Analysis and Design of Symmetric Cryptographic Algorithms

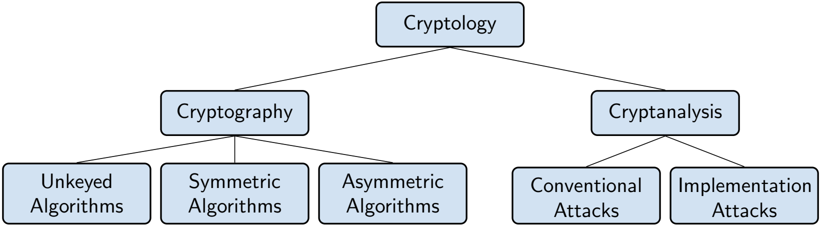
Rigorosum

Philipp Jovanovic

October 30, 2015

Cryptology

- Cryptography: *science of designing secure communication channels in presence of third parties*
- Cryptanalysis: *science of evaluating the security of cryptographic constructions*



Cryptography Everywhere

UNIVERSITÄT PASSAU

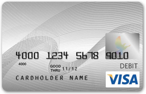
zalando

amazon

NETFLIX

Dropbox

PayPal

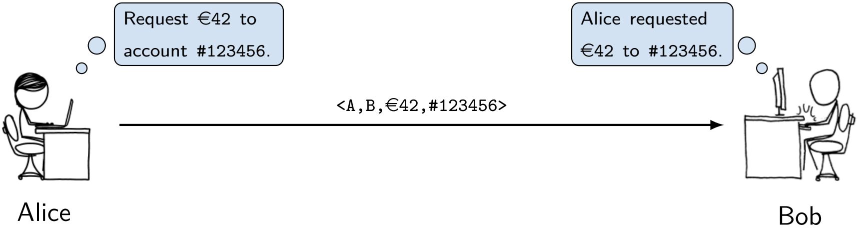


https://

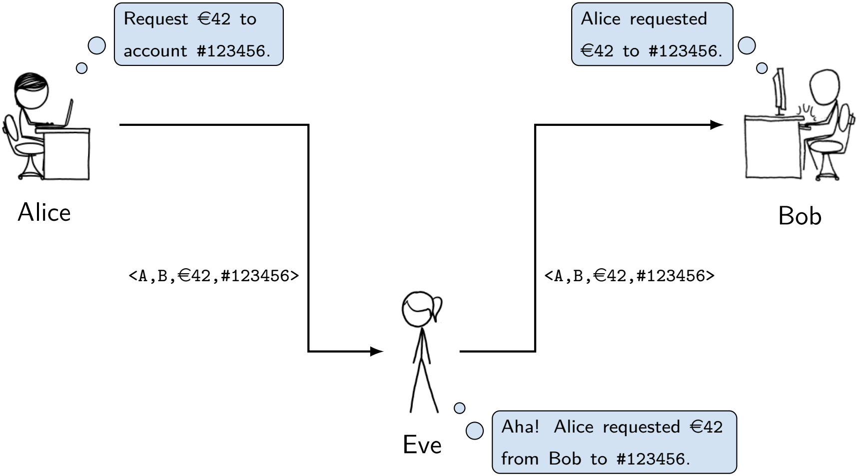
bitcoin

HARDENED
PGP

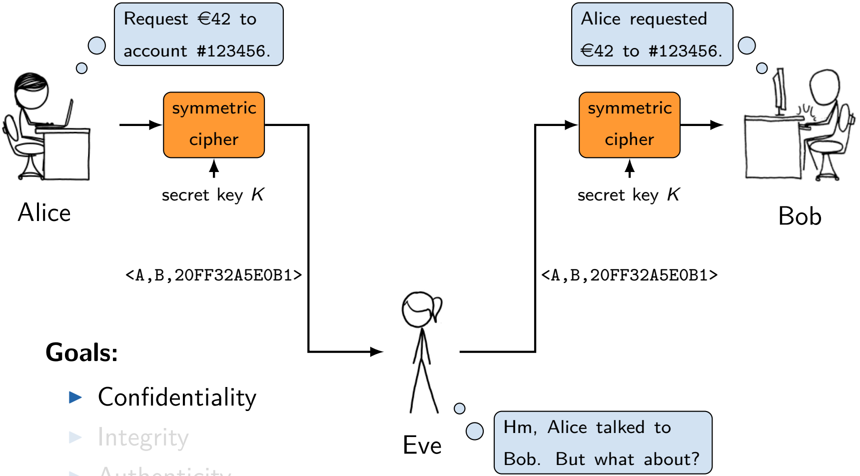
Goals of Cryptography



Goals of Cryptography



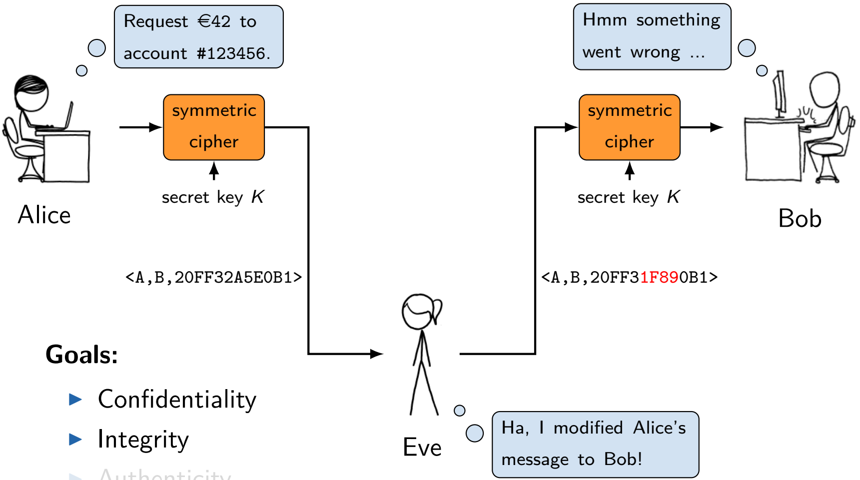
Goals of Cryptography



Goals:

- Confidentiality
- Integrity
- Authenticity

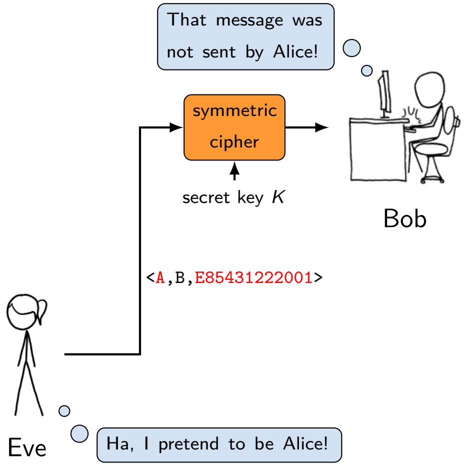
Goals of Cryptography



Goals:

- Confidentiality
- Integrity
- Authenticity

Goals of Cryptography



Goals:

- Confidentiality
- Integrity
- Authenticity

Thesis Overview

Part I: Cryptanalysis

- **Multi-Stage Fault Attacks on**
  - LED
  - PRINCE
  - **Bel-T**
- Algebraic Fault Attacks on LED64

Part II: Cryptography

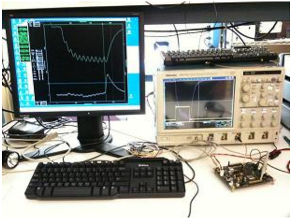
- **NORX: Parallel and Scalable Authenticated Encryption**
- Security Evaluation of NORX
  - General, algebraic, differential, rotational properties
  - NODE: (NO)RX (D)ifferential Search (E)ngine

Part I: Cryptanalysis

Multi-Stage Fault Attacks on Bel-T

Implementation Attacks

- **Goal:** recover secrets by exploiting the implementation of a cipher
- **Active:** fault-based attacks
- **Passive:** power-, timing-, electromagnetic attacks

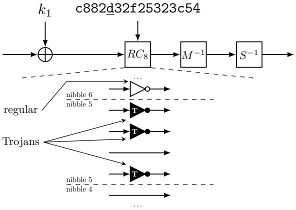


Fault Attacks

- **Approach:** fault injection through a physical disturbance
- **Realisation:** supply voltage manipulation, laser, hw trojans(!), ...
- **Analysis:** recover secret information from correct and faulty output

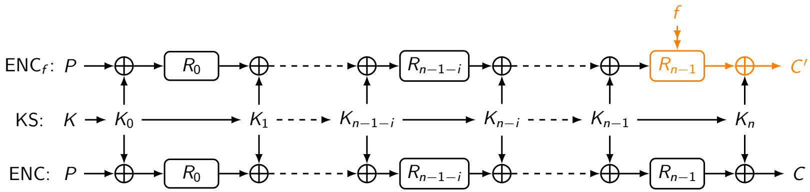
Crypto Chip

Fault-Attack Setup



Hardware Trojans

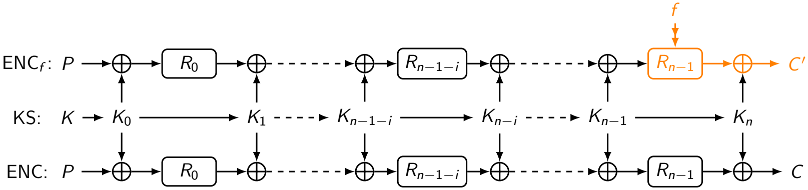
Fault Attacks (simplified)



Inject fault f in the last round R_{n-1}

- $\text{Enc}(P, K) = C$
- $\text{Enc}_f(P, K) = C'$
- Analyse (C, C') to recover K_n

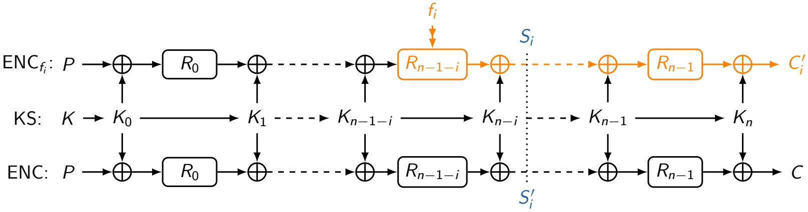
Fault Attacks (simplified)



Next steps

- KS bijective:
  - invert KS and obtain K
  - example: AES
- Otherwise: Multi-Stage Fault Attacks

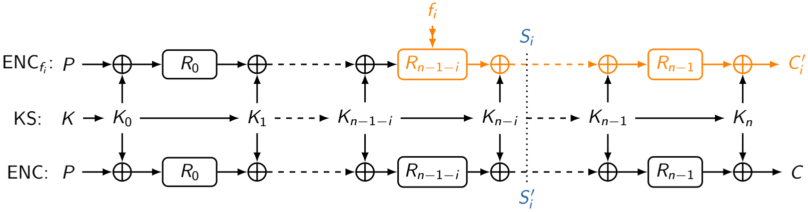
Multi-Stage Fault Attacks (simplified)



Stage i: inject fault f_i in round R_{n-1-i}

- $\mathrm{Enc}(P, K) = C$
- $\mathrm{Enc}_{f_i}(P, K) = C'_i$
- $\mathrm{Dec}(C, K_n, \ldots, K_{n-i+1}) = S_i$
- $\mathrm{Dec}(C'_i, K_n, \ldots, K_{n-i+1}) = S'_i$
- Analyse (S_i, S'_i) to recover K_{n-i}

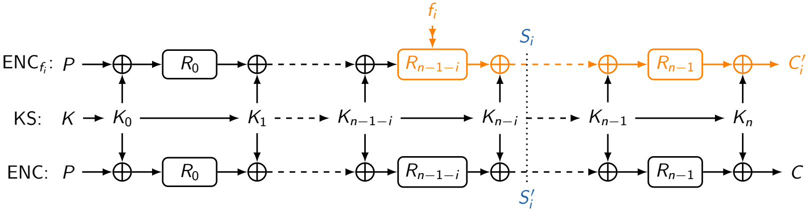
Multi-Stage Fault Attacks (simplified)



Stage i: inject fault f_i in round R_{n-1-i}

- $\mathrm{Enc}(P, K) = C$
- $\mathrm{Enc}_{f_i}(P, K) = C_i'$
- $\mathrm{Dec}(C, K_n, \ldots, K_{n-i+1}) = S_i$
- $\mathrm{Dec}(C_i', K_n, \ldots, K_{n-i+1}) = S_i'$
- Analyse (S_i, S_i') to recover K_{n-i}

Multi-Stage Fault Attacks (simplified)



Stage i: inject fault f_i in round R_{n-1-i}

- $\mathrm{Enc}(P, K) = C$
- $\mathrm{Enc}_{f_i}(P, K) = C'_i$
- $\mathrm{Dec}(C, K_n, \ldots, K_{n-i+1}) = S_i$
- $\mathrm{Dec}(C'_i, K_n, \ldots, K_{n-i+1}) = S'_i$
- Analyse (S_i, S'_i) to recover K_{n-i}

Fault Analysis of Bel-T

Overview

- Block cipher family
- Block size: 128-bit
- Key sizes: 128-bit, 192-bit, 256-bit
- Based on the Lai-Massey scheme
- National standard of the Republic of Belarus since 2011

Bel-T

- Key setup:

  256-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8$

  192-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3, \theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

  128-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5 = \theta_1, \theta_6 = \theta_2, \theta_7 = \theta_3, \theta_8 = \theta_4$

  with 32-bit values θ_i

- Key usage:

	i	K_{7i-6}	K_{7i-5}	K_{7i-4}	K_{7i-3}	K_{7i-2}	K_{7i-1}	K_{7i}	
	1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	
ENC $\downarrow$	$\vdots$	$\vdots$			$\vdots$			$\vdots$	DEC $\uparrow$
	8	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	
	i	K_{7i}	K_{7i-1}	K_{7i-2}	K_{7i-3}	K_{7i-4}	K_{7i-5}	K_{7i-6}	

- Substitution layer:

$$G_r(x) = (H(x_1) \parallel H(x_2) \parallel H(x_3) \parallel H(x_4)) \lll r$$

  with 8-bit S-box H

Bel-T

- **Key setup:**

  256-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8$

  192-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3, \theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

  128-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5 = \theta_1, \theta_6 = \theta_2, \theta_7 = \theta_3, \theta_8 = \theta_4$

  with 32-bit values θ_i

- **Key usage:**

	i	K_{7i-6}	K_{7i-5}	K_{7i-4}	K_{7i-3}	K_{7i-2}	K_{7i-1}	K_{7i}	
ENC $\downarrow$	1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	DEC $\uparrow$
	$\vdots$	$\vdots$			$\vdots$			$\vdots$	
	8	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	
	i	K_{7i}	K_{7i-1}	K_{7i-2}	K_{7i-3}	K_{7i-4}	K_{7i-5}	K_{7i-6}	

- Substitution layer:

$$G_r(x) = (H(x_1) \parallel H(x_2) \parallel H(x_3) \parallel H(x_4)) \lll r$$

with 8-bit S-box H

Bel-T

- Key setup:

  256-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8$

  192-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3, \theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

  128-bit: $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5 = \theta_1, \theta_6 = \theta_2, \theta_7 = \theta_3, \theta_8 = \theta_4$

  with 32-bit values θ_i

- Key usage:

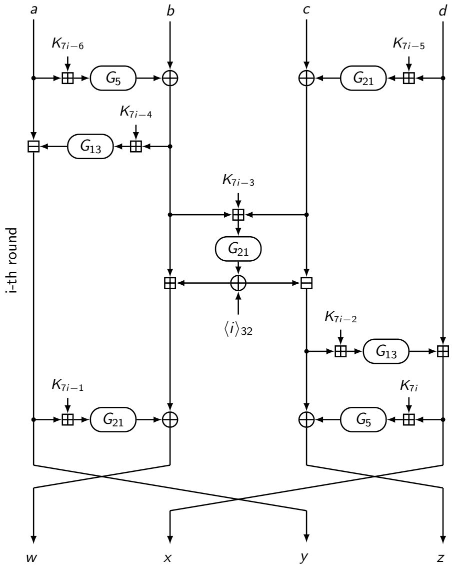
	i	K_{7i-6}	K_{7i-5}	K_{7i-4}	K_{7i-3}	K_{7i-2}	K_{7i-1}	K_{7i}	
ENC $\downarrow$	1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	$\uparrow$ DEC
	$\vdots$	$\vdots$			$\vdots$			$\vdots$	
	8	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	
	i	K_{7i}	K_{7i-1}	K_{7i-2}	K_{7i-3}	K_{7i-4}	K_{7i-5}	K_{7i-6}	

- Substitution layer:

$$G_r(x) = (H(x_1) \parallel H(x_2) \parallel H(x_3) \parallel H(x_4)) \lll r$$

  with 8-bit S-box H

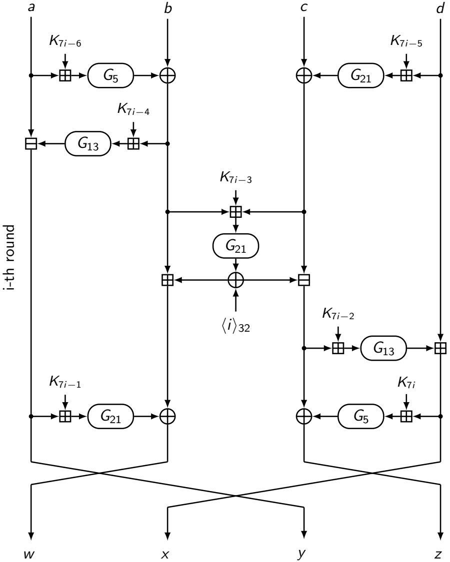
Fault Analysis of Bel-T



Setup:

- **Random-Fault Model (RFM):**
  - Chosen location
  - Random value
- **Chosen-Fault Model (CFM):**
  - Chosen location
  - Chosen value (usually zero)
- **Round 8** of encryption or decryption
- **Fault locations:** $L_1, \ldots, L_5$

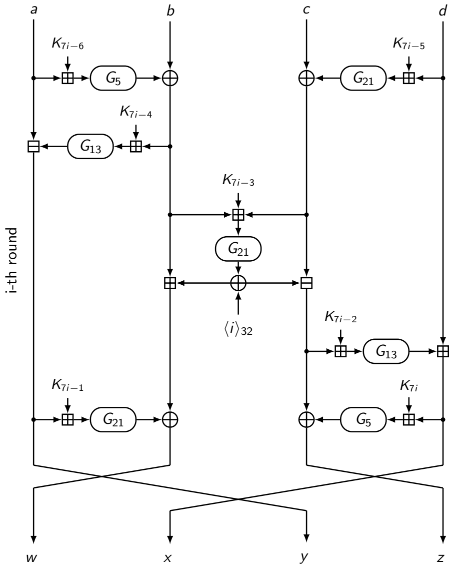
Fault Analysis of Bel-T



Setup:

- **Random-Fault Model (RFM):**
  - Chosen location
  - Random value
- **Chosen-Fault Model (CFM):**
  - Chosen location
  - Chosen value (usually zero)
- **Round 8** of encryption or decryption
- **Fault locations:** $L_1, \ldots, L_5$

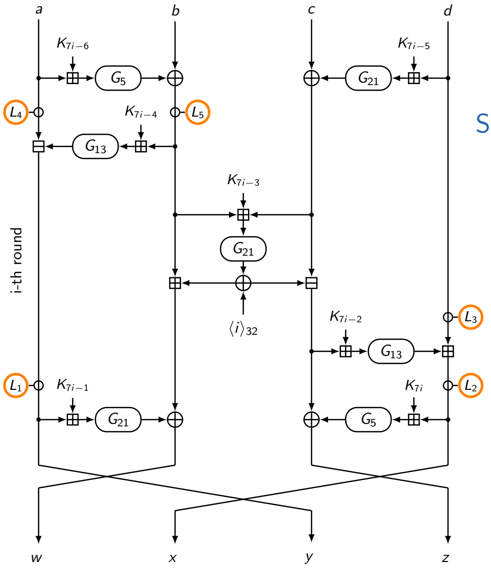
Fault Analysis of Bel-T



Setup:

- **Random-Fault Model (RFM):**
  - Chosen location
  - Random value
- **Chosen-Fault Model (CFM):**
  - Chosen location
  - Chosen value (usually zero)
- **Round 8** of encryption or decryption
- **Fault locations:** $L_1, \ldots, L_5$

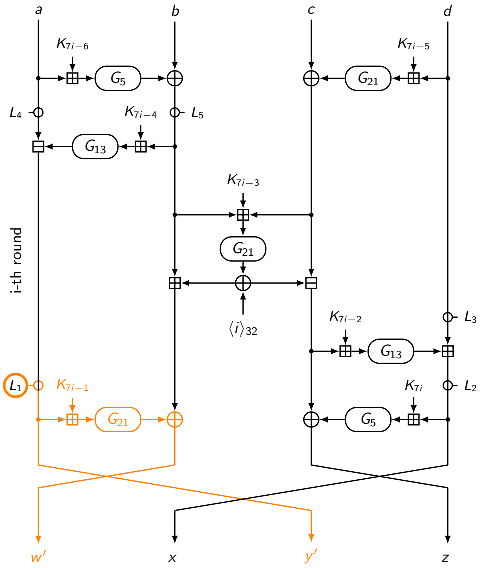
Fault Analysis of Bel-T



Setup:

- **Random-Fault Model (RFM):**
  - Chosen location
  - Random value
- **Chosen-Fault Model (CFM):**
  - Chosen location
  - Chosen value (usually zero)
- **Round 8** of encryption or decryption
- **Fault locations:** $L_1, \ldots, L_5$

Fault Analysis of Bel-T-128



- **Recovered key parts:**

  θ_1 θ_2 θ_3 θ_4

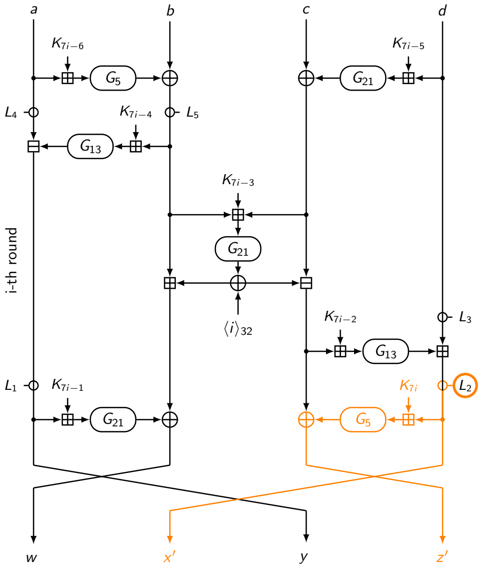
  θ_5 θ_6 θ_7 θ_8

- **Stage 1 (enc, $i = 8$):**
  - Target K_{7i-1} $(= \theta_7 = \theta_3)$
  - RFM-fault at L_1
  - Solve:

$$w \oplus w' = G_{21}(e) \oplus G_{21}(e')$$
$$e = y \boxplus \theta_7$$
$$e' = y' \boxplus \theta_7$$

- **Stage 2 (enc, $i = 8$):**
  - Target K_{7i} $(= \theta_8 = \theta_4)$
  - RFM-fault at L_2

Fault Analysis of Bel-T-128



- **Recovered key parts:**

  θ_1 θ_2 θ_3 θ_4

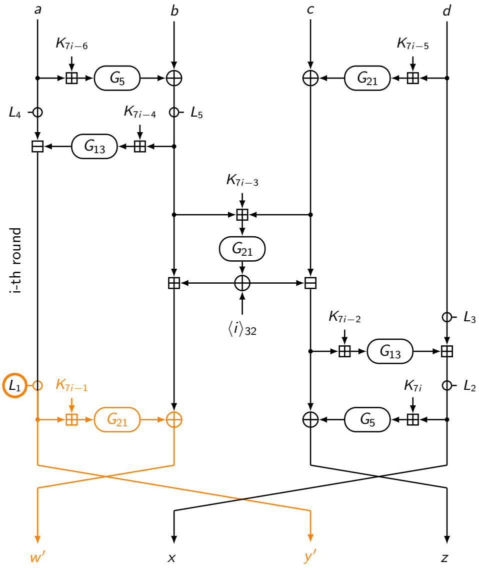
  θ_5 θ_6 θ_7 θ_8

- **Stage 1 (enc, $i = 8$):**
  - Target K_{7i-1} $(= \theta_7 = \theta_3)$
  - RFM-fault at L_1
  - Solve:

$$w \oplus w' = G_{21}(e) \oplus G_{21}(e')$$
$$e = y \boxplus \theta_7$$
$$e' = y' \boxplus \theta_7$$

- **Stage 2 (enc, $i = 8$):**
  - Target K_{7i} $(= \theta_8 = \theta_4)$
  - RFM-fault at L_2

Fault Analysis of Bel-T-128



- **Recovered key parts:**

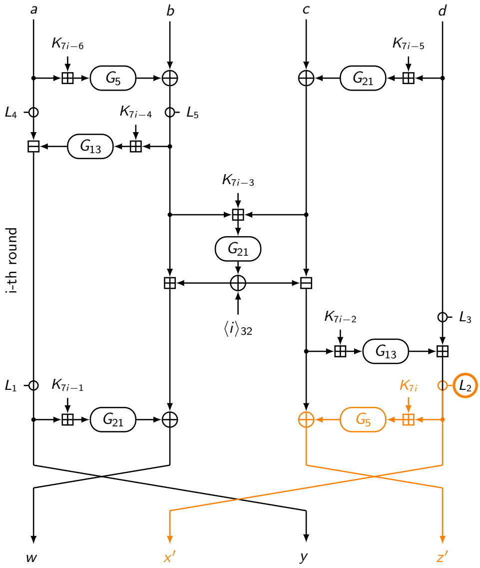
  θ_1 θ_2 θ_3 θ_4

  θ_5 θ_6 θ_7 θ_8

- **Stage 3 (dec, $i = 1$):**
  - Target K_{7i-1} ($= \theta_2 = \theta_6$)
  - RFM-fault at L_1

- **Stage 4 (dec, $i = 1$):**
  - Target K_{7i} ($= \theta_1 = \theta_5$)
  - RFM-fault at L_2

Fault Analysis of Bel-T-128



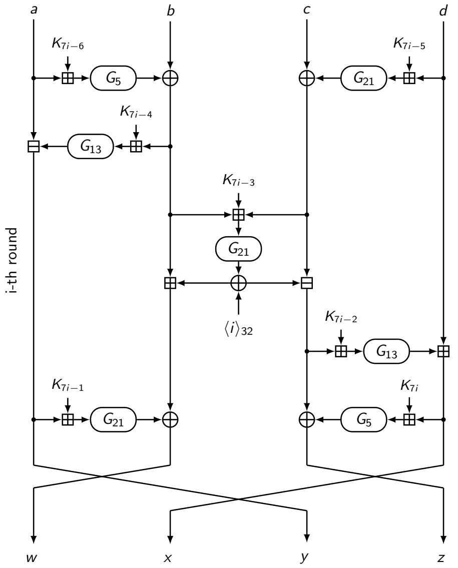
- **Recovered key parts:**

  θ_1 θ_2 θ_3 θ_4

  θ_5 θ_6 θ_7 θ_8

- **Stage 3 (dec, $i = 1$):**
  - Target K_{7i-1} $(= \theta_2 = \theta_6)$
  - RFM-fault at L_1
- **Stage 4 (dec, $i = 1$):**
  - Target K_{7i} $(= \theta_1 = \theta_5)$
  - RFM-fault at L_2

Fault Analysis of Bel-T-192



- **Recovered key parts:**

  $\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6$

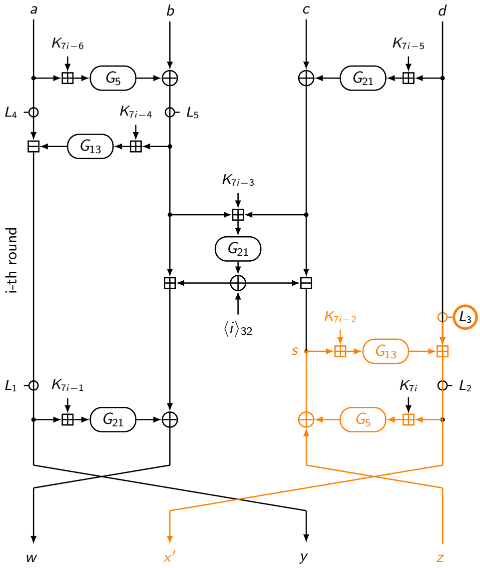
  $\theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3$

  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

- **Stages 1-4:** Recover θ_1, θ_2, θ_7, θ_8 as shown for Bel-T-128
- **Stage 5 (enc, $i = 8$):**
  - Target K_{7i-2} $(= \theta_6)$
  - CFM-fault at L_3
  - Solve: $x' = G_{13}(s \boxplus \theta_6) \boxplus 0$

  $s = G_5(x \boxplus \theta_8) \oplus z$

Fault Analysis of Bel-T-192



- **Recovered key parts:**

  $\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6$

  $\theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3$

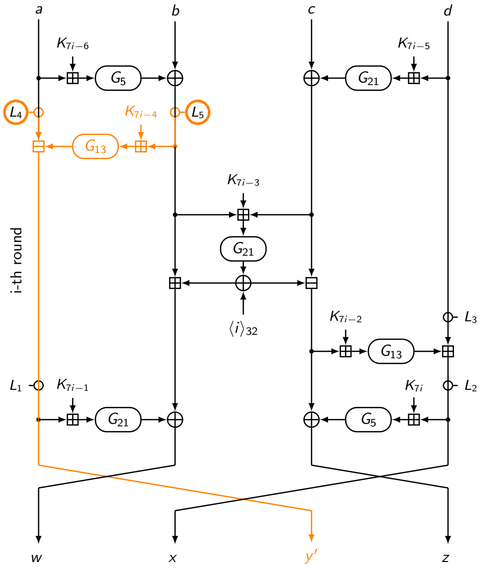
  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

- **Stages 1-4:** Recover θ_1, θ_2, θ_7, θ_8 as shown for Bel-T-128
- **Stage 5 (enc, $i = 8$):**
  - Target K_{7i-2} $(= \theta_6)$
  - CFM-fault at L_3
  - Solve:

$$x' = G_{13}(s \boxplus \theta_6) \boxplus 0$$

$$s = G_5(x \boxplus \theta_8) \oplus z$$

Fault Analysis of Bel-T-192



- **Recovered key parts:**

  $\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6$

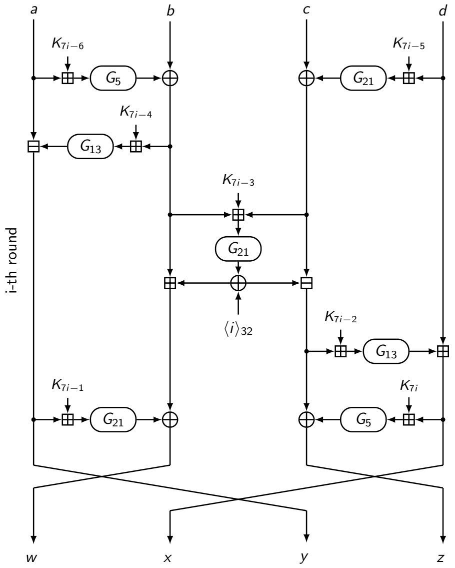
  $\theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3$

  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

- **Stage 6 (enc, $i = 8$):**
  - Target K_{7i-4} $(= \theta_4)$
  - Dual CFM-faults at L_4 and L_5
  - Solve: $y' = 0 \boxminus G_{13}(0 \boxplus \theta_4)$

- **Finally:**
  - $\theta_3 = \theta_1 \oplus \theta_2 \oplus \theta_7$
  - $\theta_5 = \theta_4 \oplus \theta_6 \oplus \theta_8$

Fault Analysis of Bel-T-192



- **Recovered key parts:**

  $\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6$

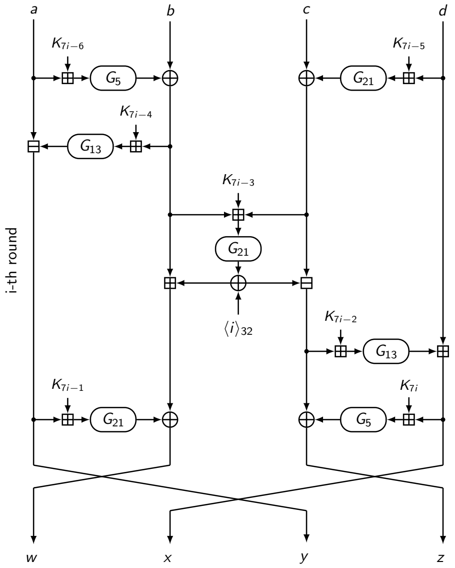
  $\theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3$

  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$

- **Stage 6 (enc, $i = 8$):**
  - Target K_{7i-4} $(= \theta_4)$
  - Dual CFM-faults at L_4 and L_5
  - Solve: $y' = 0 \boxminus G_{13}(0 \boxplus \theta_4)$

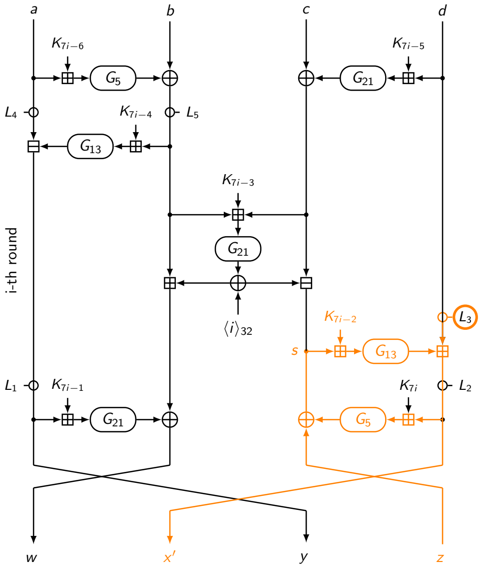
- **Finally:**
  - $\theta_3 = \theta_1 \oplus \theta_2 \oplus \theta_7$
  - $\theta_5 = \theta_4 \oplus \theta_6 \oplus \theta_8$

Fault Analysis of Bel-T-256



- **Recovered key parts:**
  θ_1 θ_2 θ_3 θ_4
  θ_5 θ_6 θ_7 θ_8
- **Stages 1-6:** Recover θ_1, θ_2, θ_4, θ_6, θ_7, θ_8 as shown for Bel-T-192
- **Stage 7 (dec, $i = 1$):**
  - Target K_{7i-2} $(= \theta_3)$
  - CFM-fault at L_3
- **Stage 8 (dec, $i = 1$):**
  - Target K_{7i-4} $(= \theta_5)$
  - Dual CFM-faults at L_4 and L_5

Fault Analysis of Bel-T-256



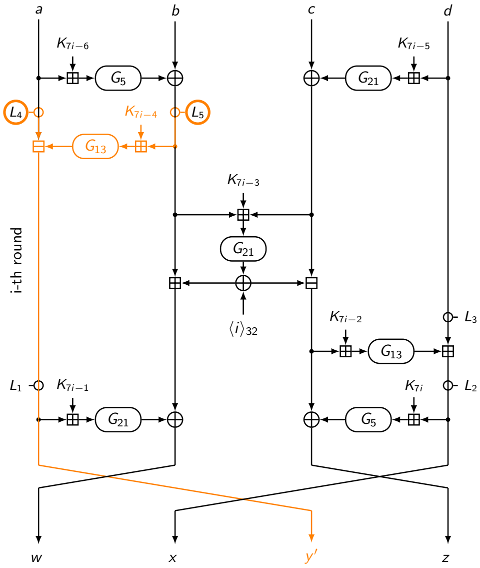
- **Recovered key parts:**

  θ_1 θ_2 θ_3 θ_4

  θ_5 θ_6 θ_7 θ_8

- **Stages 1-6:** Recover θ_1, θ_2, θ_4, θ_6, θ_7, θ_8 as shown for Bel-T-192
- **Stage 7 (dec, $i = 1$):**
  - Target K_{7i-2} $(= \theta_3)$
  - CFM-fault at L_3
- **Stage 8 (dec, $i = 1$):**
  - Target K_{7i-4} $(= \theta_5)$
  - Dual CFM-faults at L_4 and L_5

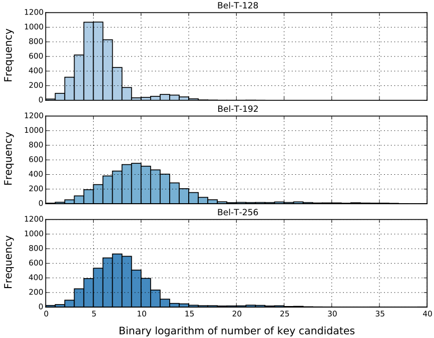
Fault Analysis of Bel-T-256



- **Recovered key parts:**
  θ_1 θ_2 θ_3 θ_4
  θ_5 θ_6 θ_7 θ_8
- **Stages 1-6:** Recover θ_1, θ_2, θ_4, θ_6, θ_7, θ_8 as shown for Bel-T-192
- **Stage 7 (dec, $i = 1$):**
  - Target K_{7i-2} ($= \theta_3$)
  - CFM-fault at L_3
- **Stage 8 (dec, $i = 1$):**
  - Target K_{7i-4} ($= \theta_5$)
  - Dual CFM-faults at L_4 and L_5

Experimental Results

		Bel-T-128	Bel-T-192	Bel-T-256
#keys	avg	5.11	10.06	7.63
($\log_2(x)$)	med	4.58	9.17	7.00
	min	0.00	0.00	0.00
	max	22.00	40.00	39.00
time/attack	sec	148	287	687



Fault Analysis of Bel-T

Summary

- First differential fault analysis of Bel-T
- Full key recovery (using at least #faults):

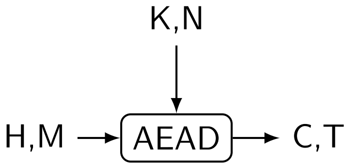
key size	RFM-faults	CFM-faults	total
128	4	0	4
192	4	2	6
256	4	6	10

- Analysis computationally inexpensive
- Extensive simulation-based experiments for verification of the developed attacks

Part II: Cryptography

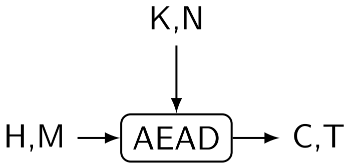
NORX: Parallel and Scalable Authenticated Encryption

Authenticated Encryption with Associated Data (AEAD)



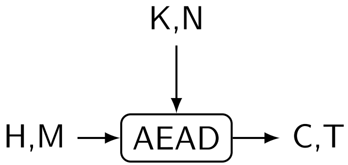
- Input: key K, nonce N, associated data H, message M
- Output: ciphertext C, authentication tag T
- Protects:
  - **confidentiality**, and **integrity/authenticity** of M
  - **integrity/authenticity** of N and H
- Realisation:
  - generic composition
  - block cipher modes
  - dedicated schemes
  - **sponge functions**
- Applications: IPsec, SSH, SSL/TLS, etc.

Authenticated Encryption with Associated Data (AEAD)



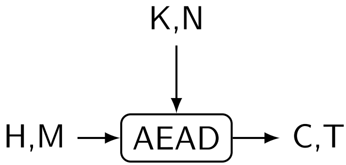
- Input: key K, nonce N, associated data H, message M
- Output: ciphertext C, authentication tag T
- Protects:
  - **confidentiality**, and **integrity/authenticity** of M
  - **integrity/authenticity** of N and H
- Realisation:
  - generic composition
  - block cipher modes
  - dedicated schemes
  - **sponge functions**
- Applications: IPsec, SSH, SSL/TLS, etc.

Authenticated Encryption with Associated Data (AEAD)



- Input: key K, nonce N, associated data H, message M
- Output: ciphertext C, authentication tag T
- Protects:
  - **confidentiality**, and **integrity/authenticity** of M
  - **integrity/authenticity** of N and H
- Realisation:
  - generic composition
  - block cipher modes
  - dedicated schemes
  - **sponge functions**
- Applications: IPsec, SSH, SSL/TLS, etc.

Authenticated Encryption with Associated Data (AEAD)



- Input: key K, nonce N, associated data H, message M
- Output: ciphertext C, authentication tag T
- Protects:
  - **confidentiality**, and **integrity/authenticity** of M
  - **integrity/authenticity** of N and H
- Realisation:
  - generic composition
  - block cipher modes
  - dedicated schemes
  - **sponge functions**
- Applications: IPsec, SSH, SSL/TLS, etc.

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

What's Wrong with Current Solutions?

Example: AES-GCM

- Complex
- Needs HW support for AES/Galois field arithmetic (x86: AESNI) to be fast ...
- ... otherwise slow and hard to implement in constant-time
- Nonce re-use: easy to recover authentication key
- Used basically everywhere:
  - NSA Suite B
  - NIST SP 800-38D
  - IPsec
  - SSH
  - SSL/TLS
  - IEEE 802.11ad (WiGig)
  - ...
- Only few alternatives

In summary: **lots of room for improvements**

CAESAR

- **C**ompetition for **A**uthenticated **E**ncryption: **S**ecurity, **A**pplicability, and **R**obustness
- **Goals**: Identify portfolio of *authenticated ciphers* that
  - offer advantages over AES-GCM (the current de-facto standard) and
  - are suitable for widespread adoption
- **Overview**:
  - 1st round
    - March 15, 2014
    - 57 candidates
  - 2nd round
    - July 7, 2015
    - 30 candidates
  - Announcement of final portfolio: $\approx$ 2017

CAESAR

- **C**ompetition for **A**uthenticated **E**ncryption: **S**ecurity, **A**pplicability, and **R**obustness

- **Goals**: Identify portfolio of *authenticated ciphers* that
  - offer advantages over AES-GCM
    (the current de-facto standard) and
  - are suitable for widespread adoption

- **Overview**:
  - 1st round
    - March 15, 2014
    - 57 candidates
  - 2nd round
    - July 7, 2015
    - 30 candidates
  - Announcement of final portfolio: $\approx$ 2017

CAESAR

- **C**ompetition for **A**uthenticated **E**ncryption: **S**ecurity, **A**pplicability, and **R**obustness
- **Goals**: Identify portfolio of *authenticated ciphers* that
  - offer advantages over AES-GCM (the current de-facto standard) and
  - are suitable for widespread adoption
- **Overview**:
  - 1st round
    - March 15, 2014
    - 57 candidates
  - 2nd round
    - July 7, 2015
    - 30 candidates
  - Announcement of final portfolio: $\approx$ 2017

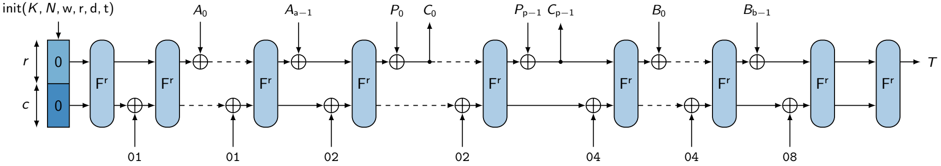
Specification of NORX

UNIVERSITÄT PASSAU

Main Design Goals

- High security
- Efficiency
- Simplicity
- Scalability
- Online
- Side-channel robustness (esp. against timing attacks)
- No AES dependence

NORX



NORX in Sequential Mode (d = 1)

Parameters

Word size	Number of rounds	Parallelism degree	Tag size
$w \in \{32, 64\}$	$1 \leq r \leq 63$	$0 \leq d \leq 255$	$t \leq 10w$

Features

- (Parallelisable) *monkeyDuplex* construction (derived from Keccak/SHA-3)
- Process header A, payload P and trailer data T in one-pass
- Data expansion via *multi-rate padding*: $X \parallel 1 \parallel 0^* \parallel 1$

The State

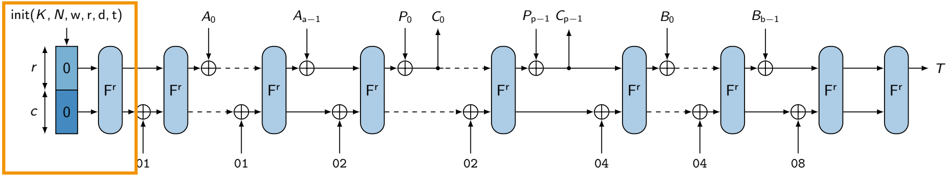
- NORX has an internal state S of *16* w-bit sized words:

w	Size	Rate	Capacity
32	512	320	192
64	1024	640	384

- Assembly of **rate** (data processing) and **capacity** (security) words:

s_0	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}

Initialisation



Initialisation

- Load **nonce**, **key** and **constants** into state S:

u_0	n_0	n_1	u_1
k_0	k_1	k_2	k_3
u_2	u_3	u_4	u_5
u_6	u_7	u_8	u_9

- Parameter integration:

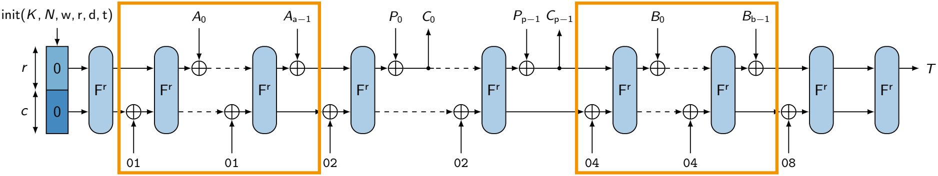
$$
\begin{aligned}
s_{12} &\leftarrow s_{12} \oplus \mathsf{w}\\
s_{13} &\leftarrow s_{13} \oplus \mathsf{r}\\
s_{14} &\leftarrow s_{14} \oplus \mathsf{d}\\
s_{15} &\leftarrow s_{15} \oplus \mathsf{t}
\end{aligned}
$$

- Apply round permutation:

$$S \leftarrow \mathsf{F}^{\mathsf{r}}(S)$$

Header/Trailer Absorption

UNIVERSITÄT PASSAU



Header/Trailer Absorption

- Integrate domain separation constant:

$$s_{15} \leftarrow s_{15} \oplus \{01, 04\}$$

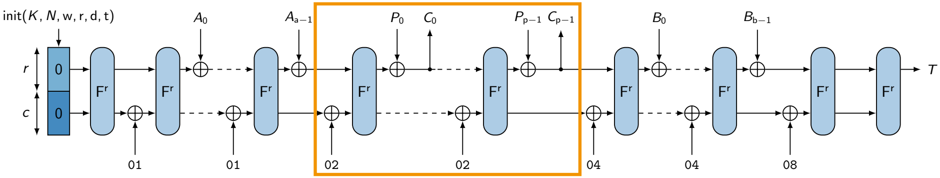
- Apply round permutation:

$$S \leftarrow \mathsf{F}^{\mathsf{r}}(S)$$

- Absorb associated-data block:

$$\begin{pmatrix} s'_0 & s'_1 & s'_2 & s'_3 \\ s'_4 & s'_5 & s'_6 & s'_7 \\ s'_8 & s'_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \leftarrow \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \oplus \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Payload Encryption



Payload Encryption

- Integrate domain separation constant:

$$s_{15} \leftarrow s_{15} \oplus 02$$

- Apply round permutation:

$$S \leftarrow \mathsf{F}^{\mathsf{r}}(S)$$

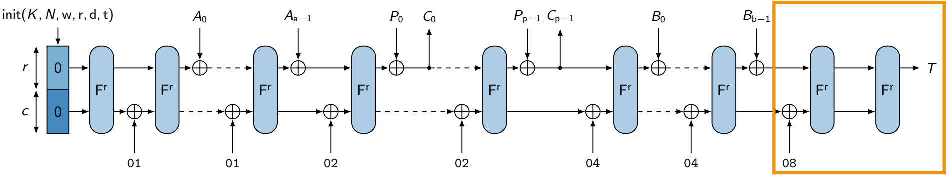
- Absorb message block:

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \leftarrow \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \oplus \begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- Set new ciphertext block:

$$c = (c_0, \ldots, c_9)$$

Tag Generation



Tag Generation

- Integrate domain separation constant:

$$s_{15} \leftarrow s_{15} \oplus 08$$

- Apply round permutation twice:

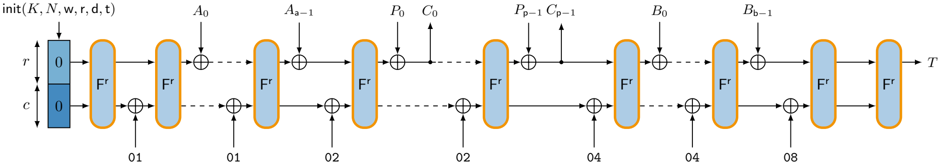
$$S \leftarrow \mathsf{F}^{\mathsf{r}}(S)$$

$$S \leftarrow \mathsf{F}^{\mathsf{r}}(S)$$

- Set authentication tag:

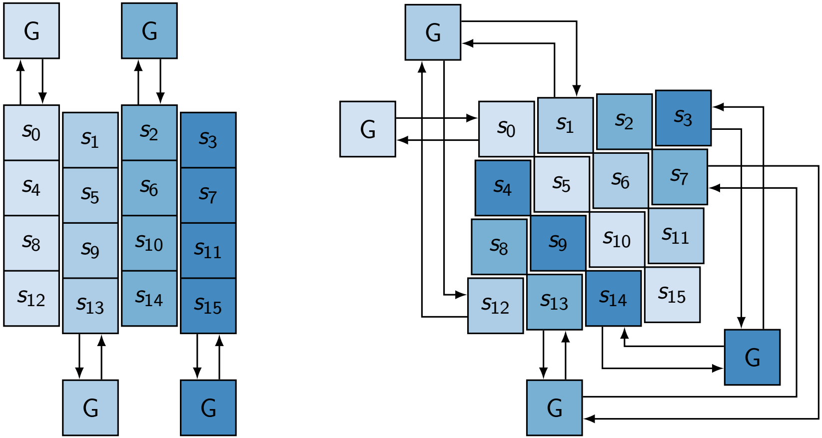
$$t = (s_0, s_1, s_2, s_3)$$

The Permutation F^{r}



The Permutation F^r

- F^r: r iterations of the permutation F
- F: (1) apply G to columns of S, (2) apply G to diagonals of S



The Permutation G

Components

- Non-linear operation

$$\mathrm{H}(x, y) = (x \oplus y) \oplus \big((x \wedge y) \ll 1\big)$$

- Cyclic rotation

$$\mathrm{ROTR}(x, r) = x \ggg r$$

- Rotation offsets (r_0, r_1, r_2, r_3)
  - 32-bit: $(8, 11, 16, 31)$
  - 64-bit: $(8, 19, 40, 63)$

$\mathrm{G}(a, b, c, d)$:

1: $a \longleftarrow \mathrm{H}(a, b)$
2: $d \longleftarrow \mathrm{ROTR}(a \oplus d, r_0)$
3: $c \longleftarrow \mathrm{H}(c, d)$
4: $b \longleftarrow \mathrm{ROTR}(b \oplus c, r_1)$
5: $a \longleftarrow \mathrm{H}(a, b)$
6: $d \longleftarrow \mathrm{ROTR}(a \oplus d, r_2)$
7: $c \longleftarrow \mathrm{H}(c, d)$
8: $b \longleftarrow \mathrm{ROTR}(b \oplus c, r_3)$

Properties of $F^r/F/G$

Features

- Derived from ARX-primitives ChaCha20 and BLAKE2
- Non-linear operation

$$\mathrm{H}(x, y) = (x \oplus y) \oplus \big((x \wedge y) \ll 1\big)$$

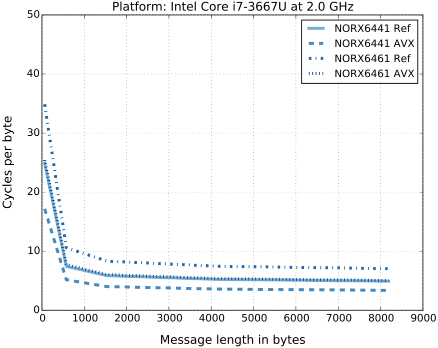
  is an "approximation" of integer addition

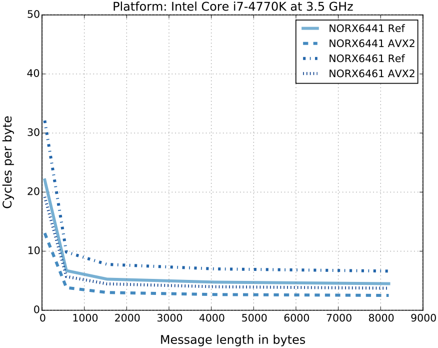
$$x + y = (x \oplus y) + \big((x \wedge y) \ll 1\big)$$

- LRX-primitive:
  - only bit-wise logical operations
  - no SBoxes
  - no integer additions
- Constant-time
- SIMD-friendly
- Hardware-friendly
- High diffusion

Performance Evaluation of NORX

Software Performance (x86)

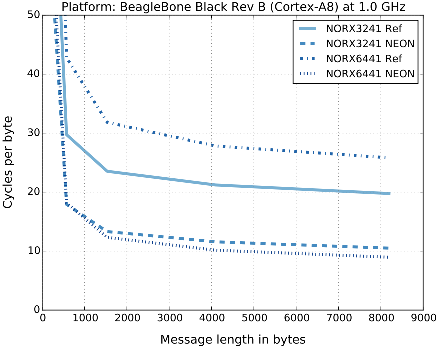


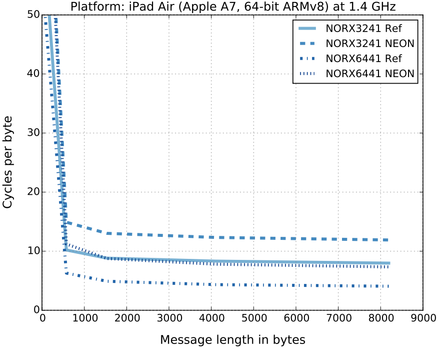


Platform	Implementation	cpb	MiBps
Ivy Bridge: i7 3667U @ 2.0 GHz	AVX	3.37	593
Haswell: i7 4770K @ 3.5 GHz	AVX2	2.51	1390

NORX64-4-1 performance

Software Performance (ARM)

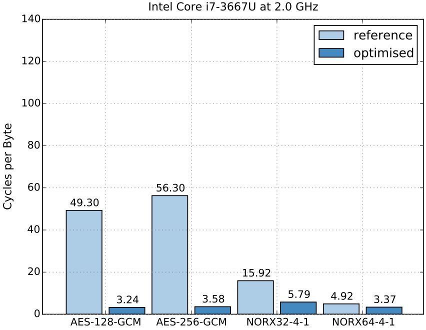


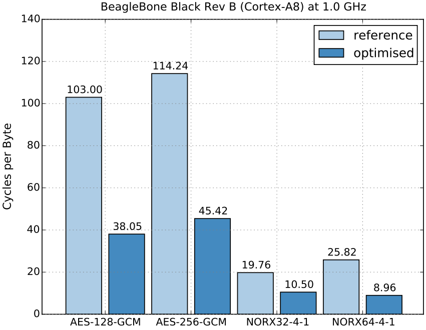


Platform	Implementation	cpb	MiBps
BBB: Cortex-A8 @ 1.0 GHz	NEON	8.96	111
iPad Air: Apple A7 @ 1.4 GHz	Ref	4.07	343

NORX64-4-1 performance

NORX vs AES-GCM





AES-GCM "standard": OpenSSL 1.0.1j compiled with no-asm flag

`> openssl speed -evp aes-{128,256}-gcm`

- **x86:** NORX slightly slower than AES-GCM (due to AESNI)
- **ARM:** NORX much faster than AES-GCM

SW Performance (SUPERCOP)

UNIVERSITÄT PASSAU

titan0	wintermute	sachr	hydra9	h6dragon	flops	h6sandy	gcc16	floodyberry	h1mips
aegis128l	aes128gcmv1	morus640128v1	aegis128l	norx6441v1	morus640128v1	morus640128v1	morus640128v1	norx6441v1	morus1280128v1
tiaoxinv1	aes256gcmv1	morus1280128v1	tiaoxinv1	morus640128v1	norx3241v1	morus1280128v1	morus1280256v1	norx6461v1	morus1280256v1
aegis128	norx6441v1	morus1280256v1	aegis128	norx3241v1	hs1sivlov1	morus1280256v1	morus1280128v1	norx3241v1	hs1sivlov1
aegis256	norx6461v1	norx6441v1	aegis256	morus1280128v1	norx6441v1	norx6441v1	hs1sivlov1	norx6444v1	wheeshtv1mr3fr1t256
kiasuneq128v1	norx3241v1	aegis128l	kiasuneq128v1	morus1280256v1	morus1280128v1	norx3241v1	norx6441v1	ascon96v1	wheeshtv1mr3fr1t128
morus1280128v1	norx6444v1	norx6461v1	silverv1	norx6461v1	morus1280256v1	norx6461v1	hs1sivv1	morus640128v1	norx3241v1
morus1280256v1	morus640128v1	aegis128	morus1280128v1	hs1sivlov1	norx3261v1	hs1sivlov1	ascon96v1	norx3261v1	aegis128l
silverv1	norx3261v1	norx3241v1	deoxysneq128128v1	norx3261v1	aegis128l	aes128gcmv1	norx6444v1	aes128gcmv1	hs1sivv1
morus640128v1	ascon96v1	aegis256	deoxysneq256128v1	aegis128l	wheeshtv1mr3fr1t128	tiaoxinv1	norx3241v1	aes256gcmv1	wheeshtv1mr3fr3t256
kiasueq128v1	ascon128v1	norx3261v1	kiasueq128v1	wheeshtv1mr3fr1t256	wheeshtv1mr3fr1t256	wheeshtv1mr3fr1t128	norx6461v1	ascon128v1	aegis128
deoxysneq128128v1	aes128otrsv1	aes128gcmv1	morus1280256v1	wheeshtv1mr3fr1t128	hs1sivv1	wheeshtv1mr3fr1t256	aegis128l	pi64cipher128v1	norx6441v1
deoxysneq256128v1	aes128otrpv1	norx6444v1	deoxyseq128128v1	tiaoxinv1	norx6461v1	aegis128l	aegis128	pi64cipher256v1	norx3261v1
aes128gcmv1	aes256otrsv1	aes256gcmv1	morus640128v1	aegis128	aegis128	aes256gcmv1	ascon128v1	aes128otrsv1	aegis256
aes256gcmv1	aesjambuv1	aes128otrsv1	deoxyseq256128v1	wheeshtv1mr3fr3t256	wheeshtv1mr3fr3t256	hs1sivv1	pi64cipher256v1	aes128otrpv1	tiaoxinv1
deoxyseq128128v1	aes256otrpv1	aes128otrpv1	aes128gcmv1	hs1sivv1	aegis256	wheeshtv1mr3fr3t256	pi64cipher128v1	omdsha512k256n256tau256v1	norx6444v1
aes128cpfbv1	omdsha512k512n256tau256v1	pi32cipher256v1	aes128cpfbv1	aegis256	aes128gcmv1	norx3261v1	aegis256	omdsha512k512n256tau256v1	hs1sivhiv1
deoxyseq256128v1	omdsha512k256n256tau256v1	pi32cipher128v1	aes256gcmv1	aes128gcmv1	aes256gcmv1	aegis128	silverv1	omdsha512k128n128tau128v1	norx6461v1
norx6441v1	omdsha512k128n128tau128v1	aes256otrsv1	aes256cpfbv1	hs1sivhiv1	hs1sivhiv1	norx6444v1	icepole256av1	pi32cipher256v1	silverv1
aes256cpfbv1	stribob192r1	aesjambuv1	norx6441v1	aes256gcmv1	wheeshtv1mr5fr7t256	aegis256	icepole128av1	pi32cipher128v1	wheeshtv1mr5fr7t256
norx6461v1	omdsha256k128n96tau64v1	aes256otrpv1	hs1sivlov1	norx6444v1	cba1	hs1sivhiv1	icepole128v1	aesjambuv1	yaes128v2
hs1sivlov1	omdsha256k128n96tau96v1	kiasuneq128v1	wheeshtv1mr3fr1t128	wheeshtv1mr5fr7t256	cba3	kiasuneq128v1	norx3261v1	aes256otrsv1	aes128otrsv1
norx3241v1	omdsha256k192n104tau128v1	ascon96v1	wheeshtv1mr3fr1t256	cba8	cba2	scream10v2	aes128cpfbv1	aes256otrpv1	aes128otrpv1
wheeshtv1mr3fr1t128	omdsha256k128n96tau128v1	pi64cipher256v1	norx3241v1	cba1	cba4	scream10v1	pi32cipher128v1	omdsha256k192n104tau128v1	icepole256av1
wheeshtv1mr3fr1t256	omdsha256k256n104tau160v1	pi64cipher128v1	norx6461v1	cba4	cba6	ascon96v1	pi32cipher256v1	omdsha256k256n104tau160v1	icepole128v1
hs1sivv1	omdsha256k256n248tau256v1	ascon128v1	wheeshtv1mr3fr3t256	cba5	cba5	scream12v2	aes128otrsv1	omdsha256k128n96tau64v1	icepole128av1
norx6444v1	lakekeyakv1	kiasueq128v1	hs1sivv1	aes128cpfbv1	cba8	wheeshtv1mr5fr7t256	yaes128v2	omdsha256k128n96tau128v1	cba1
wheeshtv1mr3fr3t256	seakeyakv1	stribob192r1	norx6444v1	cba3	aes128cpfbv1	scream12v1	aes128otrpv1	omdsha256k128n96tau96v1	cba3
ascon96v1	oceankeyakv1	trivia0v1	ascon96v1	silverv1	cba7	iscream12v2	cba2	omdsha256k256n248tau256v1	cba2
norx3261v1	ketjesrv1	ifeedaes128n104v1	aesjambuv1	cba2	aeadaes128ocbtaglen96v1	iscream12v1	cba3	lakekeyakv1	cba7
hs1sivhiv1	riverkeyakv1	ifeedaes128n96v1	norx3261v1	cba7	aeadaes128ocbtaglen128v1	pi64cipher256v1	cba1	seakeyakv1	cba4
pi64cipher256v1	ketjejrv1	lakekeyakv1	wheeshtv1mr5fr7t256	cba6	silverv1	icepole128av1	cba4	trivia0v1	cba5
aesjambuv1	trivia0v1	riverkeyakv1	hs1sivhiv1	aeadaes128ocbtaglen128v1	aeadaes128ocbtaglen64v1	silverv1	cba8	stribob192r1	cba8
scream10v2	aes128poetv1aes4	seakeyakv1	scream10v1	aeadaes128ocbtaglen64v1	cba9	icepole256av1	cba5	oceankeyakv1	cba6
scream10v1	aes128poetv1aes128	pi16cipher096v1	scream10v2	aeadaes128ocbtaglen96v1	aeadaes192ocbtaglen96v1	icepole128v1	cba7	pi16cipher096v1	aeadaes128ocbtaglen96v1
icepole256av1	aes128poetv1gf128mul	pi16cipher128v1	ascon128v1	cba9	aeadaes192ocbtaglen64v1	iscream14v2	cba6	pi16cipher128v1	aeadaes128ocbtaglen64v1
icepole128av1		lacv11	icepole128av1	yaes128v2	aeadaes192ocbtaglen128v1	iscream14v1	aes128n8clocv1	riverkeyakv1	aeadaes128ocbtaglen128v1
icepole128v1		ketjesrv1	icepole128v1	aeadaes192ocbtaglen128v1	cba10	aes128cpfbv1	aes128n12clocv1	ketjesrv1	aes128n8clocv1
ascon128v1		oceankeyakv1	icepole256av1	aeadaes192ocbtaglen96v1	yaes128v2	kiasueq128v1	aeadaes128ocbtaglen128v1	ketjejrv1	aes128n12clocv1
scream12v2		deoxysneq128128v1	pi64cipher256v1	aeadaes192ocbtaglen64v1	norx6444v1	pi64cipher128v1	aeadaes128ocbtaglen96v1	aes128poetv1aes4	aes256otrsv1
scream12v1		ketjejrv1	scream12v2	cba10	aeadaes256ocbtaglen128v1	ascon128v1	aeadaes128ocbtaglen64v1	aes128poetv1aes128	aes128cpfbv1
iscream12v2		deoxysneq256128v1	scream12v1	aeadaes256ocbtaglen128v1	aeadaes256ocbtaglen64v1	aes128otrsv1	aes128n12silcv1	aes128poetv1gf128mul	ascon96v1

Source: `http://www1.spms.ntu.edu.sg/~syllab/speed`

- NORX among the fastest CAESAR ciphers
- Fastest sponge-based scheme
- Reference implementation has competitive speed, too

What's Next?

Current Research

- NORX8/NORX16:
  - low-end devices
  - entry-level targets for cryptanalysis
- Misuse-resistant sponges

Open Tasks

- Comprehensive hardware evaluation
- Extend security analysis

What's Next?

UNIVERSITÄT PASSAU

Current Research

- NORX8/NORX16:
  - low-end devices
  - entry-level targets for cryptanalysis
- Misuse-resistant sponges

Open Tasks

- Comprehensive hardware evaluation
- Extend security analysis

Summary

Part I: Cryptanalysis

- Multi-Stage Fault Attacks on
  - LED
  - PRINCE
  - Bel-T
- Algebraic Fault Attacks on LED64

Part II: Cryptography

- NORX: Parallel and Scalable Authenticated Encryption
- Security Evaluation of NORX
  - General, algebraic, differential, rotational properties
  - NODE: (NO)RX (D)ifferential Search (E)ngine

Thank you!

Summary

Part I: Cryptanalysis

- Multi-Stage Fault Attacks on
  - LED
  - PRINCE
  - Bel-T
- Algebraic Fault Attacks on LED64

Part II: Cryptography

- NORX: Parallel and Scalable Authenticated Encryption
- Security Evaluation of NORX
  - General, algebraic, differential, rotational properties
  - NODE: (NO)RX (D)ifferential Search (E)ngine

Thank you!