Clustering Effect in Simon and Simeck

Gaëtan Leurent[1], Clara Pernot[1] and André Schrottenloher[2]

[1]Inria, Paris
[2]CWI, Amsterdam

Friday, October 1st 2021

Table of contents

Table of contents

Table of contents

Overview

Introduction of two lightweight block ciphers by NSA researchers in 2013:

- **Simon** optimized in hardware [BTSWSW, DAC'15]
- **Speck** optimized in software [BTSWSW, DAC'15]

Overview

Introduction of two lightweight block ciphers by NSA researchers in 2013:

- **Simon** optimized in hardware [BTSWSW, DAC'15]
- **Speck** optimized in software [BTSWSW, DAC'15]

Attempt of ISO standardization...
But some experts were **suspicious** about:

→ the lack of clear need for standardisation of the new ciphers

→ NSA's previous involvement in the creation and promotion of backdoored cryptographic algorithm

More than **70 papers** study **Simon** and **Speck**!

Overview

Introduction of two lightweight block ciphers by NSA researchers in 2013:

- **Simon** optimized in hardware [BTSWSW, DAC'15]
- **Speck** optimized in software [BTSWSW, DAC'15]

Attempt of ISO standardization...
But some experts were **suspicious** about:

→ the lack of clear need for standardisation of the new ciphers
→ NSA's previous involvement in the creation and promotion of backdoored cryptographic algorithm

More than **70 papers** study **Simon** and **Speck**!

⇒ A variant of **Simon** and **Speck**: **Simeck**. [YZSAG, CHES'15]

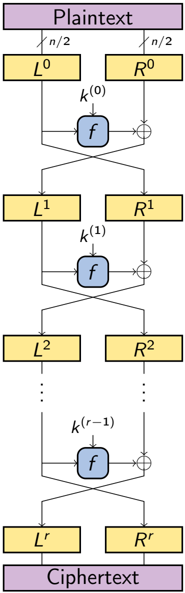
Summary of previous and new attacks

Cipher	Rounds	Attacked	Ref	Note
Simeck48/96	36	30	[QCW'16]	Linear † ‡
		32	**New**	Linear
Simeck64/128	44	37	[QCW'16]	Linear † ‡
		42	**New**	Linear
Simon96/96	52	37	[WWJZ'18]	Differential
		43	**New**	Linear
Simon96/144	54	38	[CW'16]	Linear
		45	**New**	Linear
Simon128/128	68	50	[WWJZ'18]	Differential
		53	**New**	Linear
Simon128/192	69	51	[WWJZ'18]	Differential
		55	**New**	Linear
Simon128/256	72	53	[CW'16]	Linear
		56	**New**	Linear

†The advantage is too low to do a key-recovery.

‡Attack use the duality between linear and differential distinguishers.

Feistel cipher



A **Feistel network** is characterized by:

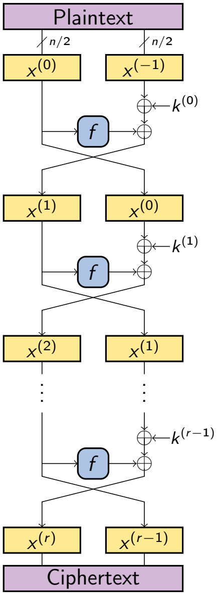
- its block size: n
- its key size: κ
- its number of round: r
- its round function: f

For each round $i = 0, \ldots, r-1$:

$$\begin{cases} R^{i+1} & = & L^i \\ L^{i+1} & = & R^i \oplus f(L^i, k^{(i)}) \end{cases}$$

Example: Data Encryption Standard (DES).

Feistel cipher



A **Feistel network** is characterized by:

- its block size: n
- its key size: κ
- its number of round: r
- its round function: f

For each round $i = 0, \ldots, r-1$:

$$\boxed{x^{(i+1)} = x^{(i-1)} \oplus f(x^{(i)}) \oplus k^{(i)}}$$

Example: Data Encryption Standard (DES).

Simon, Speck and Simeck

→ **Simon** is a Feistel network with a **quadratic round function**:

$$f(x) = ((x \lll 8) \wedge (x \lll 1)) \oplus (x \lll 2)$$

and a linear key schedule. [BTSWSW'15]

→ **Speck** is an Add-Rotate-XOR (ARX) cipher:

$$R_k(x, y) = \big(((x \lll \alpha) \boxplus y) \oplus k, (y \lll \beta) \oplus ((x \lll \alpha) \boxplus y) \oplus k\big)$$

which reuses its **round function** R_k in the **key schedule**.
[BTSWSW'15]

Simon, Speck and Simeck

→ **Simon** is a Feistel network with a **quadratic round function**:

$$f(x) = ((x \lll 8) \wedge (x \lll 1)) \oplus (x \lll 2)$$

and a linear key schedule. [BTSWSW'15]

→ **Speck** is an Add-Rotate-XOR (ARX) cipher:

$$R_k(x, y) = \big(((x \lll \alpha) \boxplus y) \oplus k, (y \lll \beta) \oplus ((x \lll \alpha) \boxplus y) \oplus k\big)$$

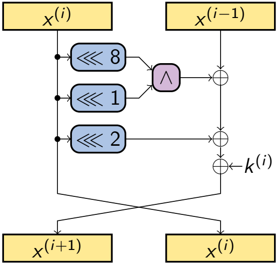
which reuses its **round function** R_k in the **key schedule**. [BTSWSW'15]

→ **Simeck** is a Feistel network with a **quadratic round function**:

$$f(x) = ((x \lll 5) \wedge x) \oplus (x \lll 1)$$

which reuses its **round function** f in the **key schedule**. [YZSAG'15]

Simon and Simeck

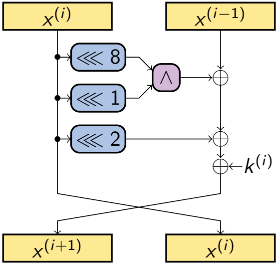


Simon round function

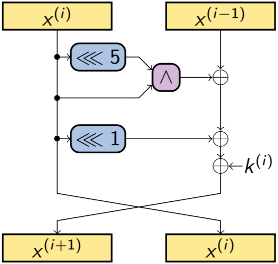
n (block size)	32	48		64		96		128		
κ (key size)	64	72	96	96	128	96	144	128	192	256
r (rounds)	32	36	36	42	44	52	54	68	69	72

$\rightarrow$ Linear key schedule.

Simon and Simeck



Simon round function



Simeck round function

n (block size)	32	48		64		96		128		
κ (key size)	64	72	96	96	128	96	144	128	192	256
r (rounds)	32	36	36	42	44	52	54	68	69	72

n	32	48	64
κ	64	96	128
r	32	36	44

$\rightarrow$ Linear key schedule.

$\rightarrow$ Non-linear key schedule which reuses f.

Table of contents

Differential Cryptanalysis (1)

A **differential** is a pair (δ, δ') such that: [BS, CRYPTO'90]

$$\Pr_{k,x}[E_k(x) \oplus E_k(x \oplus \delta) = \delta'] \gg 2^{-n}$$

Differential Cryptanalysis (1)

A **differential** is a pair (δ, δ') such that: [BS, CRYPTO'90]

$$\Pr_{k,x}[E_k(x) \oplus E_k(x \oplus \delta) = \delta'] \gg 2^{-n}$$

To obtain a differential with a high probability, we use **differential characteristic** (or trail) to specify the intermediate state difference after each round: $(\delta_0, \delta_1, \ldots, \delta_r)$.
$\rightarrow$ for one round:

$$\Pr[\delta \rightarrow \delta'] = \Pr_x[R(x) \oplus R(x \oplus \delta) = \delta']$$

Differential Cryptanalysis (1)

A **differential** is a pair (δ, δ') such that: [BS, CRYPTO'90]

$$\Pr_{k,x}[E_k(x) \oplus E_k(x \oplus \delta) = \delta'] \gg 2^{-n}$$

To obtain a differential with a high probability, we use **differential characteristic** (or trail) to specify the intermediate state difference after each round: $(\delta_0, \delta_1, \ldots, \delta_r)$.
$\rightarrow$ for one round:

$$\Pr[\delta \rightarrow \delta'] = \Pr_x[R(x) \oplus R(x \oplus \delta) = \delta']$$

Simon and **Simeck** with independent round keys are Markov ciphers, so according to Lai, Massey and Murphy [EC'91]:
$\rightarrow$ for one trail on r rounds:

$$\Pr[\delta_0 \rightarrow \delta_1 \rightarrow \ldots \rightarrow \delta_r] = \prod_{i=1}^{r} \Pr[\delta_{i-1} \rightarrow \delta_i]$$

Differential Cryptanalysis (1)

A **differential** is a pair (δ, δ') such that: [BS, CRYPTO'90]

$$\Pr_{k,x}[E_k(x) \oplus E_k(x \oplus \delta) = \delta'] \gg 2^{-n}$$

To obtain a differential with a high probability, we use **differential characteristic** (or trail) to specify the intermediate state difference after each round: $(\delta_0, \delta_1, \ldots, \delta_r)$.
$\rightarrow$ for one round:

$$\Pr[\delta \rightarrow \delta'] = \Pr_x[R(x) \oplus R(x \oplus \delta) = \delta']$$

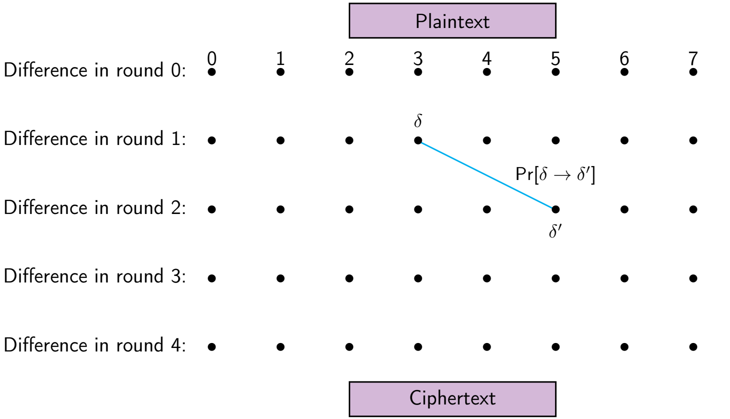
Simon and **Simeck** with independent round keys are Markov ciphers, so according to Lai, Massey and Murphy [EC'91]:
$\rightarrow$ for one trail on r rounds:

$$\Pr[\delta_0 \rightarrow \delta_1 \rightarrow \ldots \rightarrow \delta_r] = \prod_{i=1}^{r} \Pr[\delta_{i-1} \rightarrow \delta_i]$$

$\rightarrow$ for all trails on r rounds:

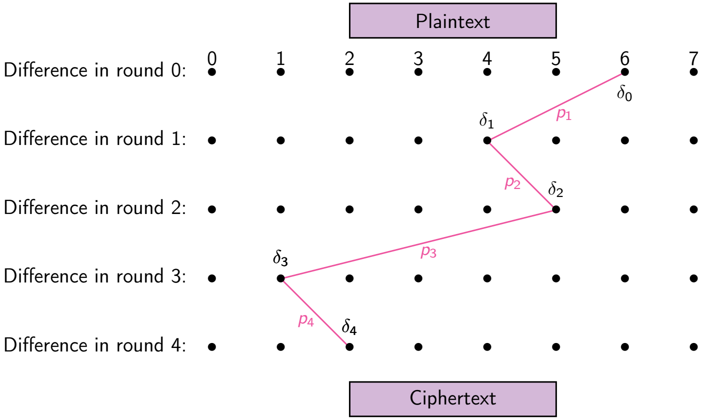
$$\Pr[\delta_0 \overset{r}{\rightsquigarrow} \delta_r] = \sum_{\delta_1, \delta_2, \ldots \delta_{r-1}} \prod_{i=1}^{r} \Pr[\delta_{i-1} \rightarrow \delta_i]$$

Differential Cryptanalysis (2)



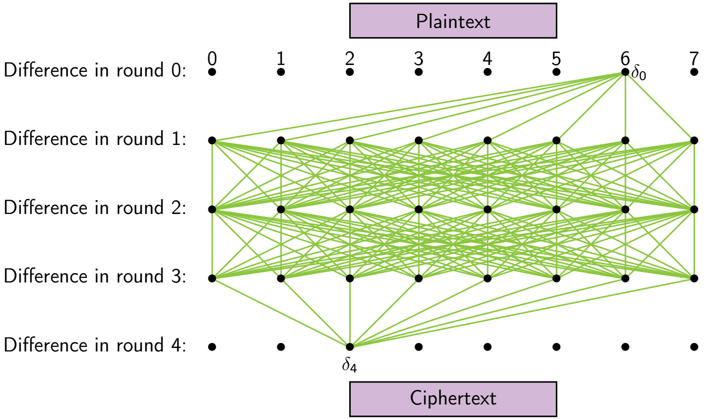
$$\boxed{\Pr[\delta \to \delta'] = \Pr_x[R(x) \oplus R(x \oplus \delta) = \delta']}$$

Differential Cryptanalysis (2)



$$\Pr[\delta_0 \to \delta_1 \to \ldots \to \delta_4] = p_1 \times p_2 \times p_3 \times p_4$$

Differential Cryptanalysis (2)



$$
Pr[\delta_0 \overset{4}{\rightsquigarrow} \delta_4] = \sum_{\delta_1, \delta_2, \delta_3} \prod_{i=1}^{4} \Pr[\delta_{i-1} \to \delta_i]
$$

Differential Cryptanalysis (3)

The transition probabilities can also be written in a matrix A:

$\rightarrow$ For one round:

$$A = \begin{pmatrix} Pr[0 \rightarrow 0] & Pr[0 \rightarrow 1] & \cdots & Pr[0 \rightarrow 2^n - 1] \\ Pr[1 \rightarrow 0] & Pr[1 \rightarrow 1] & \cdots & Pr[1 \rightarrow 2^n - 1] \\ \vdots & \vdots & \ddots & \vdots \\ Pr[2^n - 1 \rightarrow 0] & Pr[2^n - 1 \rightarrow 1] & \cdots & Pr[2^n - 1 \rightarrow 2^n - 1] \end{pmatrix}$$

$\rightarrow$ For r rounds:

$$A^r = \begin{pmatrix} Pr[0 \overset{r}{\rightsquigarrow} 0] & Pr[0 \overset{r}{\rightsquigarrow} 1] & \cdots & Pr[0 \overset{r}{\rightsquigarrow} 2^n - 1] \\ Pr[1 \overset{r}{\rightsquigarrow} 0] & Pr[1 \overset{r}{\rightsquigarrow} 1] & \cdots & Pr[1 \overset{r}{\rightsquigarrow} 2^n - 1] \\ \vdots & \vdots & \ddots & \vdots \\ Pr[2^n - 1 \overset{r}{\rightsquigarrow} 0] & Pr[2^n - 1 \overset{r}{\rightsquigarrow} 1] & \cdots & Pr[2^n - 1 \overset{r}{\rightsquigarrow} 2^n - 1] \end{pmatrix}$$

$\Rightarrow$ Computing A^r is **infeasible for practical ciphers**.

Differential Cryptanalysis

- **Differential distinguisher:**

We collect $D = \mathcal{O}(1/\Pr[\delta \rightsquigarrow \delta'])$ pairs $(P, P \oplus \delta)$ and compute:

$$Q = \#\{P : E(P) \oplus E(P \oplus \delta) = \delta'\}$$

If $\Pr[\delta \rightsquigarrow \delta'] \gg 2^{-n}$, we obtain a distinguisher:

→ $Q \approx D/\Pr[\delta \rightsquigarrow \delta']$ for the cipher

→ $Q \approx D/2^n$ for a random permutation

Table of contents

Linear Cryptanalysis

A **linear approximation** is a a pair of masks (α, α') such that:

$$|\Pr_x[x \cdot \alpha = E_k(x) \cdot \alpha'] - 1/2| \gg 2^{-n/2}$$

for most keys k. [Matsui, EC'93]

Linear Cryptanalysis

A **linear approximation** is a a pair of masks (α, α') such that:

$$|\Pr_x[x \cdot \alpha = E_k(x) \cdot \alpha'] - 1/2| \gg 2^{-n/2}$$

for most keys k. [Matsui, EC'93]

If the cipher is a key-alternating cipher with **independent round keys**:

$$c(\alpha \to \alpha') = 2 \Pr_x[x \cdot \alpha = R(x) \cdot \alpha'] - 1$$

$$c_k(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \sum_{\alpha_1, \alpha_2, \ldots \alpha_{n-1}} (-1)^{\bigoplus_i k_i \cdot \alpha_i} \prod_{i=1}^{r} c(\alpha_{i-1} \to \alpha_i)$$

Linear Cryptanalysis

A **linear approximation** is a a pair of masks (α, α') such that:

$$|\Pr_x[x \cdot \alpha = E_k(x) \cdot \alpha'] - 1/2| \gg 2^{-n/2}$$

for most keys k. [Matsui, EC'93]

If the cipher is a key-alternating cipher with **independent round keys**:

$$c(\alpha \to \alpha') = 2 \Pr_x[x \cdot \alpha = R(x) \cdot \alpha'] - 1$$

$$c_k(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \sum_{\alpha_1, \alpha_2, \ldots \alpha_{n-1}} (-1)^{\bigoplus_i k_i \cdot \alpha_i} \prod_{i=1}^{r} c(\alpha_{i-1} \to \alpha_i)$$

$\to$ When there is a **single dominant trail**, we can approximate the correlation of the linear approximation as the correlation of the trail, up to a change of sign.

$\to$ When there are **several dominant trails**, they can **interact constructively or destructively** depending on the key.

Linear Cryptanalysis

Nyberg defined the **Expected Linear Potential** and showed: [EC'94]

$$\mathrm{ELP}(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \mathrm{Exp}_k(c_k^2(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r))$$
$$= \sum_{\alpha_1,\alpha_2,\ldots\alpha_{r-1}} \prod_{i=1}^{r} c^2(\alpha_{i-1} \to \alpha_i)$$

$\to$ Similarly to the differential case, this can be seen as the computation of the powers of a **matrix** C with coefficients $c^2(\alpha \to \alpha')$.

Linear Cryptanalysis

- **Linear distinguisher:**

We collect $D = \mathcal{O}(1/\operatorname{ELP}[\alpha \rightsquigarrow \alpha'])$ pairs (P, C) and compute:

$$Q = (\#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 0\} - \#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 1\})/D$$

If $ELP[\alpha \rightsquigarrow \alpha'] \gg 2^{-n}$, we obtain a distinguisher:

$\rightarrow$ $Q^2 \approx ELP[\alpha \rightsquigarrow \alpha']$ for the cipher

$\rightarrow$ $Q^2 \approx 2^{-n/2}$ for a random permutation

Differential and Linear Distinguishers

- **Differential distinguisher:**

We collect $D = \mathcal{O}(1/\Pr[\delta \rightsquigarrow \delta'])$ pairs $(P, P \oplus \delta)$ and compute:

$$Q = \#\{P : E(P) \oplus E(P \oplus \delta) = \delta'\}$$

$\rightarrow$ $Q \approx D/\Pr[\delta \rightsquigarrow \delta']$ for the cipher

$\rightarrow$ $Q \approx D/2^n$ for a random permutation

- **Linear distinguisher:**

We collect $D = \mathcal{O}(1/\operatorname{ELP}[\alpha \rightsquigarrow \alpha'])$ pairs (P, C) and compute:

$$Q = (\#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 0\} - \#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 1\})/D$$

$\rightarrow$ $Q^2 \approx ELP[\alpha \rightsquigarrow \alpha']$ for the cipher

$\rightarrow$ $Q^2 \approx 2^{-n/2}$ for a random permutation

Differential and Linear Distinguishers

- **Differential distinguisher:**

We collect $D = \mathcal{O}(1/\Pr[\delta \rightsquigarrow \delta'])$ pairs $(P, P \oplus \delta)$ and compute:

$$Q = \#\{P : E(P) \oplus E(P \oplus \delta) = \delta'\}$$

$\rightarrow$ $Q \approx D/\Pr[\delta \rightsquigarrow \delta']$ for the cipher
$\rightarrow$ $Q \approx D/2^n$ for a random permutation

- **Linear distinguisher:**

We collect $D = \mathcal{O}(1/\operatorname{ELP}[\alpha \rightsquigarrow \alpha'])$ pairs (P, C) and compute:

$$Q = (\#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 0\} - \#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 1\})/D$$

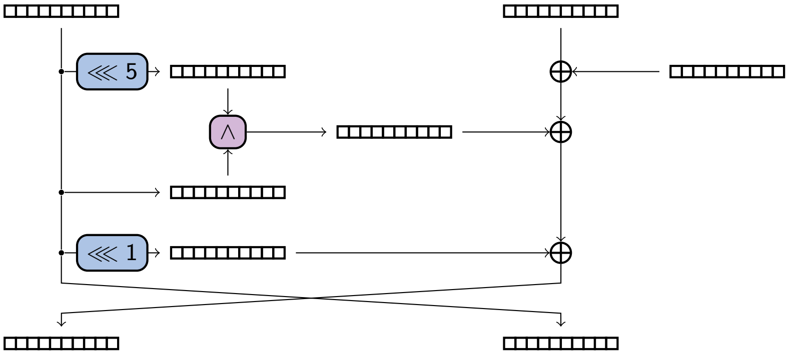
$\rightarrow$ $Q^2 \approx ELP[\alpha \rightsquigarrow \alpha']$ for the cipher
$\rightarrow$ $Q^2 \approx 2^{-n/2}$ for a random permutation

How to find stronger distinguishers for Simon and Simeck?

Table of contents

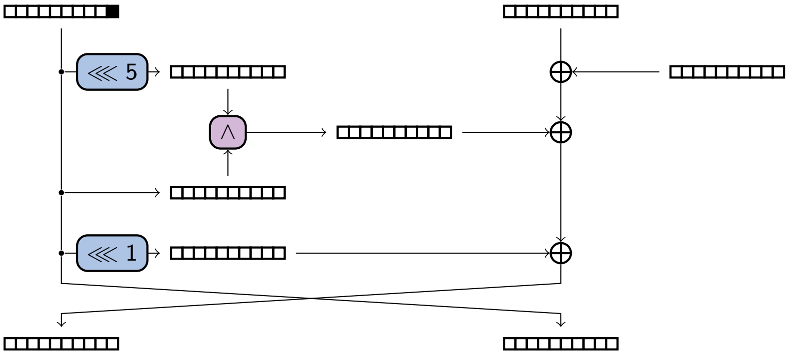
Table of contents

Probability of transition through f



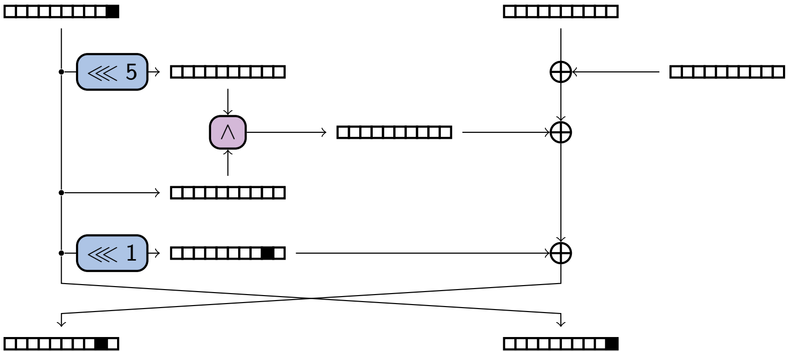
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



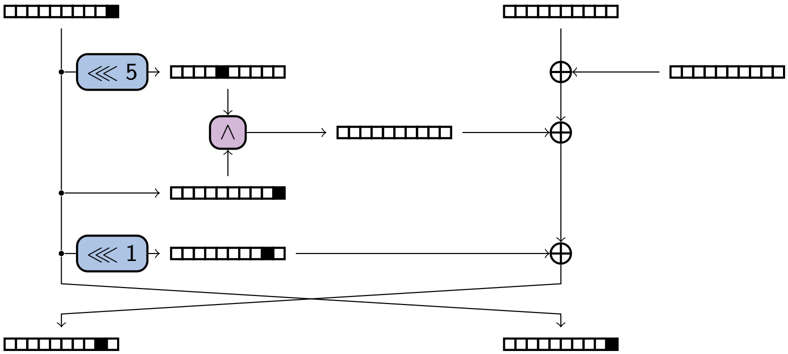
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



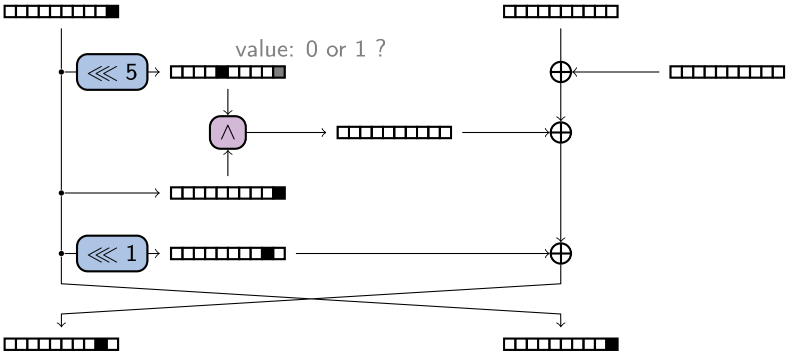
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



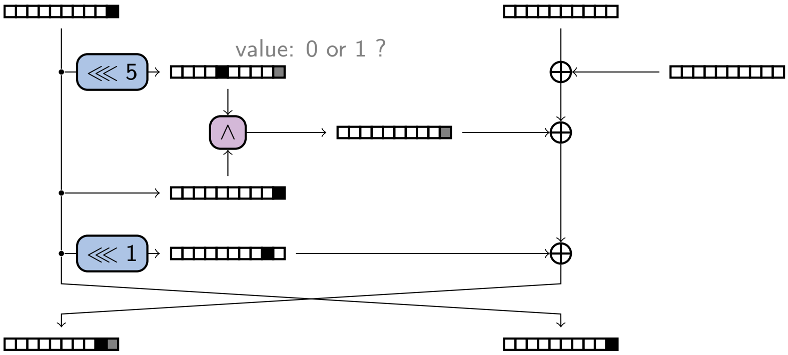
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



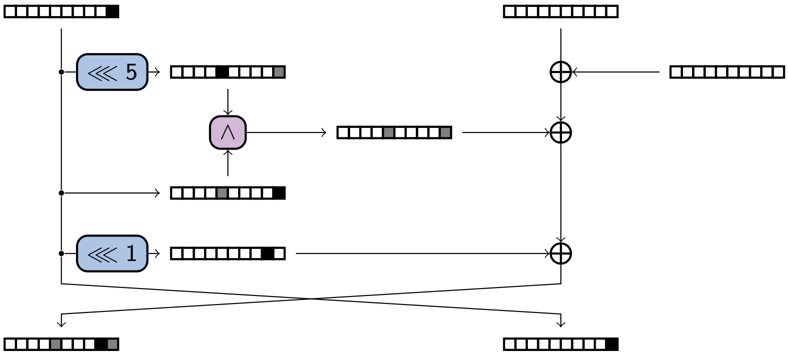
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



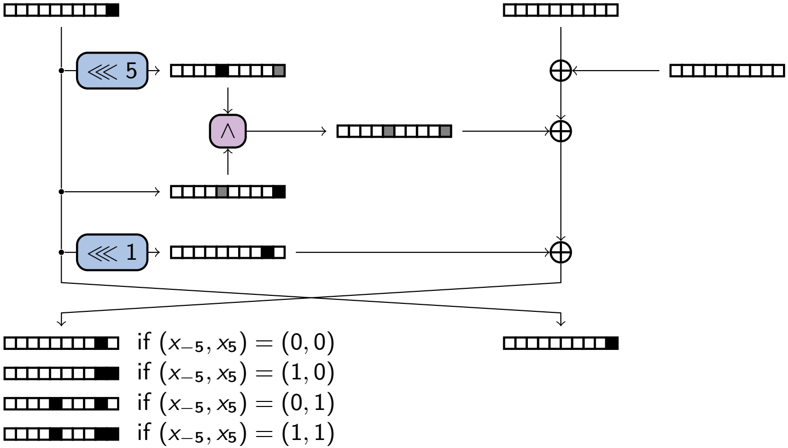
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



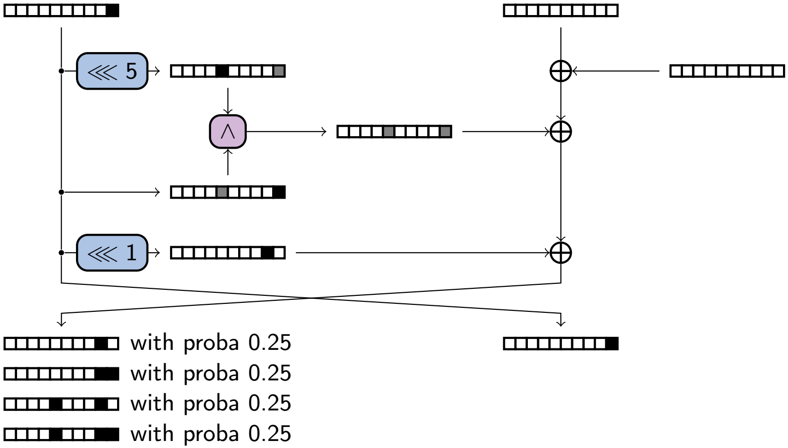
Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



Probability of transition through f

Consider a difference $\alpha = 1$ on the left part:



Probability of transition through f

Since f is **quadratic**...

- $\Rightarrow$ f' is **affine**.
- $\Rightarrow$ all the possible outputs of f' are **equally probable**.
- $\Rightarrow$ all the possible outputs of f' form a **vector space** that can be build efficiently.
- $\Rightarrow$ the **exact probability of transitions** can be computed efficiently for **Simon** and **Simeck**! [KLT, CRYPTO'15]

Probability of transition through f

Kölbl, Leander and Tiessen demonstrated that:

- For a given α, there is an affine space U_α such that

$$\Pr_x[f(\alpha \oplus x) \oplus f(x) = \beta] = \begin{cases} 2^{-\dim(U_\alpha)} & \text{if } \beta \in U_\alpha \\ 0 & \text{otherwise} \end{cases}$$

  U_α is a coset of the image of a linear function:

$$U_\alpha = \operatorname{Img}\big(x \mapsto f(x) \oplus f(x \oplus \alpha) \oplus f(\alpha)\big) \oplus f(\alpha)$$

Probability of transition through f

Kölbl, Leander and Tiessen demonstrated that:

- For a given α, there is an affine space U_α such that

$$\Pr_x[f(\alpha \oplus x) \oplus f(x) = \beta] = \begin{cases} 2^{-\dim(U_\alpha)} & \text{if } \beta \in U_\alpha \\ 0 & \text{otherwise} \end{cases}$$

U_α is a coset of the image of a linear function:

$$U_\alpha = \mathrm{Img}\left(x \mapsto f(x) \oplus f(x \oplus \alpha) \oplus f(\alpha)\right) \oplus f(\alpha)$$

Given the Feistel structure of the round function, we deduce:

$$\Pr[(\delta_L, \delta_R) \to (\delta'_L, \delta'_R)] = \begin{cases} 2^{-\dim(U_{\delta_L})} & \text{if } \delta_L = \delta'_R \text{ and } \delta_R \oplus \delta'_L \in U_{\delta_L} \\ 0 & \text{otherwise} \end{cases}$$

Table of contents

A class of high probability trails

We know how to compute $\Pr[(\delta_L, \delta_R) \to (\delta'_L, \delta'_R)]$ **easily** now...

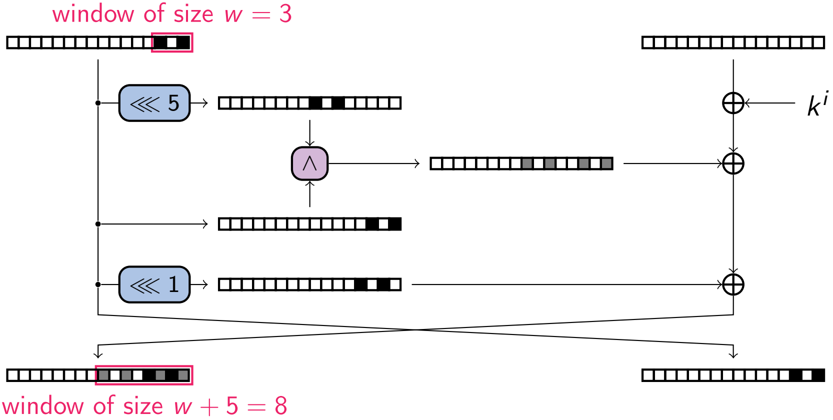
$\to$ But computing $\Pr[(\delta_L, \delta_R) \overset{r}{\rightsquigarrow} (\delta'_L, \delta'_R)]$ remains **hard**!

A class of high probability trails

We know how to compute $\Pr[(\delta_L, \delta_R) \rightarrow (\delta'_L, \delta'_R)]$ **easily** now...

$\rightarrow$ But computing $\Pr[(\delta_L, \delta_R) \overset{r}{\rightsquigarrow} (\delta'_L, \delta'_R)]$ remains **hard**!

Observation: Simeck diffusion in the **worst** case

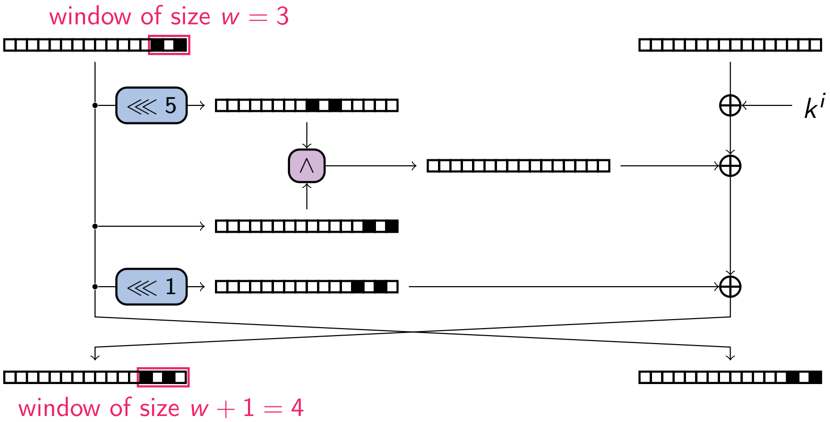


A class of high probability trails

We know how to compute $\Pr[(\delta_L, \delta_R) \to (\delta'_L, \delta'_R)]$ **easily** now...

$\to$ But computing $\Pr[(\delta_L, \delta_R) \overset{r}{\rightsquigarrow} (\delta'_L, \delta'_R)]$ remains **hard**!

Observation: Simeck diffusion in the **best** case

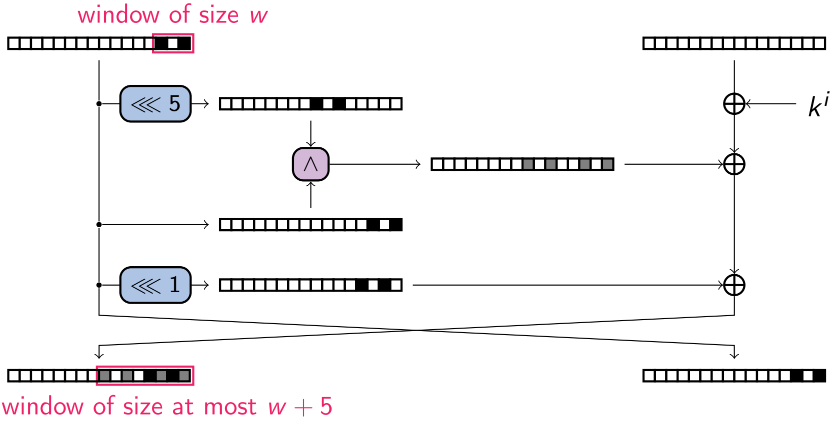


A class of high probability trails

We know how to compute $\Pr[(\delta_L, \delta_R) \to (\delta'_L, \delta'_R)]$ **easily** now...

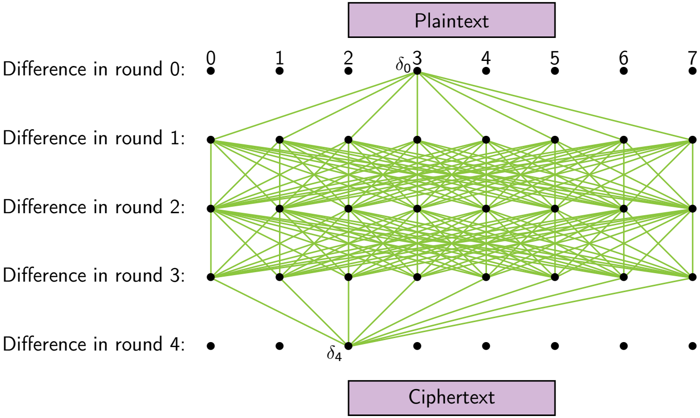
$\to$ But computing $\Pr[(\delta_L, \delta_R) \overset{r}{\rightsquigarrow} (\delta'_L, \delta'_R)]$ remains **hard**!

Conclusion: Simeck has a relatively slow diffusion!



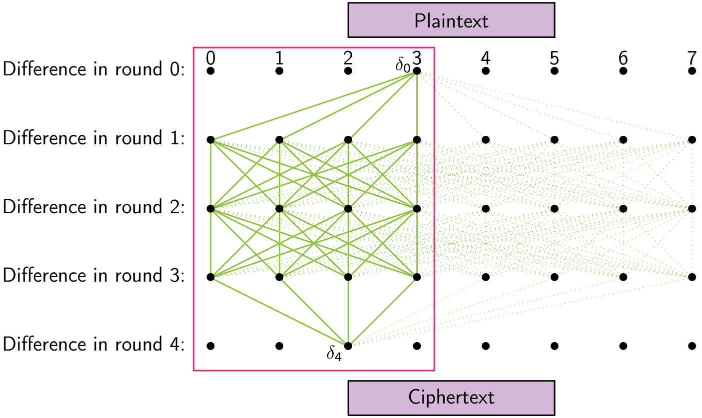
A class of high probability trails

Our idea is to focus on trails that are only active in a window of w bits:



A class of high probability trails

Our idea is to focus on trails that are only active in a window of w bits:



A class of high probability trails

- w: the size of the window ($w \leq n/2$).
- Δ_w: the vector space of differences active only in the w LSBs.
- Δ_w^2: the product $\Delta_w \times \Delta_w$ where the two words are considered.

A class of high probability trails

- w: the size of the window ($w \leq n/2$).
- Δ_w: the vector space of differences active only in the w LSBs.
- Δ_w^2: the product $\Delta_w \times \Delta_w$ where the two words are considered.

A **lower bound** of the probability of the differential (δ_0, δ_r) is computed by summing over all characteristics with intermediate differences in Δ_w^2:

$$\Pr[\delta_0 \underset{w}{\overset{r}{\rightsquigarrow}} \delta_r] = \sum_{\delta_1, \delta_2, \dots \delta_{r-1} \in \Delta_w^2} \prod_{i=1}^{r} \Pr[\delta_{i-1} \to \delta_i] \leq \Pr[\delta_0 \overset{r}{\rightsquigarrow} \delta_r]$$

A class of high probability trails

- w: the size of the window ($w \leq n/2$).
- Δ_w: the vector space of differences active only in the w LSBs.
- Δ_w^2: the product $\Delta_w \times \Delta_w$ where the two words are considered.

A **lower bound** of the probability of the differential (δ_0, δ_r) is computed by summing over all characteristics with intermediate differences in Δ_w^2:

$$\Pr[\delta_0 \underset{w}{\overset{r}{\rightsquigarrow}} \delta_r] = \sum_{\delta_1, \delta_2, \ldots \delta_{r-1} \in \Delta_w^2} \prod_{i=1}^{r} \Pr[\delta_{i-1} \to \delta_i] \leq \Pr[\delta_0 \overset{r}{\rightsquigarrow} \delta_r]$$

$\Rightarrow$ This can be done by computing A_w^r, with A_w the matrix of transitions $\Pr[\delta \to \delta']$ for all $\delta, \delta' \in \Delta_w^2$.

A class of high probability trails

- w: the size of the window ($w \leq n/2$).
- Δ_w: the vector space of differences active only in the w LSBs.
- Δ_w^2: the product $\Delta_w \times \Delta_w$ where the two words are considered.

A **lower bound** of the probability of the differential (δ_0, δ_r) is computed by summing over all characteristics with intermediate differences in Δ_w^2:

$$\Pr[\delta_0 \overset{r}{\underset{w}{\rightsquigarrow}} \delta_r] = \sum_{\delta_1,\delta_2,\ldots\delta_{r-1}\in\Delta_w^2} \prod_{i=1}^{r} \Pr[\delta_{i-1} \to \delta_i] \leq \Pr[\delta_0 \overset{r}{\rightsquigarrow} \delta_r]$$

$\Rightarrow$ This can be done by computing A_w^r, with A_w the matrix of transitions $\Pr[\delta \to \delta']$ for all $\delta, \delta' \in \Delta_w^2$.

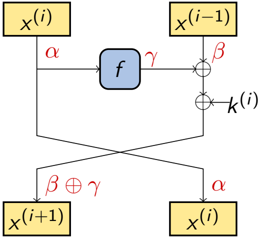
$\Rightarrow$ To reduce the memory requirement, we compute it on the fly!

A class of high probability trails

Algorithm Computation of $\Pr[(\delta_L, \delta_R) \overset{r}{\underset{w}{\rightsquigarrow}} (\delta'_L, \delta'_R)]$

Require: Pre-computation of U_α for all $\alpha \in \Delta_W$.

$X \leftarrow [0$ **for** $i \in \Delta_w^2]$
$X[\delta_L, \delta_R] \leftarrow 1$
for $0 \leq i < r$ **do**
  $Y \leftarrow [0$ **for** $i \in \Delta_w^2]$
  for $\alpha \in \Delta_w$ **do**
    for $\beta \in \Delta_w$ **do**
      for $\gamma \in U_\alpha$ **do**
        $Y[\beta \oplus \gamma, \alpha] = Y[\beta \oplus \gamma, \alpha] + 2^{-\dim(U_\alpha)} X[\alpha, \beta]$
  $X \leftarrow Y$
return $X[\delta'_L, \delta'_R]$

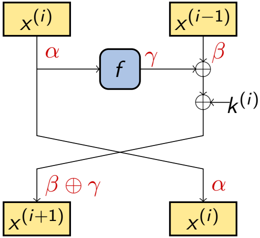


A class of high probability trails

Algorithm Computation of $\Pr[(\delta_L, \delta_R) \underset{w}{\overset{r}{\rightsquigarrow}} (\delta'_L, \delta'_R)]$

Require: Pre-computation of U_α for all $\alpha \in \Delta_W$.

$X \leftarrow [0 \text{ for } i \in \Delta_w^2]$
$X[\delta_L, \delta_R] \leftarrow 1$
for $0 \leq i < r$ **do**
  $Y \leftarrow [0 \text{ for } i \in \Delta_w^2]$
  for $\alpha \in \Delta_w$ **do**
    for $\beta \in \Delta_w$ **do**
      for $\gamma \in U_\alpha$ **do**
        $Y[\beta \oplus \gamma, \alpha] = Y[\beta \oplus \gamma, \alpha] + 2^{-\dim(U_\alpha)} X[\alpha, \beta]$
  $X \leftarrow Y$
return $X[\delta'_L, \delta'_R]$



$\Rightarrow$ This requires $r \times 2^{2w} \times \max_{\alpha \in \Delta_w} |U_\alpha|$ operations, and to store 2^{2w+1} probabilities.

$\Rightarrow$ In practice, for $w = 18$ and $r = 30$, it takes **a week** on a **48-core machine** using 1TB of RAM.

Tighter lower bound for the probability of differentials

Rounds	Differential	Proba (previous)	Reference	Proba (new)
26	$(0, 11) \to (22, 1)$	$2^{-60.02}$	[Kölbl, Roy, 16]	$2^{-54.16}$
26	$(0, 11) \to (2, 1)$	$2^{-60.09}$	[Qin, Chen, Wang, 16]	$2^{-54.16}$
27	$(0, 11) \to (5, 2)$	$2^{-61.49}$	[Liu, Li, Wang, 17]	$2^{-56.06}$
27	$(0, 11) \to (5, 2)$	$2^{-60.75}$	[Huang, Wang, Zhang, 18]	"
28	$(0, 11) \to (\mathtt{A8}, 5)$	$2^{-63.91}$	[Huang, Wang, Zhang, 18]	$2^{-59.16}$

Comparison of our lower bound on the differential probability for Simeck (with $w = 18$), and estimates used in previous attacks.

Differentials with high probabilities

The **best characteristics** we have identified are a set of 64 characteristics:

$$\{(1,2),(1,3),(1,22),(1,23),(2,5),(2,7),(2,45),(2,47)\}$$
$$\rightarrow$$
$$\{(2,1),(3,1),(22,1),(23,1),(5,2),(7,2),(45,2),(47,2)\}$$

$\Rightarrow$ However, $(0,1) \rightarrow (1,0)$ is **almost as good** and will lead to a **more efficient key-recovery** because it has fewer active bits!

Differentials with high probabilities

Computation of the log_2 of the **probability of differentials** for **Simeck**, and the total **number of trails** (using $w = 18$):

	Differential		
Rounds	$(0,1) \rightarrow (1,0)$		$(1,2) \rightarrow (2,1)$
10	$-\infty$		$-\infty$
11	-23.25	(28.0)	-27.25
12	-26.40	(36.2)	-26.17
13	-28.02	(47.2)	-26.90
14	-30.06	(58.2)	-29.59
15	-31.93	(70.8)	-31.37
⋮	⋮	⋮	⋮
20	-41.75	(131.9)	-41.26
⋮	⋮	⋮	⋮
25	-51.01	(192.9)	-50.54
⋮	⋮	⋮	⋮
30	-60.41	(254.0)	-59.92
31	-62.29	(266.2)	-61.81
32	-64.17	(278.4)	-63.69

Differentials with high probabilities

How does our lower bound vary depending on the size of the window w?

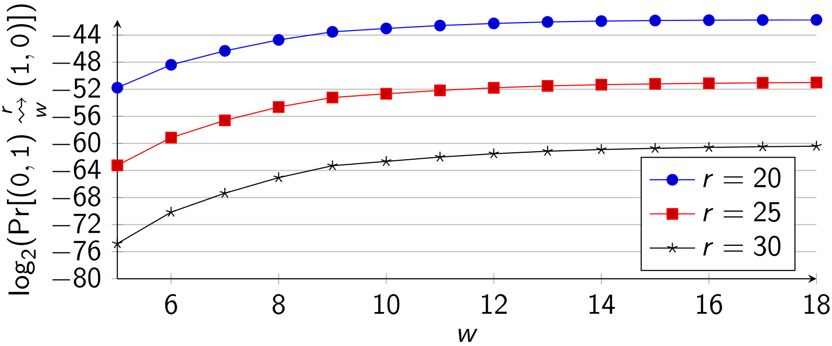


Table of contents

Stronger Linear distinguishers for Simon-like ciphers

We want to compute a **lower bound** of:

$$\mathrm{ELP}(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \sum_{\alpha_1,\alpha_2,\ldots\alpha_{r-1}} \prod_{i=1}^{r} c^2(\alpha_{i-1} \to \alpha_i)$$

Stronger Linear distinguishers for Simon-like ciphers

We want to compute a **lower bound** of:

$$\mathrm{ELP}(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \sum_{\alpha_1,\alpha_2,\ldots\alpha_{r-1}} \prod_{i=1}^{r} c^2(\alpha_{i-1} \to \alpha_i)$$

(1) Since f is **quadratic**, the **exact probability** through one round is:

$$c((\alpha_L, \alpha_R) \to (\alpha'_L, \alpha'_R))^2 = \begin{cases} 2^{-\dim(V_{\alpha_R})} & \text{if } \alpha_R = \alpha'_L \text{ and } \alpha_L \oplus \alpha'_R \in V_{\alpha_R} \\ 0 & \text{otherwise} \end{cases}$$

$$V_\alpha = \mathrm{Img}\left(x \mapsto \big((\alpha \wedge (x \lll a-b)) \oplus ((\alpha \wedge x) \ggg a-b)\big) \ggg b\right) \oplus (\alpha \ggg c)$$

[KLT, CRYPTO'15]

Stronger Linear distinguishers for Simon-like ciphers

We want to compute a **lower bound** of:

$$\mathrm{ELP}(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) = \sum_{\alpha_1,\alpha_2,\dots\alpha_{r-1}} \prod_{i=1}^{r} c^2(\alpha_{i-1} \to \alpha_i)$$

(1) Since f is **quadratic**, the **exact probability** through one round is:

$$c((\alpha_L,\alpha_R) \to (\alpha'_L,\alpha'_R))^2 = \begin{cases} 2^{-\dim(V_{\alpha_R})} & \text{if } \alpha_R = \alpha'_L \text{ and } \alpha_L \oplus \alpha'_R \in V_{\alpha_R} \\ 0 & \text{otherwise} \end{cases}$$

$$V_\alpha = \mathrm{Img}\left(x \mapsto ((\alpha \wedge (x \lll a-b)) \oplus ((\alpha \wedge x) \ggg a-b)) \ggg b\right) \oplus (\alpha \ggg c)$$

[KLT, CRYPTO'15]

(2) Approximation of the ELP using **windows** of w **bits**:

$$\mathrm{ELP}(\alpha_0 \overset{r}{\rightsquigarrow} \alpha_r) \approx \sum_{\alpha_1,\alpha_2,\dots\alpha_{r-1} \in \Delta_w^2} \prod_{i=1}^{r} c^2(\alpha_{i-1} \to \alpha_i)$$

Stronger Linear distinguishers for Simon-like ciphers

A set of 64 (almost) **optimal trails** is obtained:

$$\{(20,40),(22,40),(60,40),(62,40),(50,20),(51,20),(70,20),(71,20)\}$$
$$\rightarrow$$
$$\{(40,20),(40,22),(40,60),(40,62),(20,50),(20,51),(20,70),(20,71)\}$$

Stronger Linear distinguishers for Simon-like ciphers

A set of 64 (almost) **optimal trails** is obtained:

$$\{(20,40),(22,40),(60,40),(62,40),(50,20),(51,20),(70,20),(71,20)\}$$
$$\rightarrow$$
$$\{(40,20),(40,22),(40,60),(40,62),(20,50),(20,51),(20,70),(20,71)\}$$

$\rightarrow$ They are bit-reversed versions of the optimal differential characteristics.

Stronger Linear distinguishers for Simon-like ciphers

A set of 64 (almost) **optimal trails** is obtained:

$$\{(20,40),(22,40),(60,40),(62,40),(50,20),(51,20),(70,20),(71,20)\}$$
$$\rightarrow$$
$$\{(40,20),(40,22),(40,60),(40,62),(20,50),(20,51),(20,70),(20,71)\}$$

$\rightarrow$ They are bit-reversed versions of the optimal differential characteristics.

$\rightarrow$ For key-recovery attack, the preference is given to $(1,0) \rightarrow (0,1)$.

Lower bound of linear and differential distinguishers

Comparison of the **probability** of differentials and the linear potential of linear approximations for Simeck ($\log_2$, using $w = 18$). We also give the total number of trails included in the bound in parenthesis ($\log_2$):

	Differential			Linear		
Rounds	$(0,1) \to (1,0)$		$(1,2) \to (2,1)$	$(1,0) \to (0,1)$		$(1,2) \to (2,1)$
10	$-\infty$		$-\infty$	$-\infty$		$-\infty$
11	−23.25	(28.0)	−27.25	−23.81	(23.9)	−27.81
12	−26.40	(36.2)	−26.17	−26.39	(31.7)	−26.68
13	−28.02	(47.2)	−26.90	−27.98	(42.0)	−27.31
14	−30.06	(58.2)	−29.59	−29.95	(52.5)	−29.56
15	−31.93	(70.8)	−31.37	−31.86	(64.9)	−31.29
⋮	⋮	⋮	⋮	⋮	⋮	⋮
20	−41.75	(131.9)	−41.26	−41.74	(124.5)	−41.25
⋮	⋮	⋮	⋮	⋮	⋮	⋮
25	−51.01	(192.9)	−50.54	−51.00	(184.1)	−50.56
⋮	⋮	⋮	⋮	⋮	⋮	⋮
30	−60.41	(254.0)	−59.92	−60.36	(243.6)	−59.86
31	−62.29	(266.2)	−61.81	−62.24	(255.5)	−61.75
32	−64.17	(278.4)	−63.69	−64.12	(267.4)	−63.63
33	−66.05	(290.6)	−65.57	−66.00	(279.3)	−65.51

Links between Linear and Differential Trails

Alizadeh et al. shown that given a differential trail with probability p:

$$(\alpha_0, \beta_0) \to (\alpha_1, \beta_1) \to \ldots \to (\alpha_r, \beta_r)$$

we can convert it into a linear trail:

$$(\overleftarrow{\beta}_0, \overleftarrow{\alpha}_0) \to (\overleftarrow{\beta}_1, \overleftarrow{\alpha}_1) \to \ldots \to (\overleftarrow{\beta}_r, \overleftarrow{\alpha}_r)$$

where $\overleftarrow{x}$ denotes bit-reversed x.

Links between Linear and Differential Trails

Alizadeh et al. shown that given a differential trail with probability p:

$$(\alpha_0, \beta_0) \to (\alpha_1, \beta_1) \to \ldots \to (\alpha_r, \beta_r)$$

we can convert it into a linear trail:

$$(\overleftarrow{\beta}_0, \overleftarrow{\alpha}_0) \to (\overleftarrow{\beta}_1, \overleftarrow{\alpha}_1) \to \ldots \to (\overleftarrow{\beta}_r, \overleftarrow{\alpha}_r)$$

where $\overleftarrow{x}$ denotes bit-reversed x.

$\rightarrow$ if all the non-linear gates are independent: the linear trail has squared correlation p.

Links between Linear and Differential Trails

Alizadeh et al. shown that given a differential trail with probability p:

$$(\alpha_0, \beta_0) \to (\alpha_1, \beta_1) \to \ldots \to (\alpha_r, \beta_r)$$

we can convert it into a linear trail:

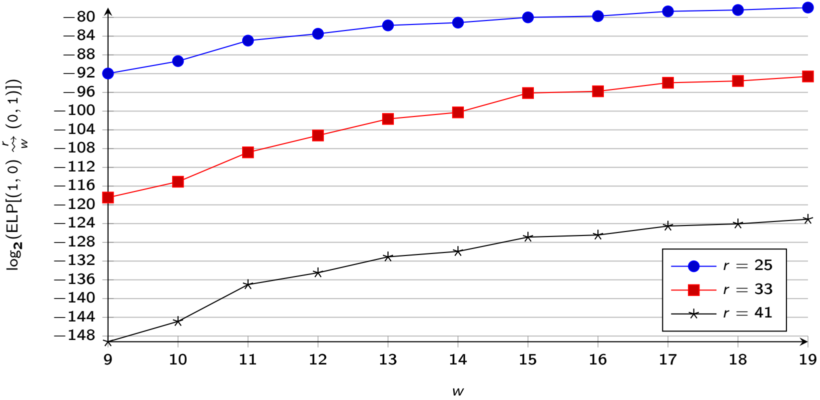
$$(\overleftarrow{\beta}_0, \overleftarrow{\alpha}_0) \to (\overleftarrow{\beta}_1, \overleftarrow{\alpha}_1) \to \ldots \to (\overleftarrow{\beta}_r, \overleftarrow{\alpha}_r)$$

where $\overleftarrow{x}$ denotes bit-reversed x.

- → if all the non-linear gates are independent: the linear trail has squared correlation p.
- → else: the probabilities of the linear and differential trails are not the same, but very similar.

What about Simon?

We also apply the same strategy against **Simon**, but the bound we obtain is **not as tight** as for Simeck: the linear potential still increases significantly with the window size w.



Effect of w on the probability of Simon linear hulls.

Table of contents

Table of contents

Reminder: Differential and Linear Distinguishers

- **Differential distinguisher:**

We collect $D = \mathcal{O}(1/\Pr[\delta \rightsquigarrow \delta'])$ pairs $(P, P \oplus \delta)$ and compute:

$$Q = \#\{P : E(P) \oplus E(P \oplus \delta) = \delta'\}$$

$\rightarrow$ $Q \approx D/\Pr[\delta \rightsquigarrow \delta']$ for the cipher
$\rightarrow$ $Q \approx D/2^n$ for a random permutation

- **Linear distinguisher:**

We collect $D = \mathcal{O}(1/\mathrm{ELP}[\alpha \rightsquigarrow \alpha'])$ pairs (P, C) and compute:

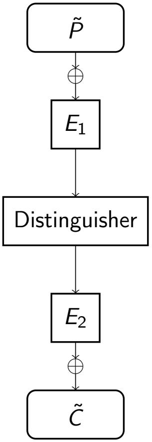
$$Q = (\#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 0\} - \#\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 1\})/D$$

$\rightarrow$ $Q^2 \approx ELP[\alpha \rightsquigarrow \alpha']$ for the cipher
$\rightarrow$ $Q^2 \approx 2^{-n/2}$ for a random permutation

Key Recovery

$\boxed{\text{Distinguisher}}$

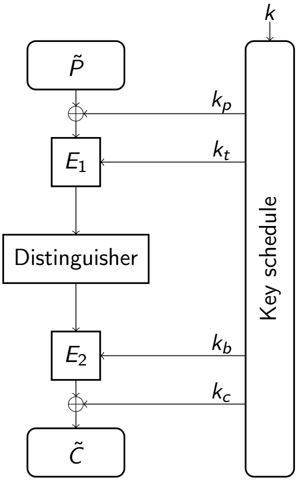
Key Recovery



General description of a cipher.

- Some rounds are added **before** and/or **after** the distinguisher.

Key Recovery



General description of a cipher.

- Some rounds are added **before** and/or **after** the distinguisher.
- The statistic used by the distinguisher is Q, and it can be evaluated using a subset of the key: (k_p, k_t, k_b, k_c).
- The total number of guessed bits is κ_g with $\kappa_g < \kappa$.

Key-recovery

Algorithm Naive key-recovery

for all $k = (k_p, k_t, k_b, k_c)$ **do**
    for all pairs in D **do**
        compute $Q(k)$
    if $Q(k) > s$ **then**
        k is a possible candidate

Complexity: $D \times 2^{\kappa_g}$ with κ_g the number of key bits of k.

Key-recovery

Algorithm Naive key-recovery

for all $k = (k_p, k_t, k_b, k_c)$ **do**
  for all pairs in D **do**
    compute $Q(k)$
  if $Q(k) > s$ **then**
    k is a possible candidate

Complexity: $D \times 2^{\kappa_g}$ with κ_g the number of key bits of k.

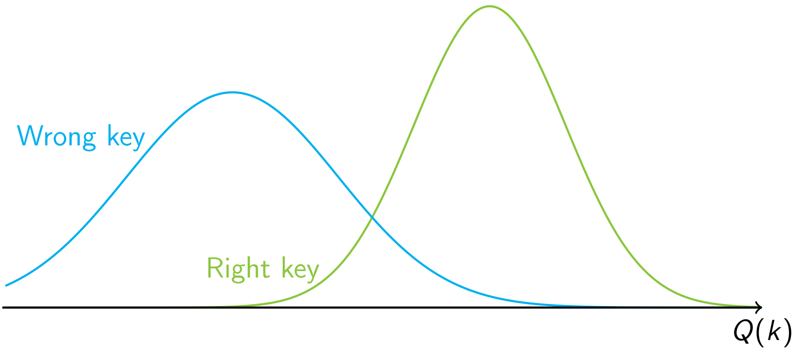
This can be reduced to approximately $D + 2^{\kappa_g}$ using algorithm tricks:

- **Dynamic key guessing** for Differential Cryptanalysis [QHS'16, WWJZ'18]
- **Fast Walsh Transform** for Linear Cryptanalysis [CSQ'07, FN'20]

Key-recovery

F_R: the probability distribution of Q for the **right** key.

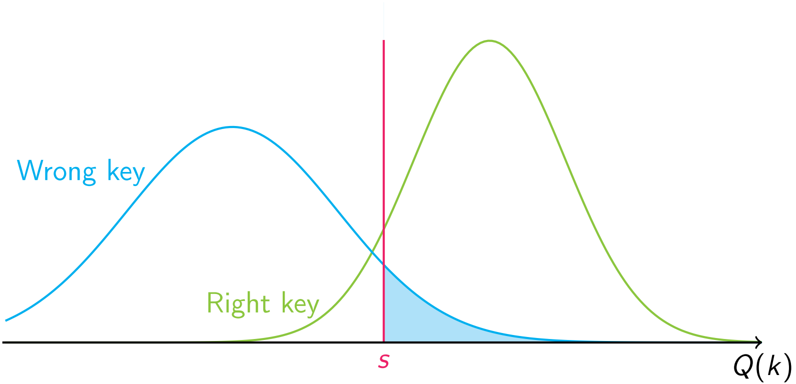
F_W: the probability distribution of Q for a **wrong** key.



Key-recovery

F_R: the probability distribution of Q for the **right** key.

F_W: the probability distribution of Q for a **wrong** key.



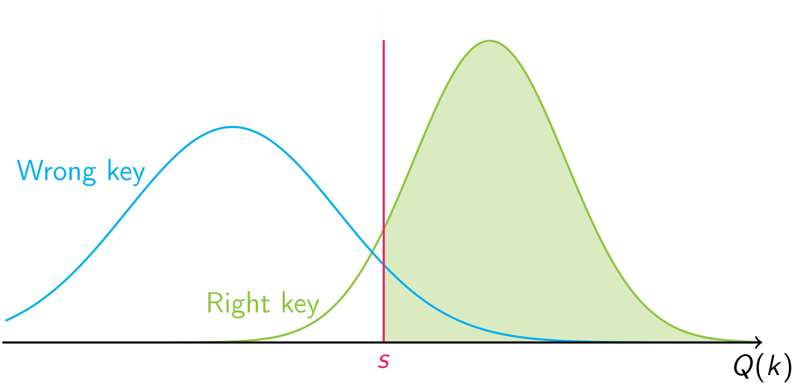
We aim to keep a proportion 2^{-a} of key candidates, so we set a threshold s:

$$2^{-a} = 1 - F_W(s) \quad \Leftrightarrow \quad s = F_w^{-1}(1 - 2^{-a})$$

Key-recovery

F_R: the probability distribution of Q for the **right** key.

F_W: the probability distribution of Q for a **wrong** key.



Then, the **success probability** is given by:

$$P_S = 1 - F_R(s)$$

Table of contents

Key Recovery Using Differential Cryptanalysis

We reuse the **dynamic key-guessing** attack. [QHS'16,WWJZ'18]

(1) Which key bits need to be guessed?

(2) How to rearrange operations to reduce time complexity?

Key Recovery Using Differential Cryptanalysis

We reuse the **dynamic key-guessing** attack. [QHS'16,WWJZ'18]

(1) Which key bits need to be guessed?

Offline part: determining the extended path associated to a differential, and then deducing the subkey bits that need to be guessed.

(2) How to rearrange operations to reduce time complexity?

Key Recovery Using Differential Cryptanalysis

We reuse the **dynamic key-guessing** attack. [QHS'16,WWJZ'18]

(1) Which key bits need to be guessed?

Offline part: determining the extended path associated to a differential, and then deducing the subkey bits that need to be guessed.

(2) How to rearrange operations to reduce time complexity?

Online part: guess subkey bits and filter data round by round, in order to compute $Q(k)$.

Dynamic key guessing: Offline Part

r	Differential path	
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	30-round differential ($3 \to 33$)	
33	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Starting from the differential $(0, 1) \to (1, 0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

Dynamic key guessing: Offline Part

r	Differential path	
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	30-round differential (3 $\rightarrow$ 33)	
33	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Starting from the differential $(0, 1) \rightarrow (1, 0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

Dynamic key guessing: Offline Part

r	Differential path	
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $*$ 0 0 0 1 $*$
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	30-round differential ($3 \to 33$)	
33	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
34	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $*$ 0 0 0 1 $*$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Starting from the differential $(0, 1) \to (1, 0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

Dynamic key guessing: Offline Part

r	Differential path	
0	000000000000000000000*000**001**	0000000000000000*000**00***01***
1	00000000000000000000000000*0001*	000000000000000000000*000**001**
2	00000000000000000000000000000001	00000000000000000000000000*0001*
3	00000000000000000000000000000000	00000000000000000000000000000001
	30-round differential (3 → 33)	
33	00000000000000000000000000000001	00000000000000000000000000000000
34	00000000000000000000000000*0001*	00000000000000000000000000000001
35	000000000000000000000*000**001**	00000000000000000000000000*0001*
36	0000000000000000*000**00***01***	000000000000000000000*000**001**
37	00000000000*000**00***0****1****	0000000000000000*000**00***01***
38	000000*000**00***0**************	00000000000*000**00***0****1****
39	0*000**00***0*******************	000000*000**00***0**************
40	**00***0************************	0*000**00***0*******************

Starting from the differential $(0, 1) \to (1, 0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

Dynamic key guessing: Offline Part

r	Differential path	
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **0** 0 0 0 **0** **0** 0 0 0 **0** **1**	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	30-round differential (3 $\rightarrow$ 33)	
33	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
34	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
35	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *
36	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *
37	0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * 1 * * * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *
38	0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * *	0 **0** 0 0 0 **0** **0** 0 0 **0** **0** * 0 **0** **0** * * **0** **0** * * * **0** * * * * **1** * * * *
39	0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * * * * * * *	0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * *
40	* * 0 0 * * * 0 *	0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * * * * * * *

Starting from the differential $(0,1) \rightarrow (1,0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

(2) Determining the sufficient bit conditions (in red).

Dynamic key guessing: Offline Part

r	Differential path	
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	30-round differential (3 → 33)	
33	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
34	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
35	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 1 *
36	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 1 * *
37	0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * 1 * * * *	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 1 * * *
38	0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * *	0 0 0 0 0 0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * 1 * * * *
39	0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * * * * * * *	0 0 0 0 0 0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * *
40	* * 0 0 * * * 0 *	0 * 0 0 0 * * 0 0 * * * 0 * * * * * * * * * * * * * * * * * * *

Starting from the differential $(0, 1) \rightarrow (1, 0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

(2) Determining the sufficient bit conditions (in red).

Dynamic key guessing: Offline Part

r	Differential path	
0	$000000000000000000000\ast 000\ast\ast 001\ast\ast$	$0000000000000000\ast 000\ast\ast 00\ast\ast\ast 01\ast\ast\ast$
1	$00000000000000000000000000\ast 0001\ast$	$000000000000000000000\ast 000\ast\ast 001\ast\ast$
2	$000000000000000000000\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{1}$	$00000000000000000000000000\ast 0001\ast$
3	$00000000000000000000000000\textcolor{red}{0}0000\textcolor{red}{0}$	$00000000000000000000000000000001$
	30-round differential ($3 \to 33$)	
33	$00000000000000000000000000000001$	$00000000000000000000000000\textcolor{red}{0}0000\textcolor{red}{0}$
34	$00000000000000000000000000\ast 0001\ast$	$000000000000000000000\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{1}$
35	$000000000000000000000\ast 000\ast\ast 001\ast\ast$	$0000000000000000\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}00\textcolor{red}{0}\textcolor{red}{0}\ast 00\textcolor{red}{0}\textcolor{red}{1}\ast$
36	$0000000000000000\ast 000\ast\ast 00\ast\ast\ast 01\ast\ast\ast$	$00000000000\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}00\textcolor{red}{0}\textcolor{red}{0}\ast 0\textcolor{red}{0}\textcolor{red}{0}\ast\ast 0\textcolor{red}{0}\textcolor{red}{1}\ast\ast$
37	$00000000000\ast 000\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast 1\ast\ast\ast\ast$	$000000\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}00\textcolor{red}{0}\textcolor{red}{0}\ast 0\textcolor{red}{0}\textcolor{red}{0}\ast\ast\textcolor{red}{0}\textcolor{red}{0}\ast\ast\ast\textcolor{red}{0}\textcolor{red}{1}\ast\ast\ast$
38	$000000\ast 000\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast$	$0\textcolor{red}{0}000\textcolor{red}{0}\textcolor{red}{0}00\textcolor{red}{0}\textcolor{red}{0}\ast 0\textcolor{red}{0}\textcolor{red}{0}\ast\ast\textcolor{red}{0}\textcolor{red}{0}\ast\ast\ast\textcolor{red}{0}\ast\ast\ast\ast\textcolor{red}{1}\ast\ast\ast\ast$
39	$0\ast 000\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast$	$000000\ast 000\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast$
40	$\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast$	$0\ast 000\ast\ast 00\ast\ast\ast 0\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast$

Starting from the differential $(0,1) \to (1,0)$ covering **30 rounds**, we add **3 rounds before**, and **7 rounds after**:

(1) Tracking the propagation of differences in the additional rounds.

(2) Determining the sufficient bit conditions (in red).

(3) Deducing the necessary bits to check the sufficient bit conditions:

$$(k_p, k_t, k_b, k_c)$$

Dynamic key guessing: Online Part (1)

Round by round, we **guess** subkey bits and **filter** the pairs that do not check the sufficient bit conditions.

At the end, for each key guess (k_p, k_t, k_b, k_c), we compute $Q(k)$ the number of pairs satisfying the differential:

$\rightarrow$ for the **right** key guess, the expected value is $\lambda_R = p \times D/2$.

$\rightarrow$ for the **wrong** key guess, the expected value is $\lambda_W = D/2^{n-1}$.

$\Rightarrow$ F_R and F_W are **Poisson law** with parameter λ_R and λ_W.

Dynamic key guessing: Online Part (2)

Then, for all key guess k such that $Q(k) > s$, the corresponding master keys are reconstructed:

- If the key schedule is **linear**: this can be done using linear algebra and an exhaustive search of the $\kappa - \kappa_g$ missing bits of the key.
- If the key schedule is **non-linear**: combining information from the top and the bottom part of the key is not immediate. Starting from the $\kappa_{max} = \max\left(\kappa_p + \kappa_t, \kappa_b + \kappa_c\right)$ bits, we do an exhaustive search of the $\kappa - \kappa_{max}$ missing bits.

Dynamic key guessing – Complexity

In total, the complexity and the probability of success are:

$$C_1 = D + 2^{\kappa_g} \cdot \lambda_W + 2^{\kappa + \kappa_{\min}} \cdot (1 - F_W(s))$$

$$P_S = 1 - F_R(s)$$

with $\kappa_{min} = \min\left(\kappa_p + \kappa_t, \kappa_b + \kappa_c\right)$.

Dynamic key guessing – Complexity

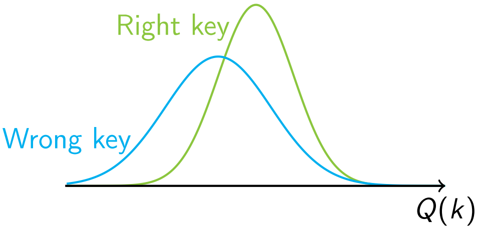
In total, the complexity and the probability of success are:

$$C_1 = D + 2^{\kappa_g} \cdot \lambda_W + 2^{\kappa + \kappa_{\min}} \cdot (1 - F_W(s))$$

$$P_S = 1 - F_R(s)$$

with $\kappa_{min} = \min\left(\kappa_p + \kappa_t, \kappa_b + \kappa_c\right)$.

$\Rightarrow$ The attack is repeated until it succeeds, using rotations of the initial differential: $C = C_1/P_S$.



Dynamic key guessing – Complexity

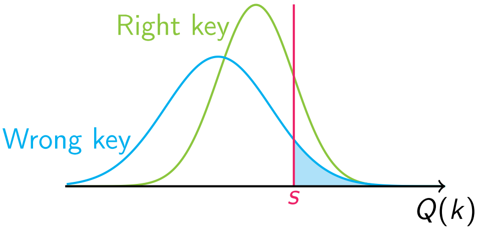
In total, the complexity and the probability of success are:

$$C_1 = D + 2^{\kappa_g} \cdot \lambda_W + 2^{\kappa+\kappa_{\min}} \cdot (1 - F_W(s))$$

$$P_S = 1 - F_R(s)$$

with $\kappa_{min} = \min\left(\kappa_p + \kappa_t, \kappa_b + \kappa_c\right)$.

$\Rightarrow$ The attack is repeated until it succeeds, using rotations of the initial differential: $C = C_1/P_S$.



Dynamic key guessing – Complexity

In total, the complexity and the probability of success are:

$$C_1 = D + 2^{\kappa_g} \cdot \lambda_W + 2^{\kappa+\kappa_{\min}} \cdot (1 - F_W(s))$$

$$P_S = 1 - F_R(s)$$

with $\kappa_{min} = \min\left(\kappa_p + \kappa_t, \kappa_b + \kappa_c\right)$.

$\Rightarrow$ The attack is repeated until it succeeds, using rotations of the initial differential: $C = C_1/P_S$.

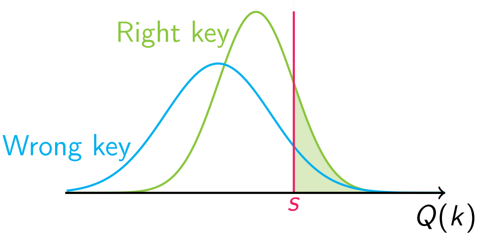


Table of contents

Key-recovery using Linear Cryptanalysis – FWT

We apply the **Fast Walsh Transform** approach proposed by [CSQ'07]:

$$
\begin{aligned}
q(k_p, k_t, k_c, k_b) &= \frac{1}{D}\left(\#\{P, C : P' \cdot \alpha = C' \cdot \beta\} - \#\{P, C : P' \cdot \alpha \neq C' \cdot \beta\}\right) \\
&= \frac{1}{D} \sum_{P,C} (-1)^{P' \cdot \alpha \oplus C' \cdot \beta}
\end{aligned}
$$

Key-recovery using Linear Cryptanalysis – FWT

We apply the **Fast Walsh Transform** approach proposed by [CSQ'07]:

$$\begin{aligned} q(k_p, k_t, k_c, k_b) &= \frac{1}{D}\big(\#\{P, C : P' \cdot \alpha = C' \cdot \beta\} - \#\{P, C : P' \cdot \alpha \neq C' \cdot \beta\}\big) \\ &= \frac{1}{D}\sum_{P,C}(-1)^{P'\cdot\alpha \oplus C'\cdot\beta} \end{aligned}$$

Let define $P' \cdot \alpha = f(k_t, k_p \oplus \chi_p(P))$ and $C' \cdot \beta = g(k_b, k_c \oplus \chi_c(C))$

$$\begin{aligned} &= \frac{1}{D}\sum_{P,C}(-1)^{f(k_t,k_p\oplus\chi_p(P))\oplus g(k_b,k_c\oplus\chi_c(C))} \\ &= \frac{1}{D}\sum_{i\in\mathbb{F}_2^{\kappa_p}}\sum_{j\in\mathbb{F}_2^{\kappa_c}} \#\{P, C : \chi_p(P) = i, \chi_c(C) = j\} \times (-1)^{f(k_t,k_p\oplus i)\oplus g(k_b,k_c\oplus j)} \end{aligned}$$

Key-recovery using Linear Cryptanalysis – FWT

We apply the **Fast Walsh Transform** approach proposed by [CSQ'07]:

$$\begin{aligned} q(k_p, k_t, k_c, k_b) &= \frac{1}{D}\left(\#\{P, C : P' \cdot \alpha = C' \cdot \beta\} - \#\{P, C : P' \cdot \alpha \neq C' \cdot \beta\}\right) \\ &= \frac{1}{D}\sum_{P,C}(-1)^{P'\cdot\alpha \oplus C'\cdot\beta} \end{aligned}$$

Let define $P' \cdot \alpha = f(k_t, k_p \oplus \chi_p(P))$ and $C' \cdot \beta = g(k_b, k_c \oplus \chi_c(C))$

$$= \frac{1}{D}\sum_{P,C}(-1)^{f(k_t, k_p \oplus \chi_p(P)) \oplus g(k_b, k_c \oplus \chi_c(C))}$$

$$= \frac{1}{D}\sum_{i \in \mathbb{F}_2^{\kappa_p}} \sum_{j \in \mathbb{F}_2^{\kappa_c}} \#\{P, C : \chi_p(P) = i, \chi_c(C) = j\} \times (-1)^{f(k_t, k_p \oplus i) \oplus g(k_b, k_c \oplus j)}$$

We remark that the previous expression is actually a convolution:

$$= \frac{1}{D}\sum_{i,j} \phi(i,j) \times \psi_{k_t,k_b}(k_p \oplus i, k_c \oplus j) = \frac{1}{D}(\phi * \psi_{k_t,k_b})(k_p, k_c),$$

with $$\begin{cases} \phi(x,y) & = \#\{P, C : \chi_p(P) = x, \chi_c(C) = y\} \\ \psi_{k_t,k_b}(x,y) & = (-1)^{f(k_t,x) \oplus g(k_b,y)} \end{cases}$$

Key-recovery using Linear Cryptanalysis – FWT

We apply the **Fast Walsh Transform** approach proposed in [CSQ'07] and improved in [FN'20] to Simeck and Simon.
The attack is decomposed in three phases:

Distillation phase. Compute $\phi(x, y) = \#\{P, C : \chi_p(P) = x, \chi_c(C) = y\}$ for $0 \leq x < 2^{\kappa_p}$, $0 \leq y < 2^{\kappa_c}$.

Analysis phase. For each guess of k_t, k_b, for all $0 \leq x < 2^{\kappa_p}$, $0 \leq y < 2^{\kappa_c}$, compute $\psi_{k_t,k_b}(x, y) = (-1)^{f(k_t,x) \oplus g(k_b,y)}$, then evaluate the convolution $\phi * \psi_{k_t,k_b}$ using the **Fast Walsh Transform**.

Search phase. For all keys with $q(k_p, k_t, k_c, k_b) \geq s$, exhaustively try all master keys corresponding to k_p, k_t, k_c, k_b.

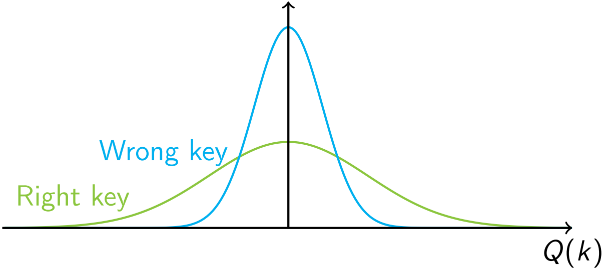
Key-recovery using Linear Cryptanalysis – FWT

How to estimate the **Success Probability** when they are **several dominant trails**?

As seen previously, they can interact **constructively**, or **destructively**...
But the correlation for the **right** and the **wrong** key follow **normal distribution** with parameters: [BN, ToSC'16]

$$\mu_R = 0 \qquad \sigma_R^2 = B/D + \mathrm{ELP}$$

$$\mu_W = 0 \qquad \sigma_W^2 = B/D + 2^{-n}$$

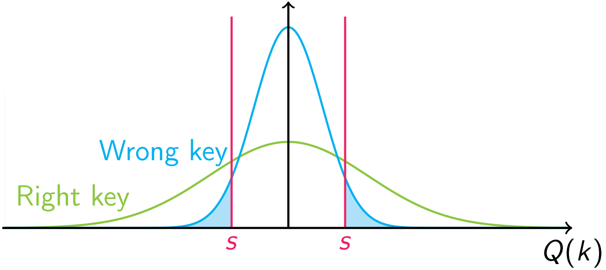


Key-recovery using Linear Cryptanalysis – FWT

How to estimate the **Success Probability** when they are **several dominant trails**?

As seen previously, they can interact **constructively**, or **destructively**...
But the correlation for the **right** and the **wrong** key follow **normal distribution** with parameters: [BN, ToSC'16]

$$\mu_R = 0 \qquad \sigma_R^2 = B/D + \mathrm{ELP}$$
$$\mu_W = 0 \qquad \sigma_W^2 = B/D + 2^{-n}$$

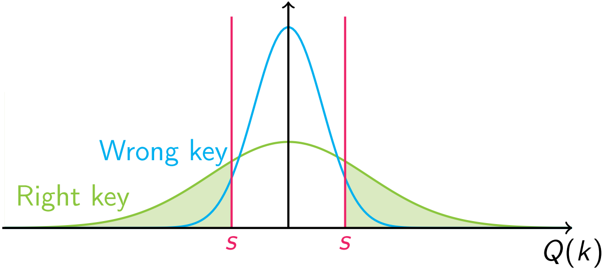


Key-recovery using Linear Cryptanalysis – FWT

How to estimate the **Success Probability** when they are **several dominant trails**?

As seen previously, they can interact **constructively**, or **destructively**...
But the correlation for the **right** and the **wrong** key follow **normal distribution** with parameters: [BN, ToSC'16]

$$\mu_R = 0 \qquad \sigma_R^2 = B/D + \mathrm{ELP}$$
$$\mu_W = 0 \qquad \sigma_W^2 = B/D + 2^{-n}$$



Linear VS Differential Key-recovery

We have seen previously that **linear** and **differential distinguishers** are very **close**...

Linear VS Differential Key-recovery

We have seen previously that **linear** and **differential distinguishers** are very **close**...

But what about the **key-recovery** part?

Linear VS Differential Key-recovery

We have seen previously that **linear** and **differential distinguishers** are very **close**...

But what about the **key-recovery** part?

The main difference come from the **number of bits** that have to be **guessed**:

Key bits	Differential		Linear	
Rounds	total	independent	total	independent
1	0	0	0	0
2	2	2	2	2
3	9	9	7	7
4	27	27	16	16
5	56	56	30	30
6	88	88	50	48
7	120	114	75	68
8			104	88

Comparison of key recovery rounds for differential and linear attacks against Simeck64/128.

Key-Recovery Parameters

Examples of set of parameters for Simeck64/128:

- **Differential cryptanalysis**:

$$Rounds = 40 = 3 + 30 + 7 \quad D = 2^{64}$$

$$\kappa_{min} = 9 \quad \kappa_{max} = 114 \quad \lambda_R = 2^{2.59} \quad \lambda_W = 2^{-1} \quad s = 6$$

$$\Rightarrow C_1 = 2^{122} \quad P_S = 0.4 \quad C = 2^{123.4}$$

- **Linear cryptanalysis**:

$$Rounds = 42 = 8 + 30 + 4 \quad D = 2^{64}$$

$$\kappa_{min} = 16 \quad \kappa_{max} = 88 \quad a = 29$$

$$\Rightarrow C_1 = 2^{118} \quad P_S = 0.1 \quad C = 2^{121.5}$$

Table of contents

Results on Simeck

Cipher	Rounds	Attacked	Data	Time	Ref	Note
Simeck48/96	36	30	$2^{47.66}$	$2^{88.04}$	[QCW'16]	Linear † ‡
		32	2^{47}	$2^{90.9}$	**New**	Linear
Simeck64/128	44	37	$2^{63.09}$	$2^{121.25}$	[QCW'16]	Linear † ‡
		42	$2^{63.5}$	$2^{123.9}$	**New**	Linear

Summary of previous and new attacks against Simeck.

† The advantage is too low to do a key-recovery.

‡ Attack use the duality between linear and differential distinguishers.

Results on Simon

Cipher	Rounds	Attacked	Data	Time	Ref	Note
Simon96/96	52	37	2^{95}	$2^{87.2}$	[WWJZ'18]	Diff.
		43	2^{94}	$2^{89.6}$	**New**	Linear
Simon96/144	54	38	$2^{95.2}$	2^{136}	[CW'16]	Linear
		45	2^{95}	$2^{136.5}$	[CW'16]	Linear
Simon128/128	68	50	2^{127}	$2^{119.2}$	[WWJZ'18]	Diff.
		53	2^{127}	2^{121}	**New**	Linear
Simon128/192	69	51	2^{127}	$2^{183.2}$	[WWJZ'18]	Diff.
		55	2^{127}	$2^{185.2}$	**New**	Linear
Simon128/256	72	53	$2^{127.6}$	2^{249}	[CW'16]	Linear
		56	2^{126}	2^{249}	**New**	Linear

Summary of previous and new attacks against Simon.

Results on Simon

We show that **Simon96/96** and **Simon96/144** only have **17%** of the **rounds as security margin**, which **contradicts** what the designers wrote:

Assumption [Simon designers, ePrint2017/560]

"After almost 4 years of concerted effort by academic researchers, the various versions of Simon and Speck retain a margin averaging around 30%, and ***in every case over 25%****. The design team's analysis when making stepping decisions was consistent with these numbers."*

Conclusion

- Using differential and linear paths with **all intermediate states in a fixed window of** w **bits**, we obtain better probabilities for existing differential and linear distinguishers.

Conclusion

- Using differential and linear paths with **all intermediate states in a fixed window of** w **bits**, we obtain better probabilities for existing differential and linear distinguishers.
- We also obtain good differential and linear approximation with the **minimum number of active bits**, so that the key-recovery part is also improved.

Conclusion

- Using differential and linear paths with **all intermediate states in a fixed window of** w **bits**, we obtain better probabilities for existing differential and linear distinguishers.
- We also obtain good differential and linear approximation with the **minimum number of active bits**, so that the key-recovery part is also improved.
- By applying this to advanced existing linear and differential attacks, we improved previous results and obtain an attack on **42 out of 44 rounds for Simeck64/128**, and **43 out of 52 rounds of Simon96/96**...

Conclusion

- Using differential and linear paths with **all intermediate states in a fixed window of** w **bits**, we obtain better probabilities for existing differential and linear distinguishers.
- We also obtain good differential and linear approximation with the **minimum number of active bits**, so that the key-recovery part is also improved.
- By applying this to advanced existing linear and differential attacks, we improved previous results and obtain an attack on **42 out of 44 rounds for Simeck64/128**, and **43 out of 52 rounds of Simon96/96**...
- Concerning Simon, the lower bound of the probability of the linear approximation seems not as tight as for Simeck: **further work** can probably improve our results...

Conclusion

- Using differential and linear paths with **all intermediate states in a fixed window of** w **bits**, we obtain better probabilities for existing differential and linear distinguishers.
- We also obtain good differential and linear approximation with the **minimum number of active bits**, so that the key-recovery part is also improved.
- By applying this to advanced existing linear and differential attacks, we improved previous results and obtain an attack on **42 out of 44 rounds for Simeck64/128**, and **43 out of 52 rounds of Simon96/96**...
- Concerning Simon, the lower bound of the probability of the linear approximation seems not as tight as for Simeck: **further work** can probably improve our results...

For more details:

`https://eprint.iacr.org/2021/1198`