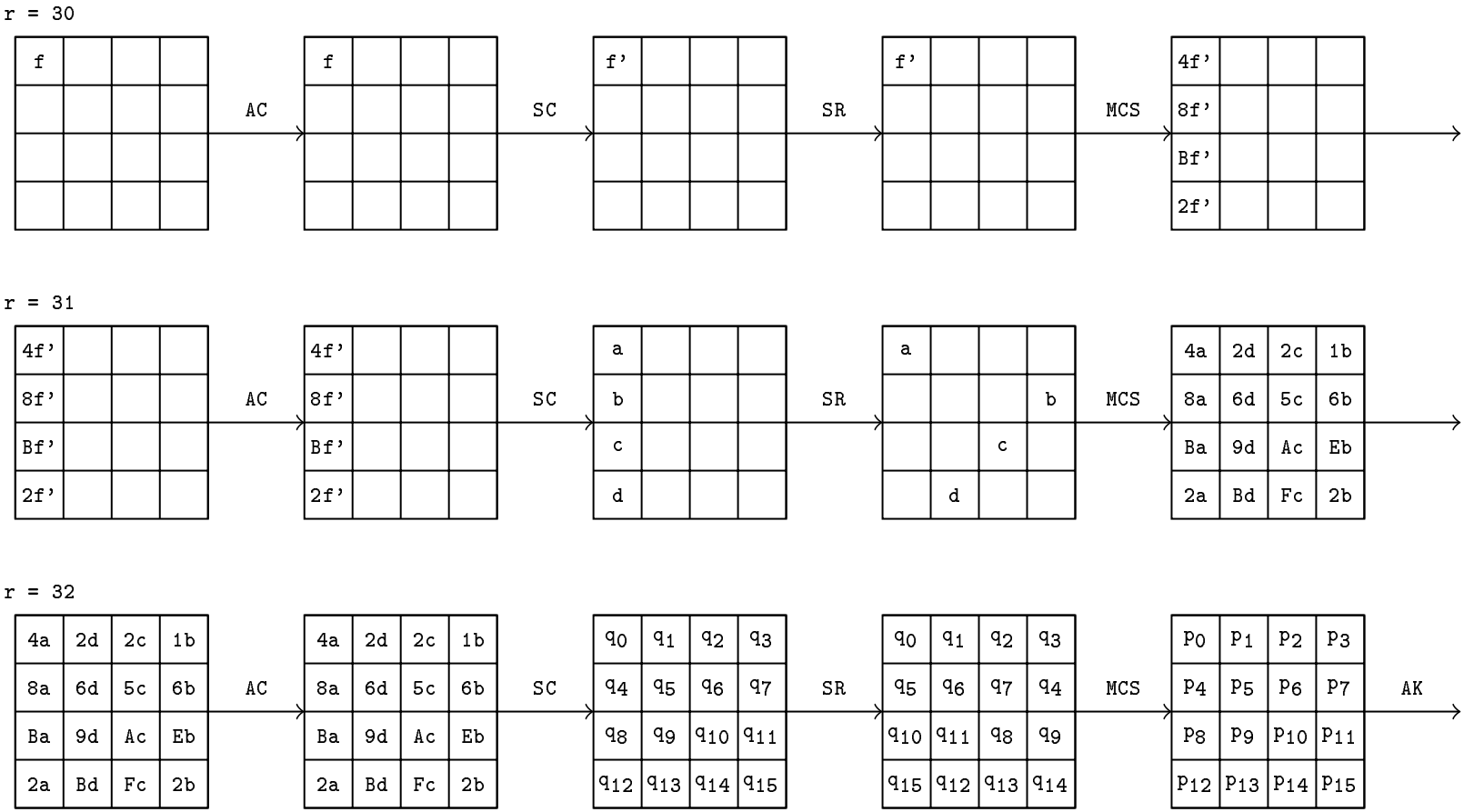
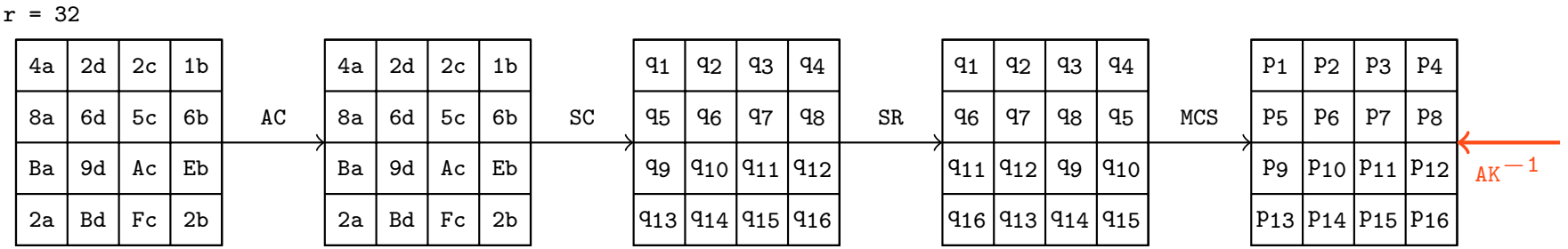


Figure: Fault propagation in the LED cipher.

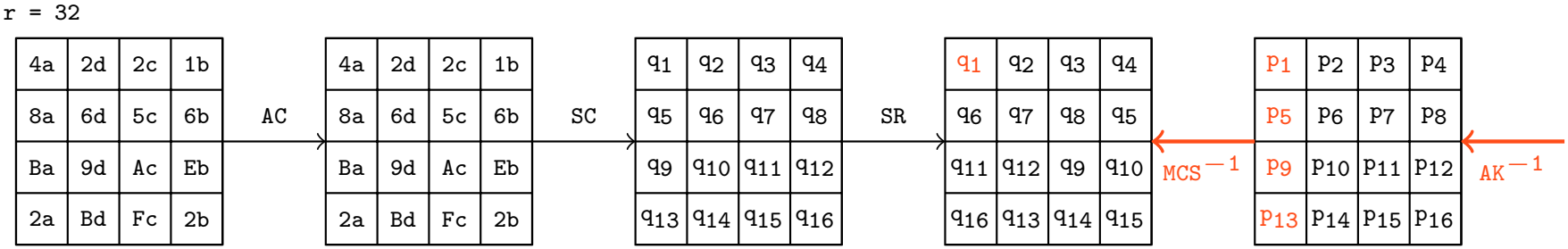
A Fault Attack on LED-64



Inversion of the LED steps

$\mathtt{AK}^{-1}$: Compute $c_i + k_i$ and $c'_i + k_i$, for $i \in \{1, \ldots, 16\}$.

A Fault Attack on LED-64



Inversion of the LED Steps

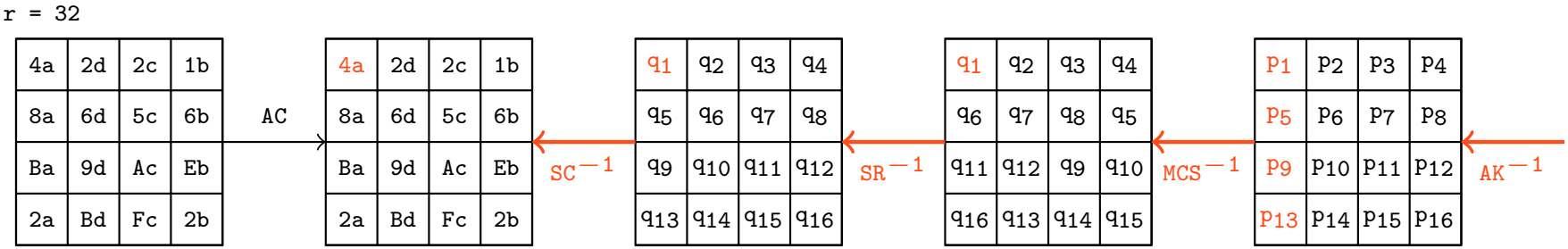
$\texttt{MCS}^{-1}$: Use the inverse matrix

$$M^{-1} = \begin{pmatrix} \texttt{C} & \texttt{C} & \texttt{D} & 4 \\ 3 & 8 & 4 & 5 \\ 7 & 6 & 2 & \texttt{E} \\ \texttt{D} & 9 & 9 & \texttt{D} \end{pmatrix}$$

of M from the $\texttt{MCS}$ operation to get:

$$\texttt{C} \cdot (c_1 + k_1) + \texttt{C} \cdot (c_5 + k_5) + \texttt{D} \cdot (c_9 + k_9) + 4 \cdot (c_{13} + k_{13}),$$
$$\texttt{C} \cdot (c'_1 + k_1) + \texttt{C} \cdot (c'_5 + k_5) + \texttt{D} \cdot (c'_9 + k_9) + 4 \cdot (c'_{13} + k_{13})$$

The other expressions are computed in a similar way.



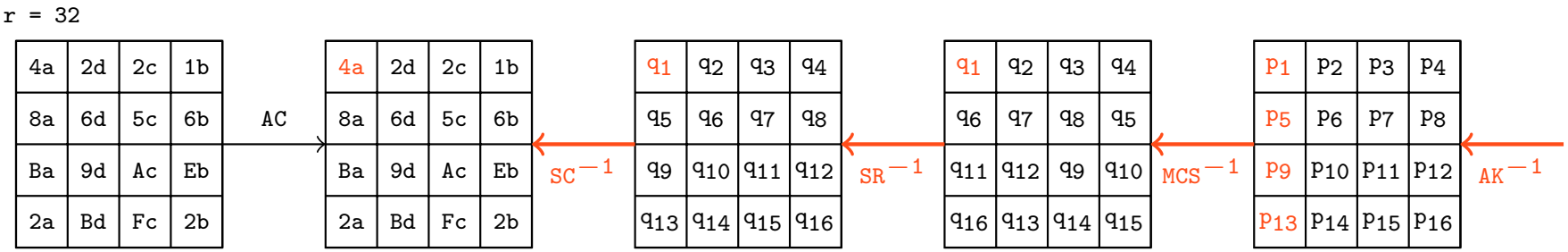
Inversion of the LED Steps

- $\texttt{SR}^{-1}$: Only shifts the entries of the state matrix, no effects on the computed expressions.
- $\texttt{SC}^{-1}$: Results in

$$S^{-1}(\texttt{C} \cdot (c_1 + k_1) + \texttt{C} \cdot (c_5 + k_5) + \texttt{D} \cdot (c_9 + k_9) + 4 \cdot (c_{13} + k_{13})),$$
$$S^{-1}(\texttt{C} \cdot (c'_1 + k_1) + \texttt{C} \cdot (c'_5 + k_5) + \texttt{D} \cdot (c'_9 + k_9) + 4 \cdot (c'_{13} + k_{13}))$$

where S^{-1} is the inverse of the LED SBox. The remaining expressions are computed in the same way again.

A Fault Attack on LED-64

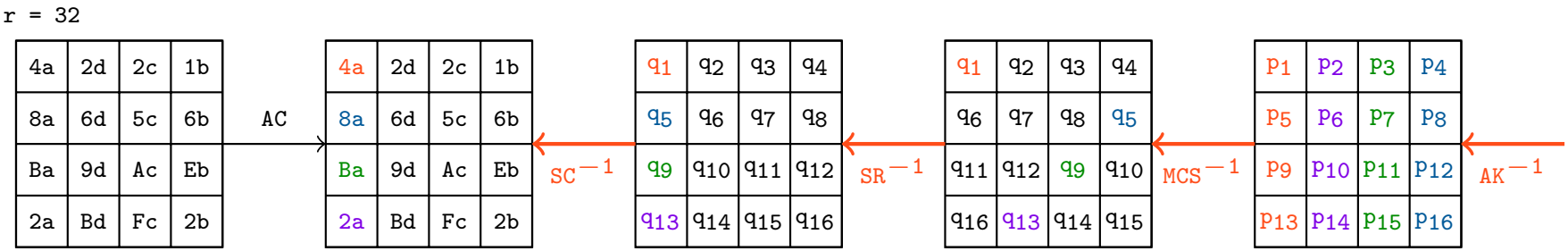


Generation of the fault equations

Compute the XOR difference of two related expressions and obtain a fault equation:

$$4 \cdot a = S^{-1}(\mathtt{C} \cdot (c_1 + k_1) + \mathtt{C} \cdot (c_5 + k_5) + \mathtt{D} \cdot (c_9 + k_9) + 4 \cdot (c_{13} + k_{13})) +$$
$$S^{-1}(\mathtt{C} \cdot (c'_1 + k_1) + \mathtt{C} \cdot (c'_5 + k_5) + \mathtt{D} \cdot (c'_9 + k_9) + 4 \cdot (c'_{13} + k_{13})).$$

A Fault Attack on LED-64



Fault equations for the first column

$$
\begin{aligned}
4 \cdot a = {} & S^{-1}(\mathtt{C} \cdot (c_1 + k_1) + \mathtt{C} \cdot (c_5 + k_5) + \mathtt{D} \cdot (c_9 + k_9) + 4 \cdot (c_{13} + k_{13})) + \\
& S^{-1}(\mathtt{C} \cdot (c'_1 + k_1) + \mathtt{C} \cdot (c'_5 + k_5) + \mathtt{D} \cdot (c'_9 + k_9) + 4 \cdot (c'_{13} + k_{13})) \\
8 \cdot a = {} & S^{-1}(3 \cdot (c_4 + k_4) + 8 \cdot (c_8 + k_8) + 4 \cdot (c_{12} + k_{12}) + 5 \cdot (c_{16} + k_{16})) + \\
& S^{-1}(3 \cdot (c'_4 + k_4) + 8 \cdot (c'_8 + k_8) + 4 \cdot (c'_{12} + k_{12}) + 5 \cdot (c'_{16} + k_{16})) \\
\mathtt{B} \cdot a = {} & S^{-1}(7 \cdot (c_3 + k_3) + 6 \cdot (c_7 + k_7) + 2 \cdot (c_{11} + k_{11}) + \mathtt{E} \cdot (c_{15} + k_{15})) + \\
& S^{-1}(7 \cdot (c'_3 + k_3) + 6 \cdot (c'_7 + k_7) + 2 \cdot (c'_{11} + k_{11}) + \mathtt{E} \cdot (c'_{15} + k_{15})) \\
2 \cdot a = {} & S^{-1}(\mathtt{D} \cdot (c_2 + k_2) + 9 \cdot (c_6 + k_6) + 9 \cdot (c_{10} + k_{10}) + \mathtt{D} \cdot (c_{14} + k_{14})) + \\
& S^{-1}(\mathtt{D} \cdot (c'_2 + k_2) + 9 \cdot (c'_6 + k_6) + 9 \cdot (c'_{10} + k_{10}) + \mathtt{D} \cdot (c'_{14} + k_{14}))
\end{aligned}
$$

Obtain 16 equations $E_{x,i}$ where $x \in \{a, b, c, d\}$, $i \in \{1, 2, 3, 4\}$.

Key tuple filtering (1)

- Evaluate all equations $E_{x,i}$ for all $k_l \in \mathbb{F}_{16}$ that appear in $E_{x,i}$.
  Note: Every $E_{x,i}$ depends exactly on four key indeterminates.
  (# Evaluations: $16^5 = 1048576$)
- Every evaluation produces an element $s \in \mathbb{F}_{16}$. Save the key tuple which produced s to the j-th entry of the list $S_{x,i}$ where $j = \phi(s)_{10}$.
- Determine for every $x \in \{a, b, c, d\}$ the set of possible values j_x of x such that:

$$S_{x,1}(j_x),\ S_{x,2}(j_x),\ S_{x,3}(j_x),\ S_{x,4}(j_x) \neq \emptyset$$

- From this obtain fault tuples $t = (j_a, j_b, j_c, j_d)$.

Key tuple filtering (1)

- Evaluate all equations $E_{x,i}$ for all $k_l \in \mathbb{F}_{16}$ that appear in $E_{x,i}$.
  Note: Every $E_{x,i}$ depends exactly on four key indeterminates.
  (# Evaluations: $16^5 = 1048576$)
- Every evaluation produces an element $s \in \mathbb{F}_{16}$. Save the key tuple which produced s to the j-th entry of the list $S_{x,i}$ where $j = \phi(s)_{10}$.
- Determine for every $x \in \{a, b, c, d\}$ the set of possible values j_x of x such that:

$$S_{x,1}(j_x),\ S_{x,2}(j_x),\ S_{x,3}(j_x),\ S_{x,4}(j_x) \neq \emptyset$$

- From this obtain fault tuples $t = (j_a, j_b, j_c, j_d)$.

UNIVERSITÄT PASSAU

Key tuple filtering (1)

- Evaluate all equations $E_{x,i}$ for all $k_l \in \mathbb{F}_{16}$ that appear in $E_{x,i}$.
  Note: Every $E_{x,i}$ depends exactly on four key indeterminates.
  (# Evaluations: $16^5 = 1048576$)
- Every evaluation produces an element $s \in \mathbb{F}_{16}$. Save the key tuple which produced s to the j-th entry of the list $S_{x,i}$ where $j = \phi(s)_{10}$.
- Determine for every $x \in \{a, b, c, d\}$ the set of possible values j_x of x such that:

$$S_{x,1}(j_x),\ S_{x,2}(j_x),\ S_{x,3}(j_x),\ S_{x,4}(j_x) \neq \emptyset$$

- From this obtain fault tuples $t = (j_a, j_b, j_c, j_d)$.

UNIVERSITÄT PASSAU

Key tuple filtering (1)

- Evaluate all equations $E_{x,i}$ for all $k_l \in \mathbb{F}_{16}$ that appear in $E_{x,i}$.
  Note: Every $E_{x,i}$ depends exactly on four key indeterminates.
  (# Evaluations: $16^5 = 1048576$)
- Every evaluation produces an element $s \in \mathbb{F}_{16}$. Save the key tuple which produced s to the j-th entry of the list $S_{x,i}$ where $j = \phi(s)_{10}$.
- Determine for every $x \in \{a, b, c, d\}$ the set of possible values j_x of x such that:

$$S_{x,1}(j_x),\ S_{x,2}(j_x),\ S_{x,3}(j_x),\ S_{x,4}(j_x) \neq \emptyset$$

- From this obtain fault tuples $t = (j_a, j_b, j_c, j_d)$.

UNIVERSITÄT PASSAU

Key tuple filtering (2)

- Compute the following intersections:

$$
\begin{aligned}
(k_1, k_5, k_9, k_{13}) &: S_{a,1}(j_a) \cap S_{d,2}(j_d) \cap S_{c,3}(j_c) \cap S_{b,4}(j_b) \\
(k_2, k_6, k_{10}, k_{14}) &: S_{a,4}(j_a) \cap S_{d,1}(j_d) \cap S_{c,2}(j_c) \cap S_{b,3}(j_b) \\
(k_3, k_7, k_{11}, k_{15}) &: S_{a,3}(j_a) \cap S_{d,4}(j_d) \cap S_{c,1}(j_c) \cap S_{b,2}(j_b) \\
(k_4, k_8, k_{12}, k_{16}) &: S_{a,2}(j_a) \cap S_{d,3}(j_d) \cap S_{c,4}(j_c) \cap S_{b,1}(j_b)
\end{aligned}
$$

- Recombine the key values $(k_1, \ldots, k_{16})$ using all possible choises of the four intersections and get key candidate sets.
- Each intersection contains typically $2^4 - 2^8$ elements. Thus the size of a candidate set is usually in the range $2^{19} - 2^{26}$.

Final step: Perform a brute force search.

UNIVERSITÄT PASSAU

Key tuple filtering (2)

- Compute the following intersections:

$$
\begin{aligned}
(k_1, k_5, k_9, k_{13}) &: S_{a,1}(j_a) \cap S_{d,2}(j_d) \cap S_{c,3}(j_c) \cap S_{b,4}(j_b) \\
(k_2, k_6, k_{10}, k_{14}) &: S_{a,4}(j_a) \cap S_{d,1}(j_d) \cap S_{c,2}(j_c) \cap S_{b,3}(j_b) \\
(k_3, k_7, k_{11}, k_{15}) &: S_{a,3}(j_a) \cap S_{d,4}(j_d) \cap S_{c,1}(j_c) \cap S_{b,2}(j_b) \\
(k_4, k_8, k_{12}, k_{16}) &: S_{a,2}(j_a) \cap S_{d,3}(j_d) \cap S_{c,4}(j_c) \cap S_{b,1}(j_b)
\end{aligned}
$$

- Recombine the key values $(k_1, \ldots, k_{16})$ using all possible choises of the four intersections and get key candidate sets.
- Each intersection contains typically $2^4 - 2^8$ elements. Thus the size of a candidate set is usually in the range $2^{19} - 2^{26}$.

Final step: Perform a brute force search.

Key tuple filtering (2)

- Compute the following intersections:

$$
\begin{aligned}
(k_1, k_5, k_9, k_{13}) &: S_{a,1}(j_a) \cap S_{d,2}(j_d) \cap S_{c,3}(j_c) \cap S_{b,4}(j_b) \\
(k_2, k_6, k_{10}, k_{14}) &: S_{a,4}(j_a) \cap S_{d,1}(j_d) \cap S_{c,2}(j_c) \cap S_{b,3}(j_b) \\
(k_3, k_7, k_{11}, k_{15}) &: S_{a,3}(j_a) \cap S_{d,4}(j_d) \cap S_{c,1}(j_c) \cap S_{b,2}(j_b) \\
(k_4, k_8, k_{12}, k_{16}) &: S_{a,2}(j_a) \cap S_{d,3}(j_d) \cap S_{c,4}(j_c) \cap S_{b,1}(j_b)
\end{aligned}
$$

- Recombine the key values $(k_1, \ldots, k_{16})$ using all possible choises of the four intersections and get key candidate sets.
- Each intersection contains typically $2^4 - 2^8$ elements. Thus the size of a candidate set is usually in the range $2^{19} - 2^{26}$.

Final step: Perform a brute force search.

UNIVERSITÄT PASSAU

Key tuple filtering (2)

- Compute the following intersections:

$$
\begin{aligned}
(k_1, k_5, k_9, k_{13}) &: S_{a,1}(j_a) \cap S_{d,2}(j_d) \cap S_{c,3}(j_c) \cap S_{b,4}(j_b) \\
(k_2, k_6, k_{10}, k_{14}) &: S_{a,4}(j_a) \cap S_{d,1}(j_d) \cap S_{c,2}(j_c) \cap S_{b,3}(j_b) \\
(k_3, k_7, k_{11}, k_{15}) &: S_{a,3}(j_a) \cap S_{d,4}(j_d) \cap S_{c,1}(j_c) \cap S_{b,2}(j_b) \\
(k_4, k_8, k_{12}, k_{16}) &: S_{a,2}(j_a) \cap S_{d,3}(j_d) \cap S_{c,4}(j_c) \cap S_{b,1}(j_b)
\end{aligned}
$$

- Recombine the key values $(k_1, \ldots, k_{16})$ using all possible choises of the four intersections and get key candidate sets.
- Each intersection contains typically $2^4 - 2^8$ elements. Thus the size of a candidate set is usually in the range $2^{19} - 2^{26}$.

Final step: Perform a brute force search.

An example

Assume we have the following setup:

$$k = \texttt{01234567 89ABCDEF}$$
$$m = \texttt{01234567 89ABCDEF}$$
$$c = \texttt{FDD6FB98 45F81456}$$
$$c' = \texttt{51B8AB31 169AC161}$$

The faulty ciphertext c' was obtained by injecting the error value $e = 8$ into the first entry of the state matrix at the beginning of the 30-th round.

A Fault Attack on LED-64

a	1	2	3	4	5	6	7	8	**9**	A	B	C	D	E	F
$\#S_{a,1}$	2^{14}	0	2^{14}	0	0	0	0	0	$\mathbf{2^{14}}$	0	2^{14}	0	0	0	0
$\#S_{a,2}$	0	0	0	0	0	0	0	2^{14}	$\mathbf{2^{14}}$	0	0	2^{14}	2^{14}	0	0
$\#S_{a,3}$	0	0	0	2^{14}	0	0	2^{14}	0	$\mathbf{2^{14}}$	2^{14}	0	0	0	0	0
$\#S_{a,4}$	0	2^{13}	0	2^{13}	0	2^{13}	2^{13}	2^{13}	$\mathbf{2^{14}}$	0	0	0	0	0	2^{13}

b	1	2	3	4	**5**	6	7	8	9	A	B	C	D	E	F
$\#S_{b,1}$	0	0	2^{13}	0	$\mathbf{2^{13}}$	0	0	0	2^{14}	0	2^{13}	0	2^{13}	0	2^{14}
$\#S_{b,2}$	0	0	0	2^{13}	$\mathbf{2^{13}}$	2^{14}	0	2^{13}	2^{13}	0	2^{14}	0	0	0	0
$\#S_{b,3}$	2^{13}	0	2^{14}	2^{13}	$\mathbf{2^{13}}$	0	2^{13}	0	0	0	2^{13}	2^{13}	0	0	0
$\#S_{b,4}$	2^{14}	2^{14}	0	0	$\mathbf{2^{14}}$	2^{14}	0	0	0	0	0	0	0	0	0

c	1	2	3	4	5	6	7	**8**	9	A	**B**	C	D	E	F
$\#S_{c,1}$	0	0	2^{14}	2^{14}	0	0	0	$\mathbf{2^{13}}$	0	0	$\mathbf{2^{13}}$	2^{13}	0	0	2^{13}
$\#S_{c,2}$	2^{13}	0	0	0	0	0	2^{13}	$\mathbf{2^{13}}$	2^{13}	2^{13}	$\mathbf{2^{14}}$	0	2^{13}	0	0
$\#S_{c,3}$	2^{13}	0	0	0	2^{13}	2^{14}	0	$\mathbf{2^{14}}$	0	0	$\mathbf{2^{13}}$	0	0	0	2^{13}
$\#S_{c,4}$	0	2^{13}	2^{13}	0	0	0	0	$\mathbf{2^{14}}$	2^{13}	0	$\mathbf{2^{13}}$	2^{13}	0	0	2^{13}

d	1	**2**	3	4	5	6	7	8	9	A	B	C	D	E	F
$\#S_{d,1}$	2^{13}	$\mathbf{2^{13}}$	2^{13}	0	0	0	2^{13}	0	0	2^{13}	2^{14}	0	2^{13}	0	0
$\#S_{d,2}$	2^{13}	$\mathbf{2^{13}}$	2^{13}	2^{14}	0	2^{13}	0	0	0	2^{13}	0	2^{13}	0	0	0
$\#S_{d,3}$	0	$\mathbf{2^{14}}$	2^{13}	0	0	2^{13}	0	2^{13}	2^{13}	0	2^{13}	0	0	0	2^{13}
$\#S_{d,4}$	2^{13}	$\mathbf{2^{13}}$	0	2^{13}	0	0	2^{13}	0	2^{13}	2^{13}	0	2^{13}	0	0	2^{13}

Results

The distribution tables give us the two fault tuples (9,5,8,2) and (9,5,B,2). This results in:

-	(9,5,8,2)	(9,5,B,2)
(k_1, k_5, k_9, k_{13})	2^7	2^6
$(k_2, k_6, k_{10}, k_{14})$	2^5	2^6
$(k_3, k_7, k_{11}, k_{15})$	2^5	2^5
$(k_4, k_8, k_{12}, k_{16})$	2^7	2^6
$\#keys$	2^{24}	2^{23}

Observation

The key candidate sets are by construction pairwise disjoint and only one of them contains the secret key.

UNIVERSITÄT PASSAU

Results

The distribution tables give us the two fault tuples (9,5,8,2) and (9,5,B,2). This results in:

-	(9,5,8,2)	(9,5,B,2)
(k_1, k_5, k_9, k_{13})	2^7	2^6
$(k_2, k_6, k_{10}, k_{14})$	2^5	2^6
$(k_3, k_7, k_{11}, k_{15})$	2^5	2^5
$(k_4, k_8, k_{12}, k_{16})$	2^7	2^6
$\#keys$	2^{24}	2^{23}

Observation

The key candidate sets are by construction pairwise disjoint and only one of them contains the secret key.

A Fault Attack on LED-64

Question: Can we discard candidate sets not containing the secret key?

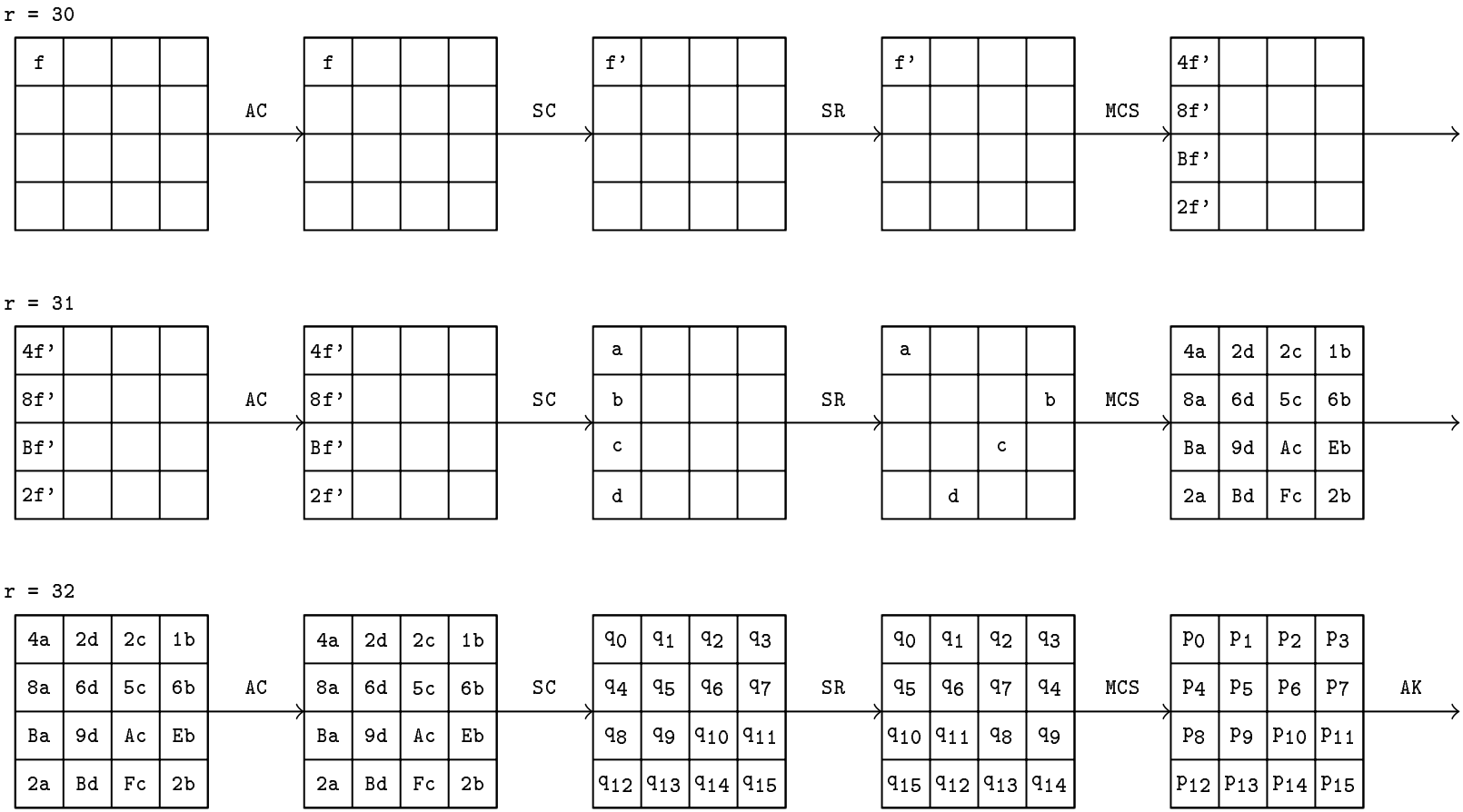


Figure: Fault propagation in the LED cipher.

A Fault Attack on LED-64

Answer: Yes (in many cases) by using key set filtering.

Key set filtering

- One can construct the following four equations:

$$
\begin{aligned}
4f' &= S^{-1}(y_0) + S^{-1}(y_0 + a) \\
8f' &= S^{-1}(y_4) + S^{-1}(y_4 + b) \\
Bf' &= S^{-1}(y_8) + S^{-1}(y_8 + c) \\
2f' &= S^{-1}(y_{12}) + S^{-1}(y_{12} + d)
\end{aligned}
$$

  where $y_i \in \mathbb{F}_{16}$ are (guessed) state elements from round 31 after the SC operation was applied.
- Apply the same mechanisms as with key tuple filtering and discard all fault tuples (and thus key sets) that do not fulfill the above equations.

Answer: Yes (in many cases) by using key set filtering.

Key set filtering

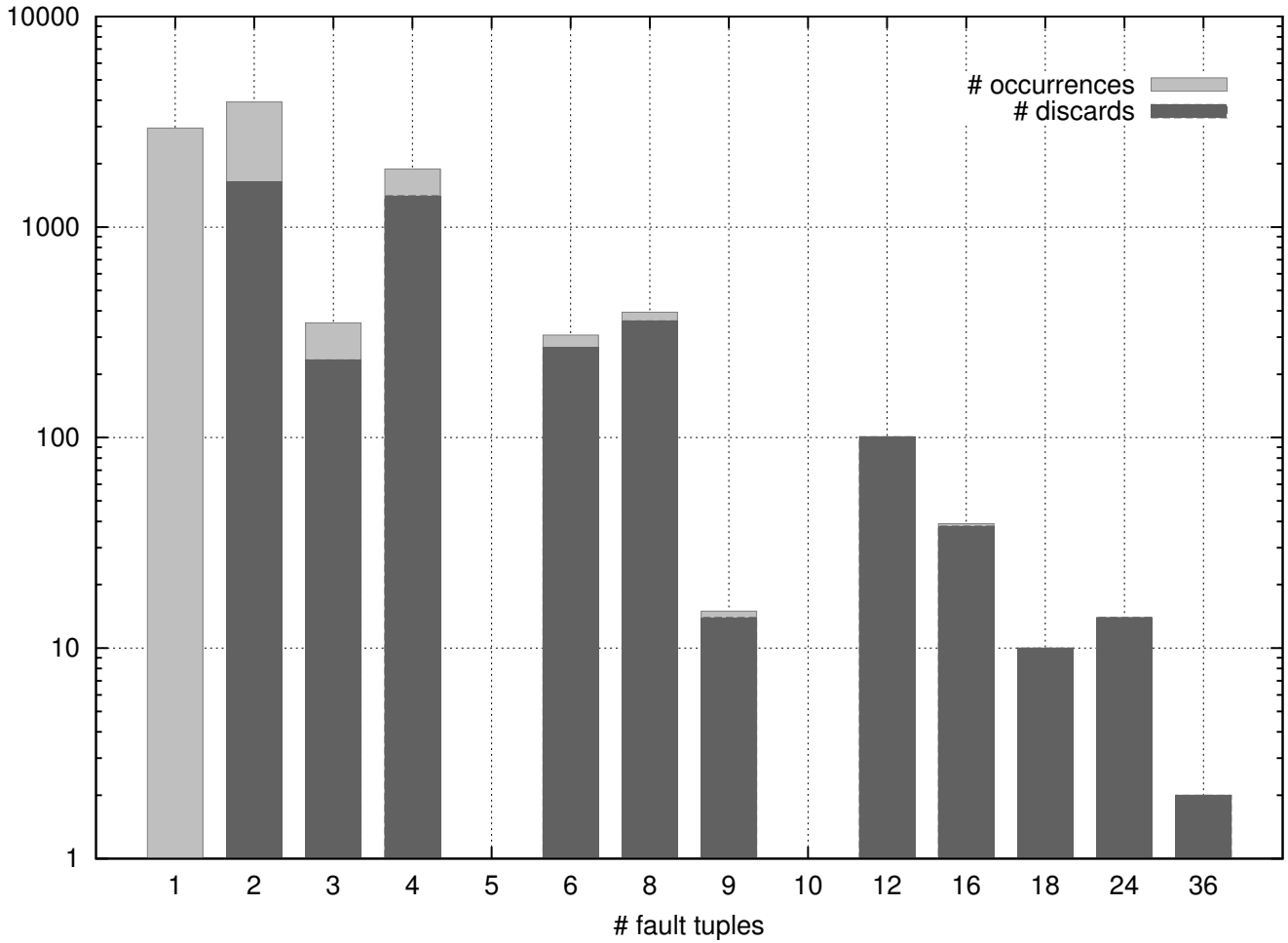
- One can construct the following four equations:

$$
\begin{aligned}
4f' &= S^{-1}(y_0) + S^{-1}(y_0 + a) \\
8f' &= S^{-1}(y_4) + S^{-1}(y_4 + b) \\
Bf' &= S^{-1}(y_8) + S^{-1}(y_8 + c) \\
2f' &= S^{-1}(y_{12}) + S^{-1}(y_{12} + d)
\end{aligned}
$$

  where $y_i \in \mathbb{F}_{16}$ are (guessed) state elements from round 31 after the SC operation was applied.
- Apply the same mechanisms as with key tuple filtering and discard all fault tuples (and thus key sets) that do not fulfill the above equations.

UNIVERSITÄT PASSAU

Experimental results



UNIVERSITÄT PASSAU

Experimental results

#ft	1	2	3	4	5	6	8
occurred	2952	3926	351	1887	1	307	394
discarded	-	1640	234	1410	1	268	359
#ft	9	10	12	16	18	24	36
occurred	15	1	101	39	10	14	2
discarded	14	1	101	38	10	14	2

Table: Efficiency of key set filtering.

#ft	1	2	3	4	5	6	8
ødiscarded	-	0.4	0.9	1.4	2.0	2.5	3.6
#ft	9	10	12	16	18	24	36
ødiscarded	3.7	5.0	6.1	8.4	8.4	12.6	24.0

Table: Average number of discards.

Outline

A Two-Stage Fault Attack on LED-128

Question: Can we apply the previously described attack on LED-128, too?

Reminder: General layout of LED-64 and LED-128.

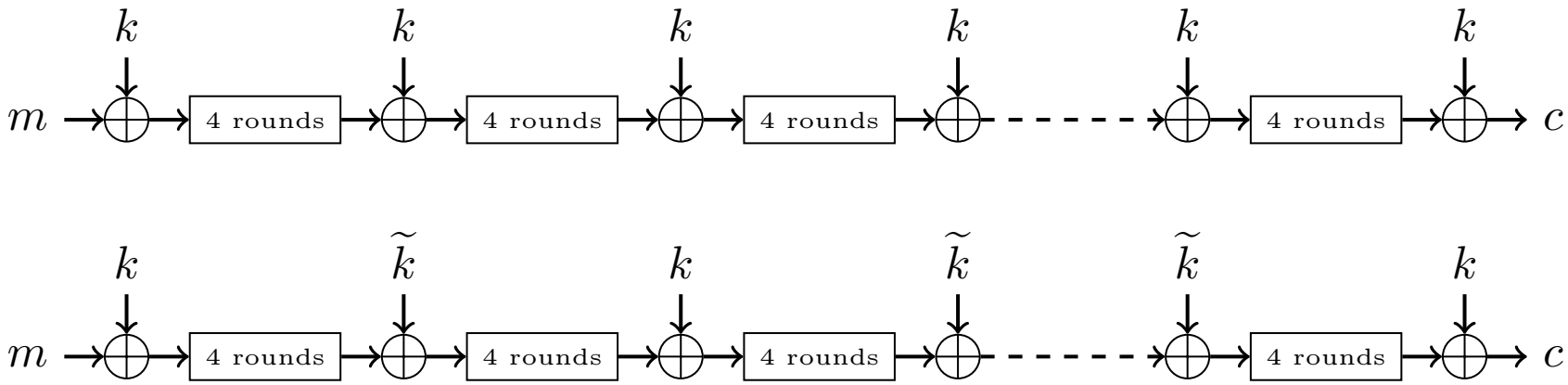


Figure: LED key usage: 64-bit key (top) and 128-bit key (bottom).

Problem with LED-128

The 64-bit keys k and $\widetilde{k}$ are not connected through some kind of key schedule and therefore independent from each other.

Reminder: General layout of LED-64 and LED-128.

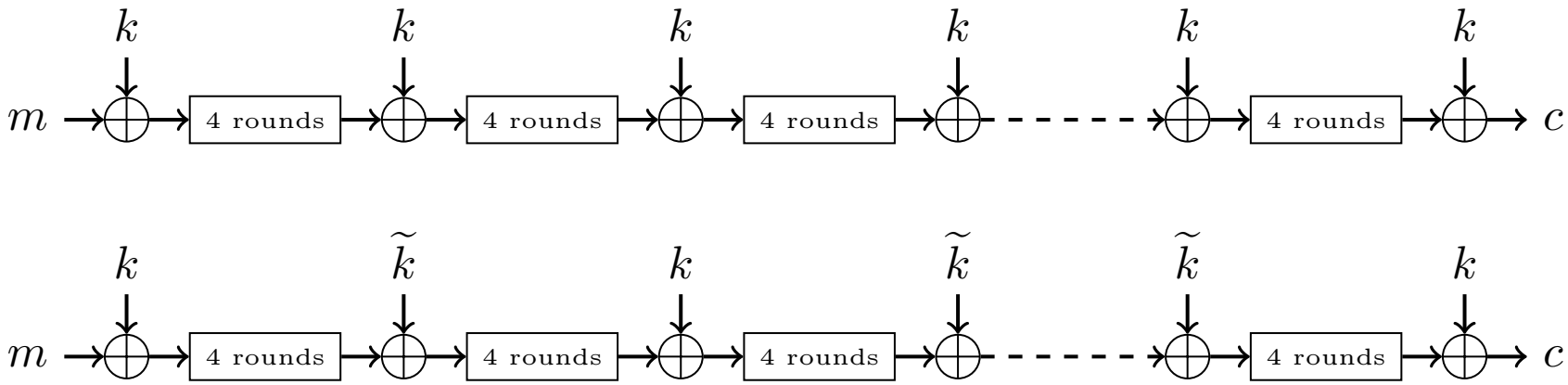


Figure: LED key usage: 64-bit key (top) and 128-bit key (bottom).

Problem with LED-128

The 64-bit keys k and $\widetilde{k}$ are not connected through some kind of key schedule and therefore independent from each other.

A Two-Stage Fault Attack on LED-128

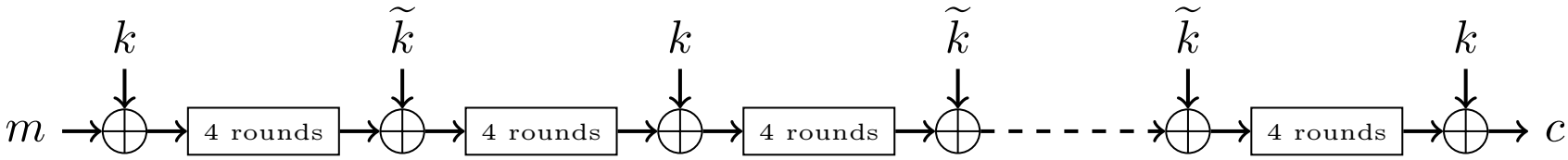


Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ $(n \leq 256)$ for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

A Two-Stage Fault Attack on LED-128

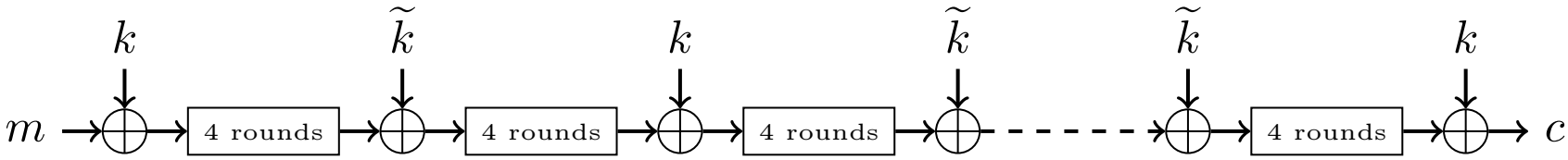


Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ ($n \leq 256$) for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

A Two-Stage Fault Attack on LED-128

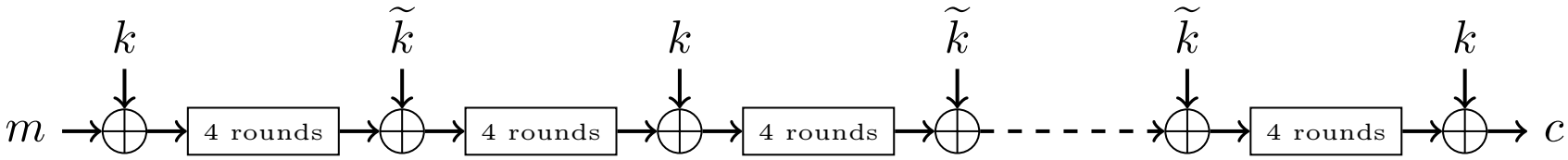


Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ ($n \leq 256$) for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

A Two-Stage Fault Attack on LED-128

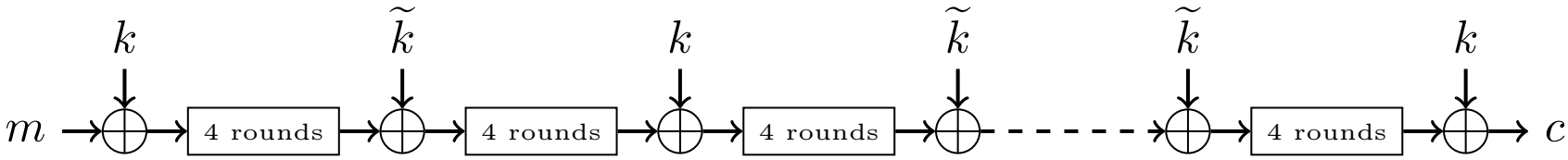


Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ ($n \leq 256$) for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

A Two-Stage Fault Attack on LED-128

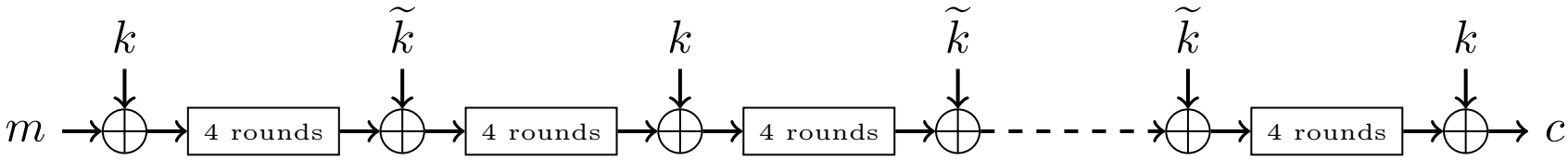


Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ ($n \leq 256$) for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

A Two-Stage Fault Attack on LED-128

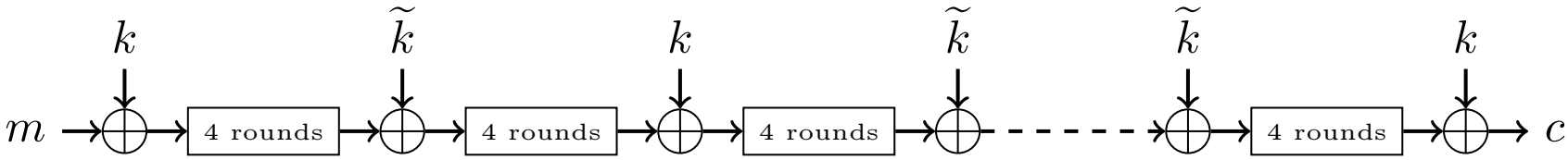
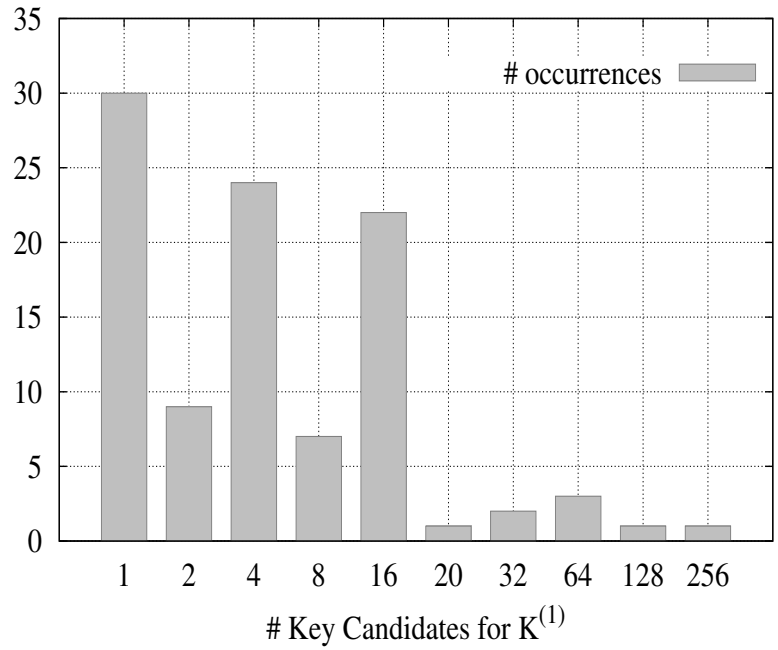


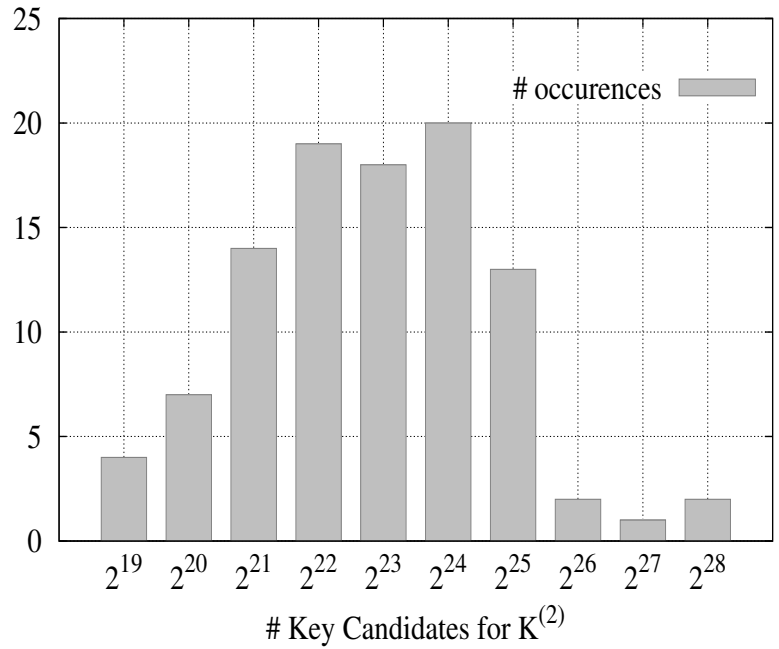
Figure: LED-128.

Outline of the attack

1. Inject a fault in 46-th round and obtain c'. Repeat it and obtain $c'' \neq c'$.
2. Inject a fault in 42-th round and obtain $\widetilde{c}$.
3. Filter key candidates for k using (c, c') and (c, c'') and obtain a small number of candidates $k_1, \ldots, k_n$ ($n \leq 256$) for k.
4. Use k_i to "decrypt" the last four rounds of c and $\widetilde{c}$, resulting in correct and faulty candidate states s_i and $\widetilde{s}_i$.
5. Filter key candidates for $\widetilde{k}$ using $(s_i, \widetilde{s}_i)$. Note: Many of the pairs give empty key candidate sets!
6. Perform a brute-force search on all the combinations of $(k_i, \widetilde{k}_j)$.

Experimental results





- The diagrams above shows the key space sizes for 100 runs of the attack.
- The average running time for the filtering step was 517.41 seconds and 1012.51 seconds for the brute-force search.

Outline

Conclusion

- Implementation attacks are dangerous and can compromise the security of chips.
- Attack technologies are constantly improving and so should defensive measures.
- Designers (of hardware / software) must be aware of the latest attacks to be able to develop suitable countermeasures.
- There is no such thing as absolute protection: Given enough time and resources any protection can be broken.
- “Security is a process, not a product” - Bruce Schneier.

Conclusion

- Implementation attacks are dangerous and can compromise the security of chips.
- Attack technologies are constantly improving and so should defensive measures.
- Designers (of hardware / software) must be aware of the latest attacks to be able to develop suitable countermeasures.
- There is no such thing as absolute protection: Given enough time and resources any protection can be broken.
- “Security is a process, not a product” - Bruce Schneier.

Conclusion

- Implementation attacks are dangerous and can compromise the security of chips.
- Attack technologies are constantly improving and so should defensive measures.
- Designers (of hardware / software) must be aware of the latest attacks to be able to develop suitable countermeasures.
- There is no such thing as absolute protection: Given enough time and resources any protection can be broken.
- “Security is a process, not a product” - Bruce Schneier.

Conclusion

- Implementation attacks are dangerous and can compromise the security of chips.
- Attack technologies are constantly improving and so should defensive measures.
- Designers (of hardware / software) must be aware of the latest attacks to be able to develop suitable countermeasures.
- There is no such thing as absolute protection: Given enough time and resources any protection can be broken.
- “Security is a process, not a product” - Bruce Schneier.

Conclusion

- Implementation attacks are dangerous and can compromise the security of chips.
- Attack technologies are constantly improving and so should defensive measures.
- Designers (of hardware / software) must be aware of the latest attacks to be able to develop suitable countermeasures.
- There is no such thing as absolute protection: Given enough time and resources any protection can be broken.
- “Security is a process, not a product” - Bruce Schneier.

Outlook

What's next?

- Fault attacks on other block ciphers, stream ciphers or hash functions.
- Algebraic Fault Attacks.
- FIRE: Fault Injection for Reverse Engineering.
- ...

Commercials

Interested?

- Looking for a subject for a bachelor / master thesis?
- Contact me: jovanovi@fim.uni-passau.de

Fin

Thank you for your attention!