Eurocrypt 2025

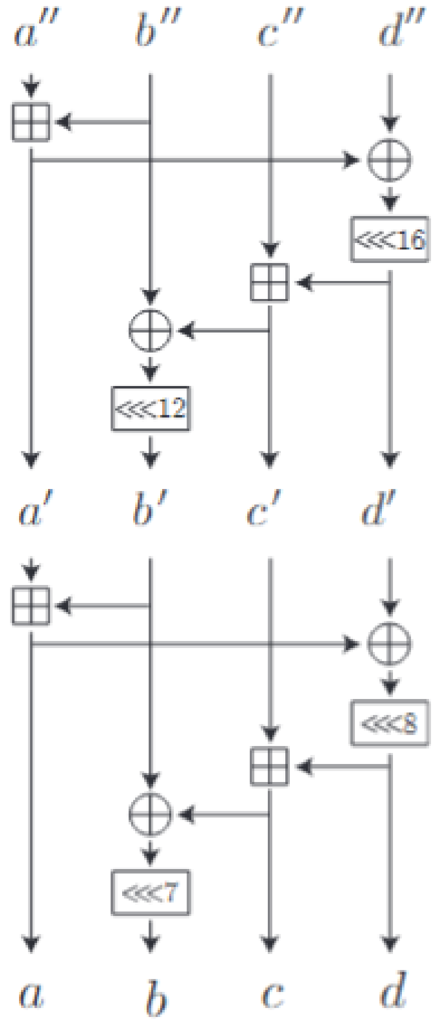
Improved Cryptanalysis of ChaCha: Beating PNBs with Bit Puncturing

Antonio Flórez-Gutiérrez (NTT Social Informatics Laboratories)

Yosuke Todo (NTT Social Informatics Laboratories)

Overview

- ChaCha (since 2008)
  - ARX, one of the most deployed stream ciphers.
- PNBs: Probabilistic Neural Bits (AFK+08, FSE)
  - **Experimental** approximation of the key-recovery map.
- Motivation
  - Theoretical in-depth analysis didn't achieve better results than the black-box experimental analysis (PNBs).
  - Apply **puncturing** (FT24, Eurocrypt), which provides **theoretically optimal** approximation.
  - A new tool, **trail enumeration puncturing,** beats PNBs.



$QR(a'', b'', c'', d'')$

Summary of Results

Round	Data	Time	Note	Ref
6	2^73.7	2^75.5	PNBs w/ syncopation	Wang et al., CRYPTO, 2023
	2^41.6	2^71.0	PNBs w/ linear decomposition	Dey, IEEE-IT, 2024
	2^51.0	**2^61.4**		Ours
	2^55.7	**2^57.4**		Ours
7	2^102.6	2^189.7	DL hull and PNBs	Xu et al., ToSC, 2024
	2^127.7	**2^148.2**		Ours
	2^102.9	**2^154.2**		Ours
7.5	2^32.6	2^255.2	PNBs w/ linear decomposition	Dey, IEEE-IT, 2024
	2^127.1	**2^250.2**		Ours

Review of the Existing Works. What is difficult? What is problem?

ChaCha

$$V^0 = \begin{pmatrix} v_0^0 & v_1^0 & v_2^0 & v_3^0 \\ v_4^0 & v_5^0 & v_6^0 & v_7^0 \\ v_8^0 & v_9^0 & v_{10}^0 & v_{11}^0 \\ v_{12}^0 & v_{13}^0 & v_{14}^0 & v_{15}^0 \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & t_1 & t_2 & t_3 \end{pmatrix}.$$

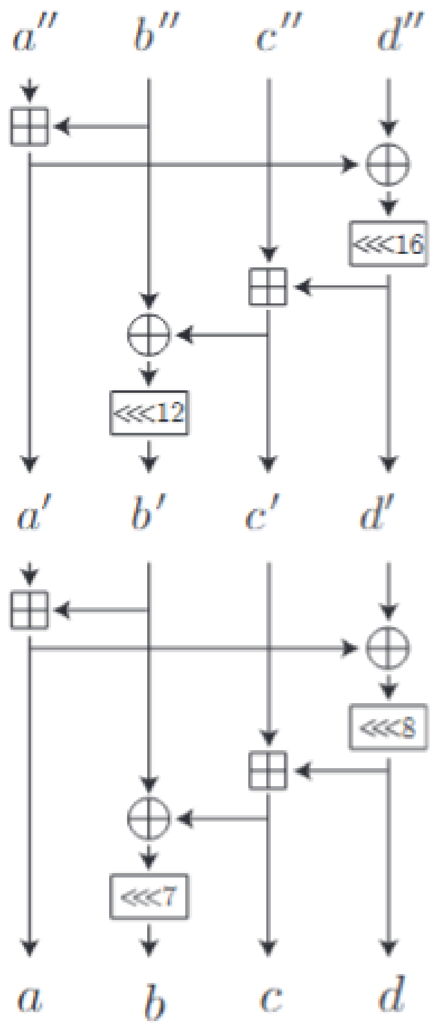
Odd rounds

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}.$$

Even rounds

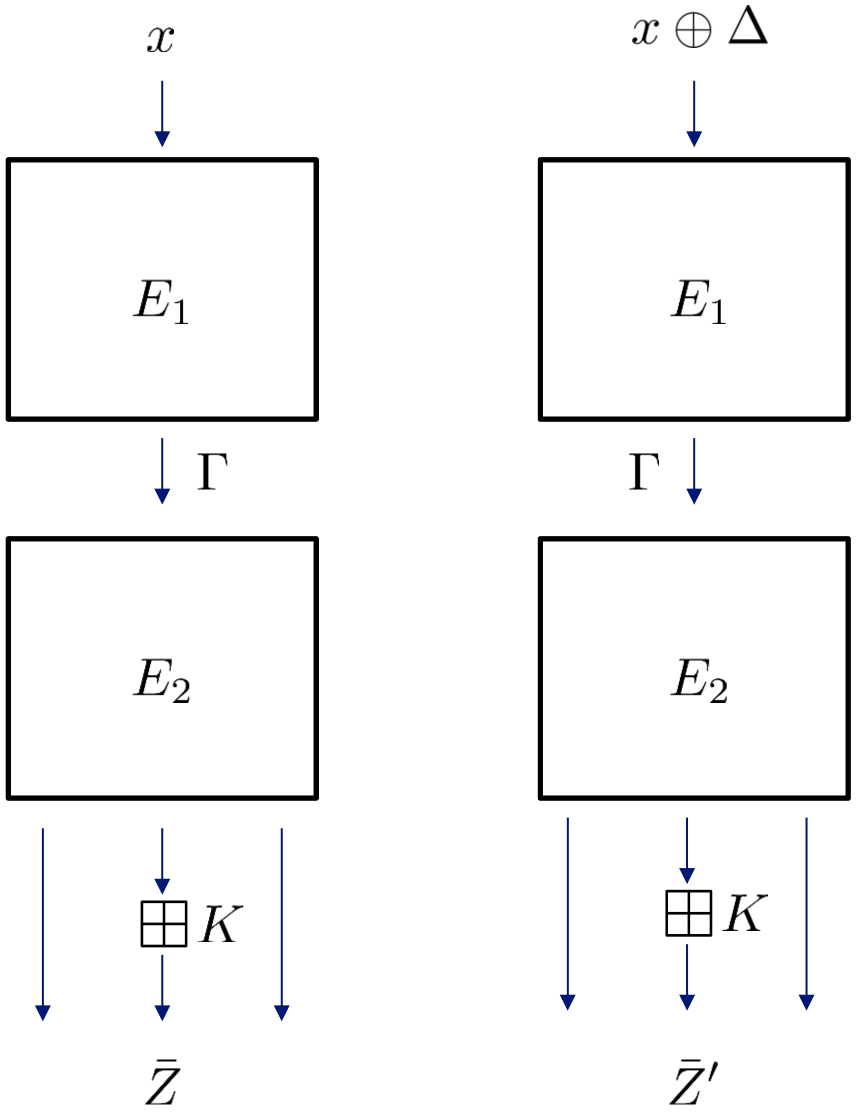
$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}.$$

$$KS = V^0 + V^R = \begin{pmatrix} c_0 + v_0^R & c_1 + v_1^R & c_2 + v_2^R & c_3 + v_3^R \\ k_0 + v_4^R & k_1 + v_5^R & k_2 + v_6^R & k_3 + v_7^R \\ k_4 + v_8^R & k_5 + v_9^R & k_6 + v_{10}^R & k_7 + v_{11}^R \\ t_0 + v_{12}^R & t_1 + v_{13}^R & t_2 + v_{14}^R & t_3 + v_{15}^R \end{pmatrix}$$



$QR(a'', b'', c'', d'')$

Differential-linear attack



Differential-linear distinguisher (aka autocorrelation)

$$\mathrm{Aut}_{E_1}(\Delta,\Gamma) = \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}(-1)^{\langle \Gamma, E_1(x)\oplus E_1(x\oplus\Delta)\rangle}.$$

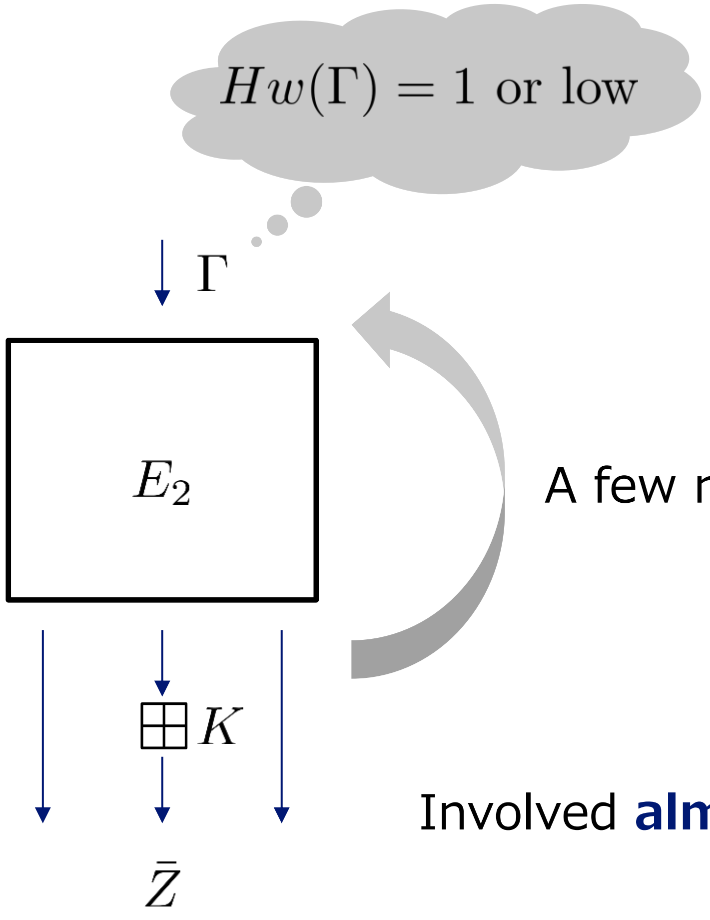
Guess K and check

$$\frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}(-1)^{\langle \Gamma, E_2^{-1}(\bar{Z}-K)\oplus E_2^{-1}(\bar{Z}'-K)\rangle}.$$

Correct guess → high correlation
Wrong guess → random (hypothesis)

Key recovery involves many bits quickly

Quick diffusion by ARX



A few rounds

Involved **almost full bits** of *K*.

Guess K and check

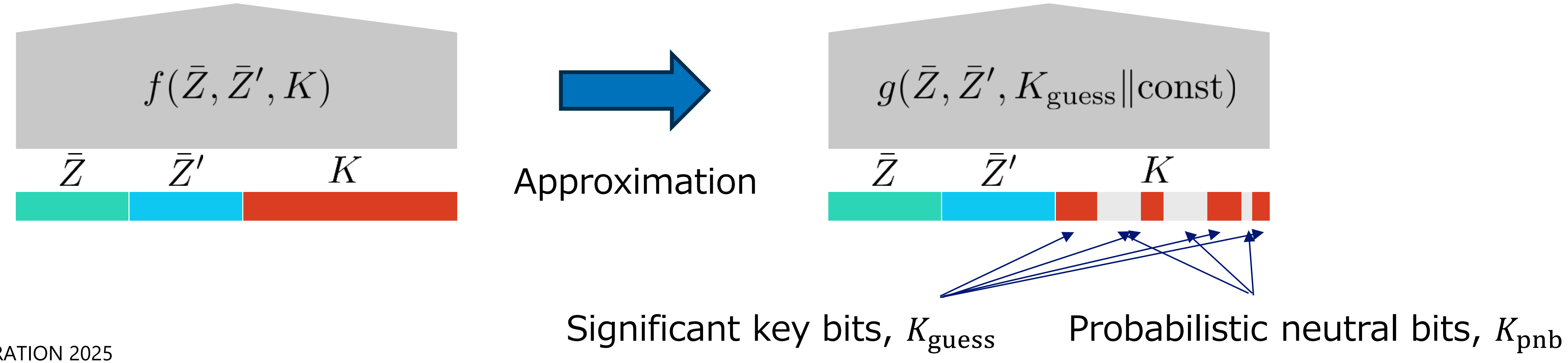
$$\frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}(-1)^{\langle \Gamma, E_2^{-1}(\bar{Z}-K)\oplus E_2^{-1}(\bar{Z}'-K)\rangle}.$$

This cost is extremely high.

PNBs [AFK+08]

Approximate the key-recovery map.

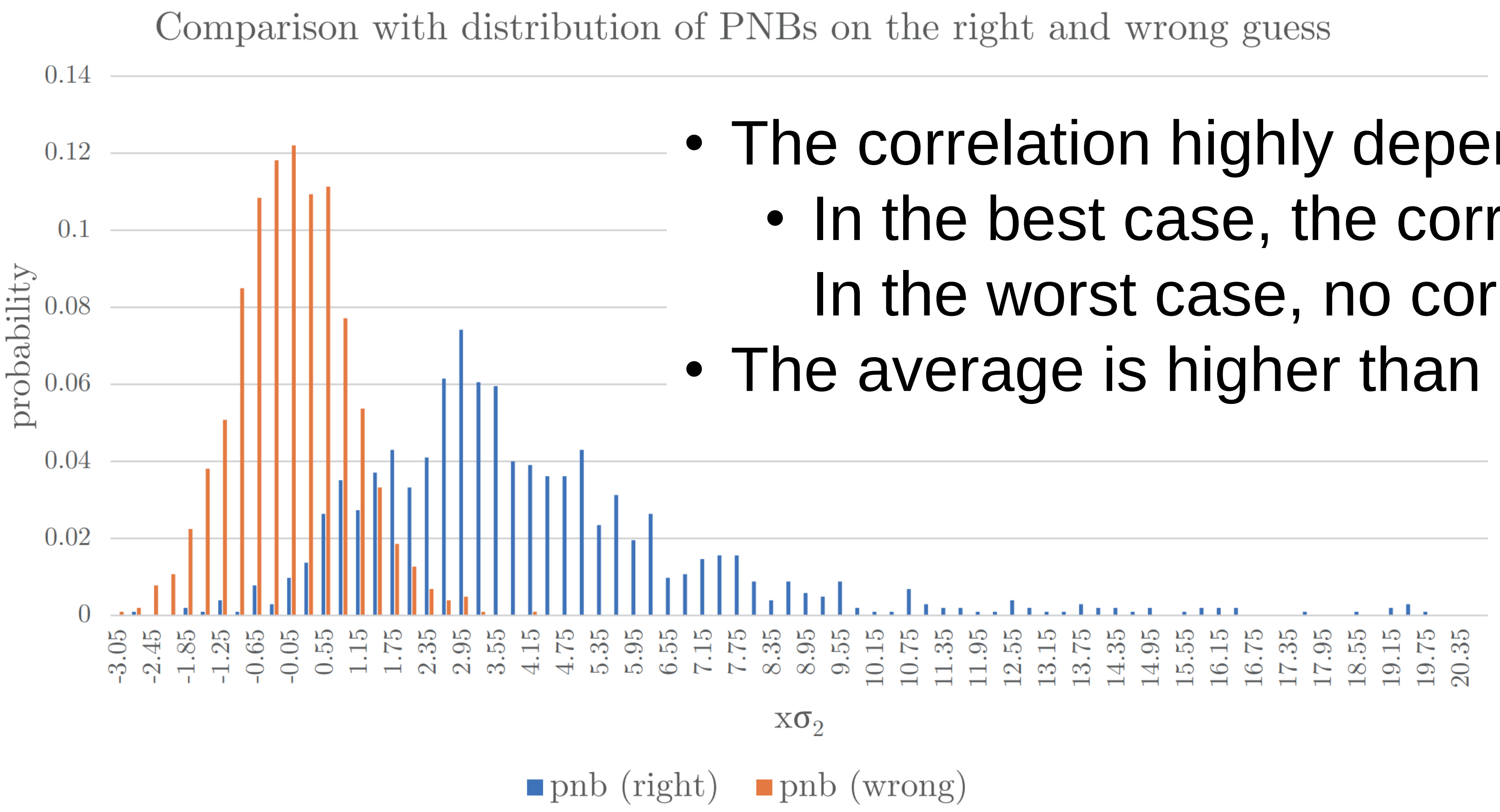
- K is divided into two parts, significant key bits and probabilistic neutral bits, $K = K_{\text{guess}}||K_{\text{pnb}}$.
- We use $f(\bar{Z}, \bar{Z}', K_{\text{guess}}||c)$ for the key recovery, where c is fixed constants (usually, all 0)



What is problem in PNBs?

- We never analyze the inside of E_2 carefully.
  - Set of PNBs is experimentally identified.
  - The resulting correlation is also experimentally obtained.
- There is no plausible evidence that we set 0 for PNBs.
  - Some papers suggested 10* is more adequate, but heuristic.
  - Remember PNBs are **"key"**, which is **unknown** for attackers.

What is problem in PNBs?



- The correlation highly depends on the key.
  - In the best case, the correlation is 1!!
    In the worst case, no correlation.
- The average is higher than the median.

In [AFK+08], the authors recommend to use the median.
Then, the success probability is 50%.

New Theory and New Tool

Puncturing (FT24) –high level idea

- Key recovery function $f: \mathbb{F}_2^n \to \mathbb{F}_2$, and its Walsh spectrum $\hat{f}$.
- Puncturing forces some non-zero entries to 0.

$$\widehat{g}(u) = \begin{cases} \widehat{f}(u) & \text{if } u \notin \mathcal{P}, \\ 0 & \text{if } u \in \mathcal{P}. \end{cases} \qquad \rho^2 = \sum_{u \notin \mathcal{P}} \hat{f}(u)^2 = \langle f, g \rangle = cor(f, g)$$

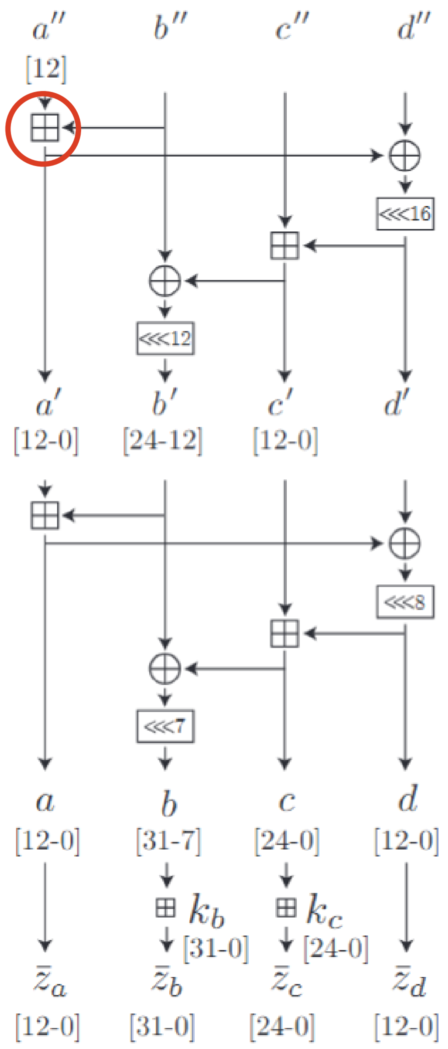
- Use $\hat{g}$ instead of $\hat{f}$ for the key recovery.
  - We need ρ^{-2}-times data complexity.
  - Punctured bits are excluded from the key recovery.
    - › When key bits are excluded, we don't need to guess the key bits.

How to apply puncturing to ARX

- We need to know the Walsh spectrum of the key-recovery to apply puncturing.
- Unlike the S-box-based cipher, it's not easy.
- Example.
  - Assume that we want to evaluate a''[12].

$$(-1)^{a''[12]} = f(z_a[12-0], z_b[31-0], z_c[24-0], z_d[12-0], k_b[31-0], k_c[24-0]).$$

  - It involves 83-bit output and 57-bit key.

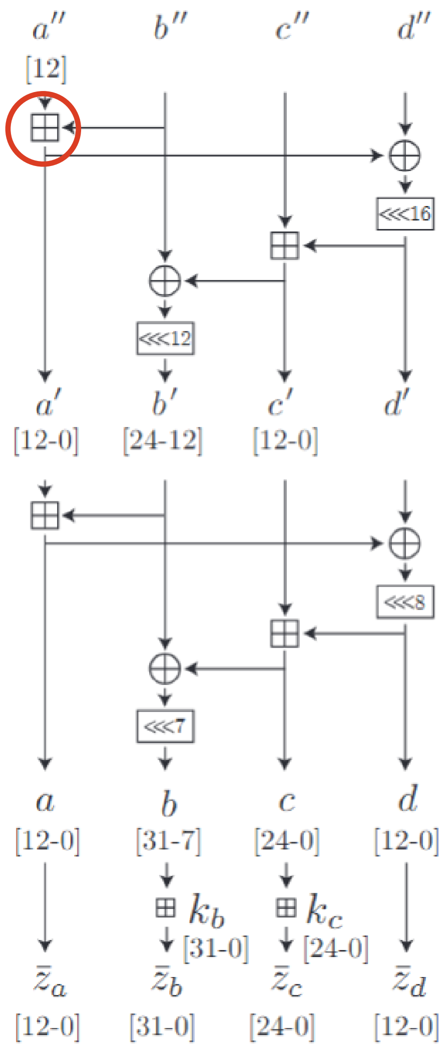


Example, Quarter rounds of ChaCha

Useful observation

- Each Walsh coefficient is the correlation of a linear approximation, and it can be computed as the sum of the (signed) correlations of all linear trails in the approximation's linear hull.
- We enumerate many linear trails to recover Walsh spectrum coefficients.

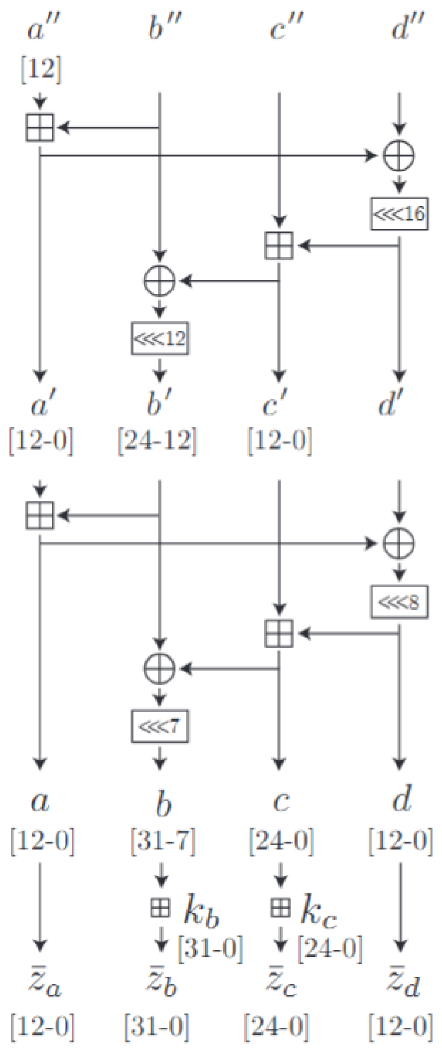
Trail enumeration puncturing



Understanding trail enumeration

1st step, target linear mask.

(00001000, 00000000, 00000000, 00000000) 1



Understanding trail enumeration

1st step, target linear mask.

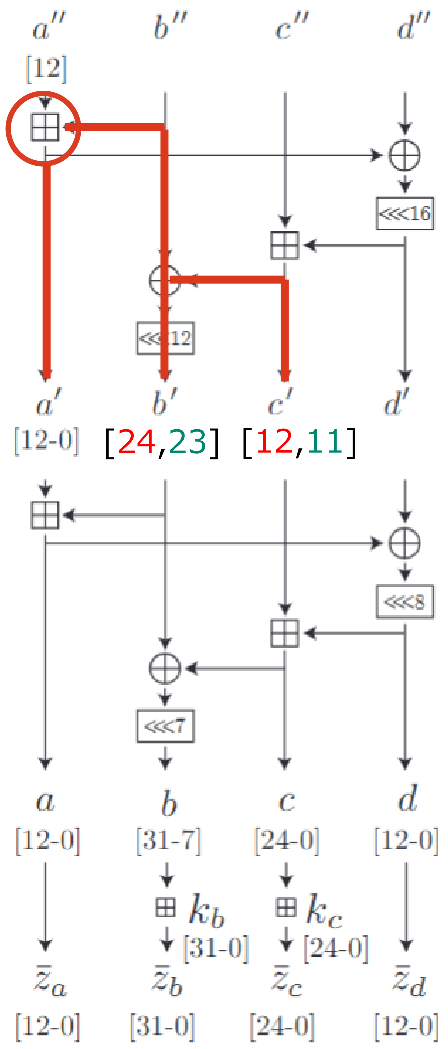
(00001000, 00000000, 00000000, 00000000) 1

2nd step, evaluate linear transition of the modular addition.

(00001800, 01000000, 00001000, 00000000) 2^{-1}
(00001400, 01000000, 00001000, 00000000) 2^{-2}
…
(00001001, 01000000, 00001000, 00000000) 2^{-12}
(00001000, 01000000, 00001000, 00000000) 2^{-12}

(00001000, 01800000, 00001800, 00000000) 2^{-1}
(00001C00, 01800000, 00001800, 00000000) 2^{-2}
…
(00001801, 01800000, 00001800, 00000000) 2^{-12}
(00001800, 01800000, 00001800, 00000000) 2^{-12}

There are 26 coefficients.



Understanding trail enumeration

1st step, target linear mask.

(00001000, 00000000, 00000000, 00000000) 1

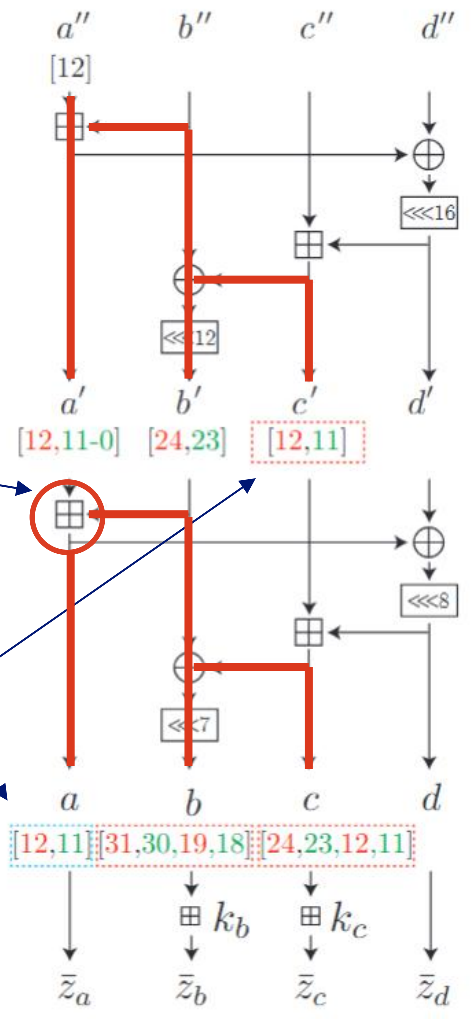
2nd step, evaluate linear transition of the modular addition.

26 coefficients

3rd step, evaluate linear transition of the next modular addition.

(00001000, C0080000, 01801000, 00000000, 00001800) 2^{-12}
(00001000, C00C0000, 01801800, 00000000, 00001800) $2^{-1.58}$
(00001800, C00C0000, 01001800, 00000000, 00001000) $2^{-1.58}$
(00001800, C0080000, 01001000, 00000000, 00001000) 2^{-12}
(00001800, C00C0000, 01801800, 00000000, 00001800) 2^{-12}
(00001800, C0080000, 01801000, 00000000, 00001800) $2^{-1.58}$
(00001000, C00C0000, 01001800, 00000000, 00001000) 2^{-12}
(00001000, C0080000, 01001000, 00000000, 00001000) $2^{-1.58}$

There are 8 coefficients.



Understanding trail enumeration

1st step, target linear mask.

(00001000, 00000000, 00000000, 00000000)   1

2nd step, evaluate linear transition of the modular addition.

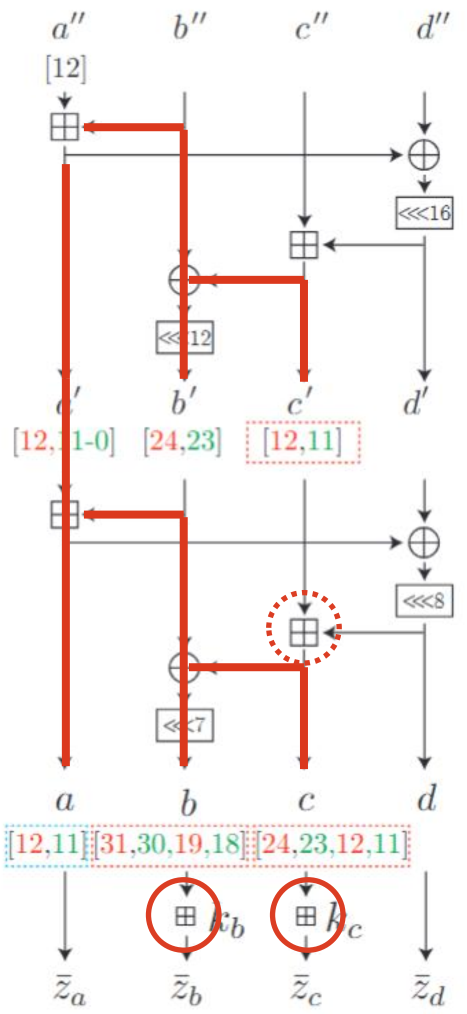
26 coefficients

3rd step, evaluate linear transition of the next modular addition.

8 coefficients

4th step, evaluate linear transition of the last key addition.

We guess Kb[31,30,19,18] and kc[24,23,12,11].
We use 2 bits for each keystream branch.
768 coefficients. 2^{10} dimension. Puncturing correlation $2^{-6.17}$.



Understanding trail enumeration

1st step, target linear mask.

(00001000, 00000000, 00000000, 00000000)   1

2nd step, evaluate linear transition of the modular addition.

26 coefficients

3rd step, evaluate linear transition of the next modular addition.

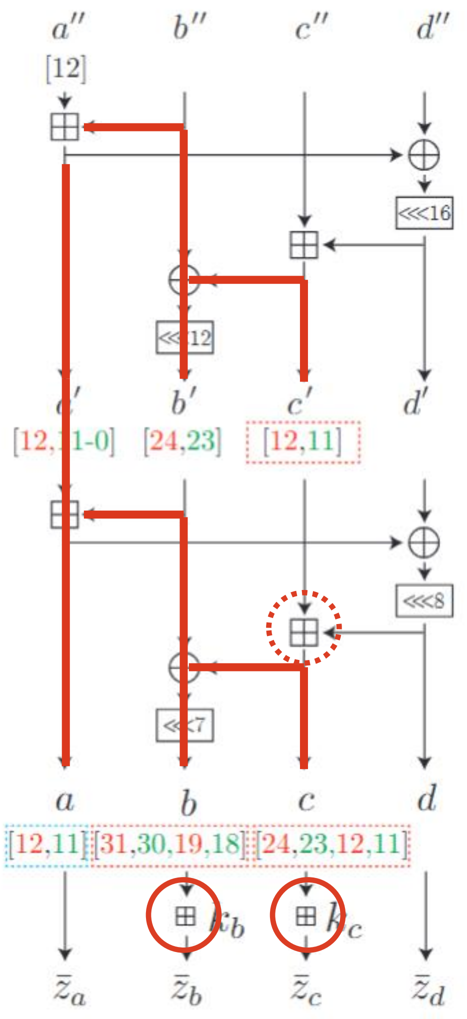
8 coefficients

4th step, evaluate linear transition of the last key addition.

768 coefficients.

Obtain pseudoboolean function, g

Apply the Fast Walsh Transform (FWT), whose cost is 10×2^{10}.
The original function involves **83-bit output and 57-bit key**.
The approximation involves **6-bit output and 4-bit key** only.
To compensate the approximation, we need $2^{6.17}$-times data.



What is different from PNBs?

PNBs

- Experimental
- Each output of the approximation is bool.
- It is the value when probabilistic neutral key bits are set to constants.

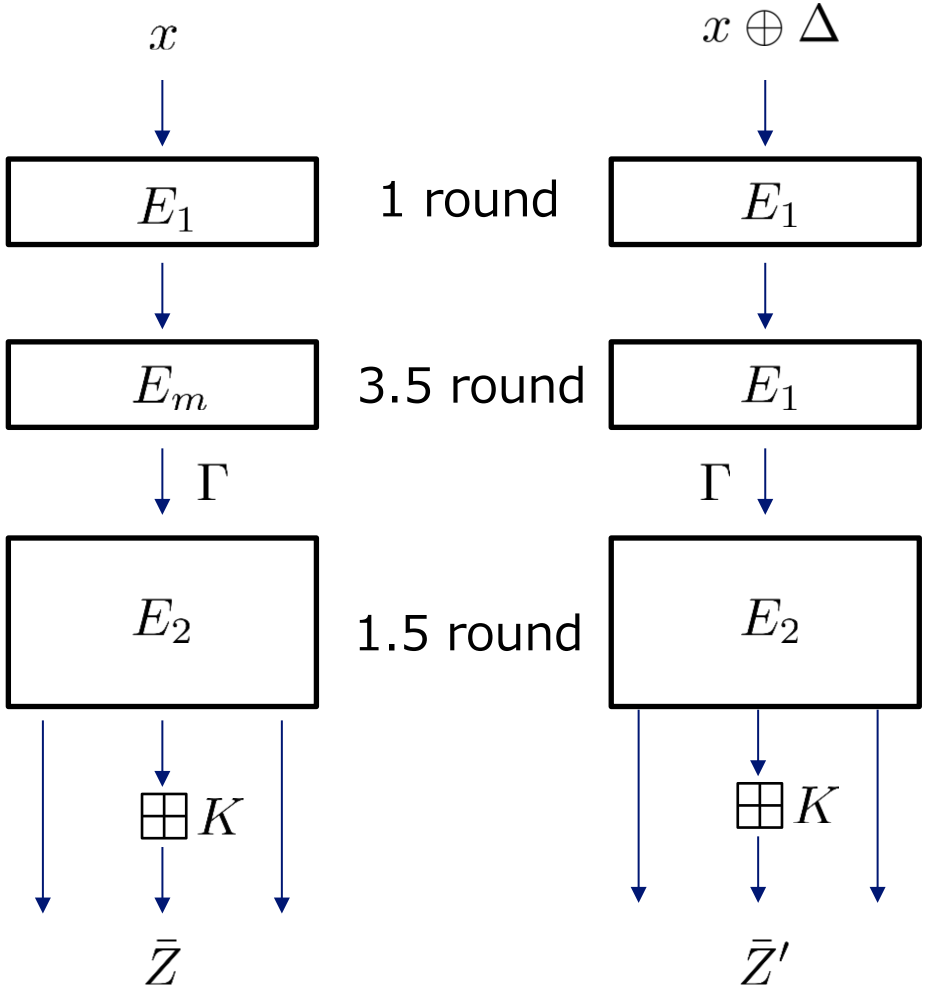
Puncturing

- Theoretical
- Each output of the approximation is real value.
- It is the average over all values of punctured key/output bits.

Key recovery attack on ChaCha6 using Puncturing

Attack against ChaCha6

4.5-round DL distinguisher (BLT20)



$$\Delta^0_{12}[6]$$

$$\downarrow p = 2^{-5}$$

$$(\Delta^1_0[2], \Delta^1_4[29,17,9,5], \Delta^1_8[30,22,10], \Delta^1_{12}[30,10])$$

$$\downarrow c = 2^{-8.3}$$

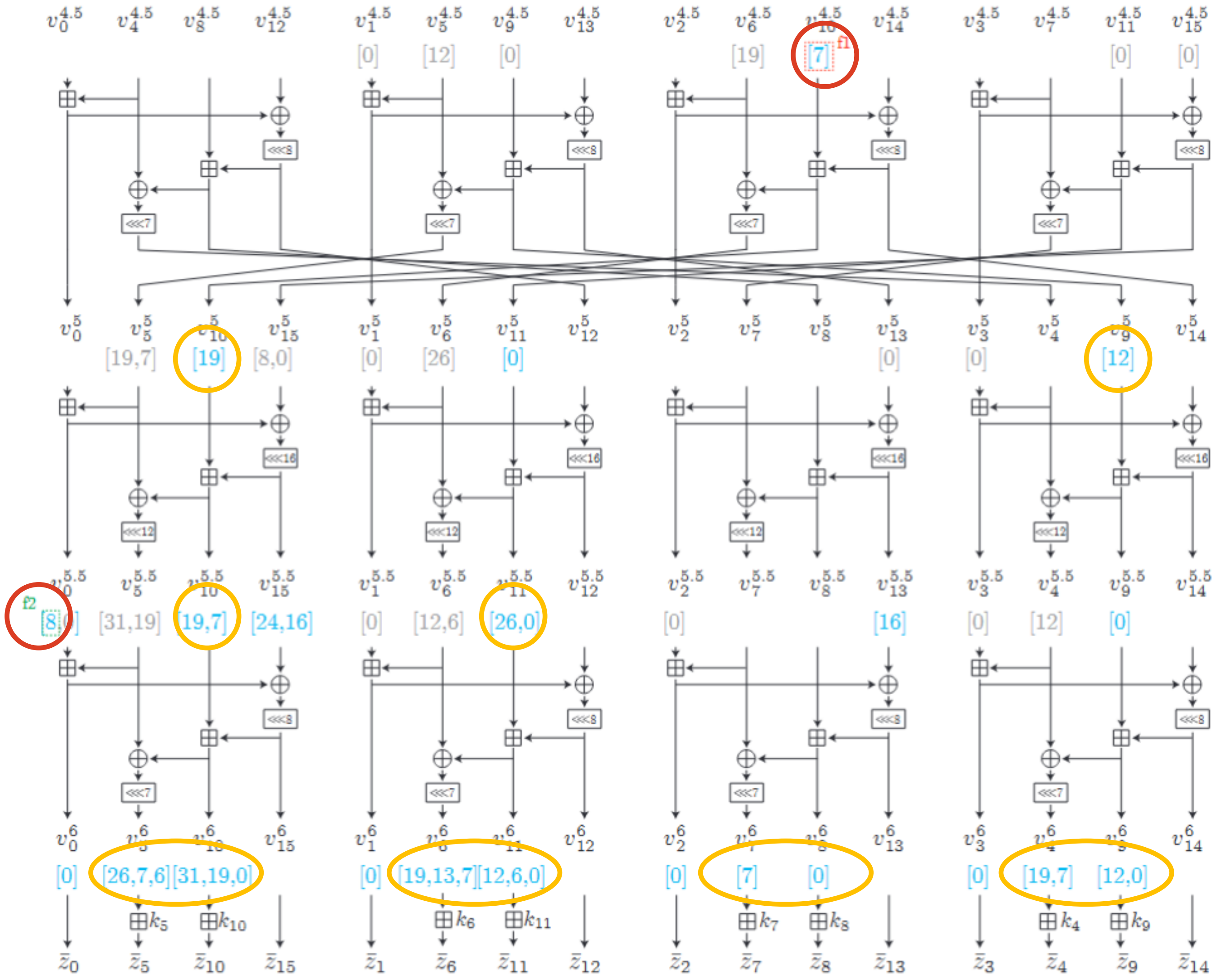
$$(\Gamma^{4.5}_1[0], \Gamma^{4.5}_5[12], \Gamma^{4.5}_6[19], \Gamma^{4.5}_9[0], \Gamma^{4.5}_{10}[7], \Gamma^{4.5}_{11}[0], \Gamma^{4.5}_{15}[0])$$

Key recovery

Attack against ChaCha6

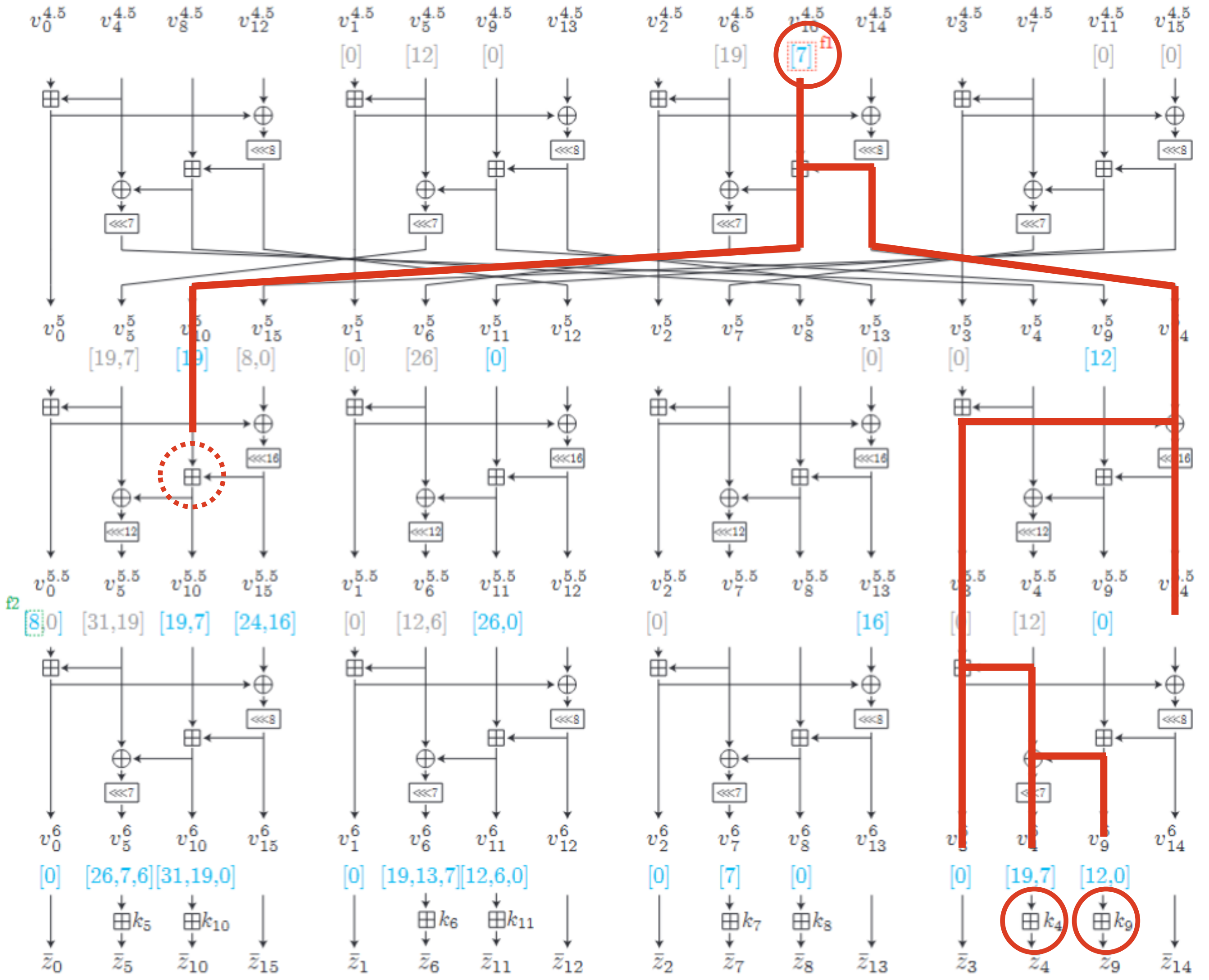
Only **two target bits**, $v_{10}^{4.5}[7]$ and $v_{0}^{5.5}[8]$, involve more than one modular addition.

The others (18) involve at most one modular addition.



Attack against ChaCha6

f1, $v_{10}^{4.5}[7]$



Attack against ChaCha6

f1, $v_{10}^{4.5}[7]$

Guess only 4-bit key.

$k_4[13], k_9[6], k_{10}[6,5]$

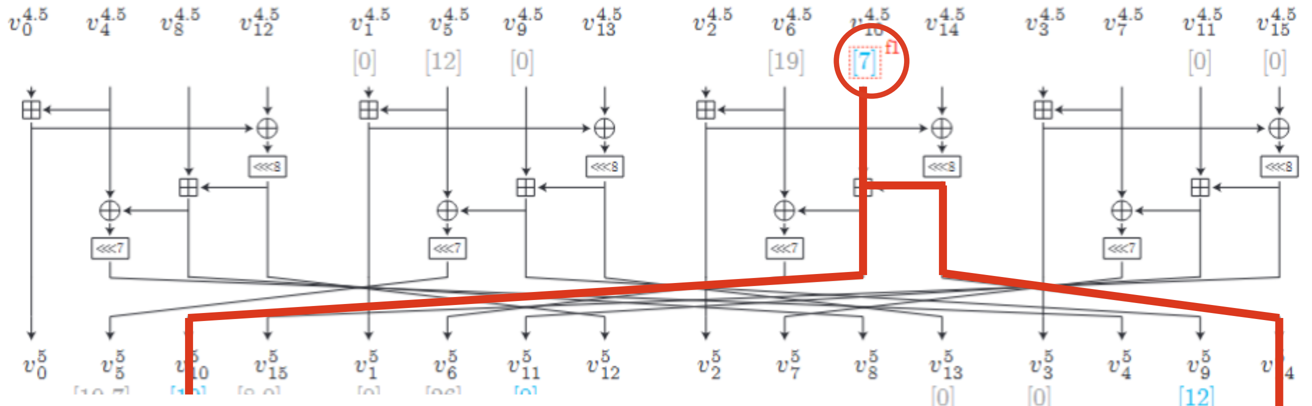
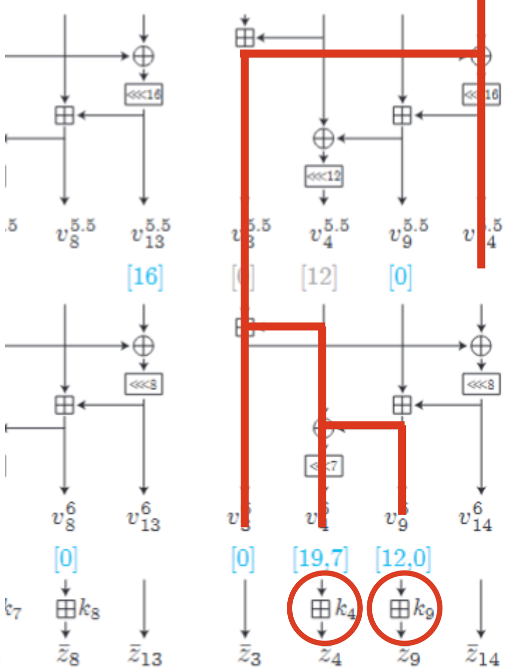


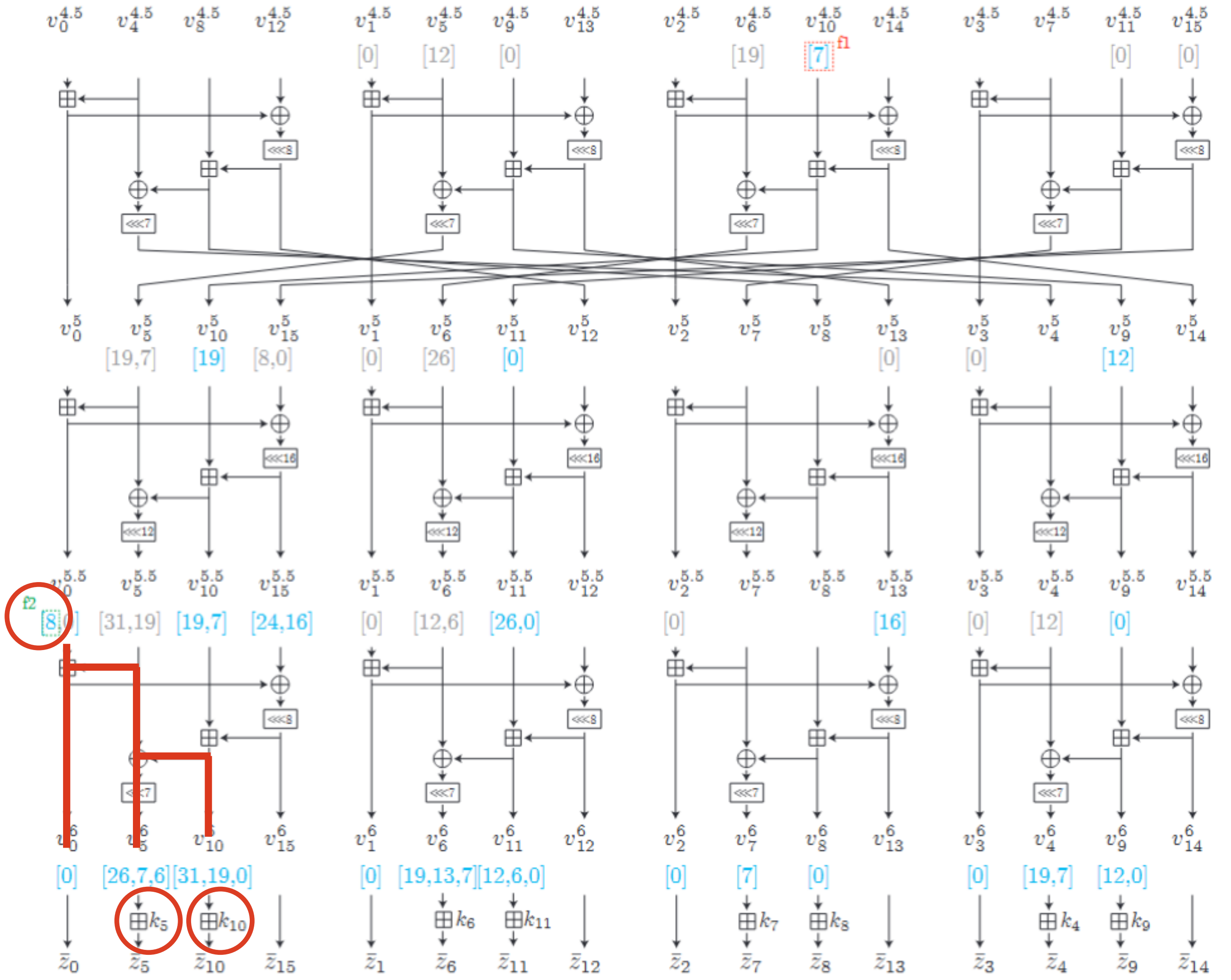
Table 4: Trail enumeration for $f_1 = (-1)^{v_{10}^{4.5}[7]}$

round	active	#coeffs	ρ^2
5	$v_{10}^{5}[7,6\text{-}0]$, $v_{14}^{5}[7,6\text{-}2]$	158	$2^{-0.029}$
5.5	$v_{10}^{5}[7,6\text{-}0]$, $v_{3}^{5.5}[7,6\text{-}2]$, $v_{14}^{5.5}[23,22\text{-}18]$	158	$2^{-0.029}$
6	$v_{10}^{5}[7,6\text{-}0]$, $v_{14}^{5.5}[23,22\text{-}18]$, $v_{3}^{6}[7,6\text{-}2]$, $v_{4}^{6}[14,13\text{-}9]$, $v_{9}^{6}[7,6\text{-}2]$	24248	$2^{-0.15}$
end	$\bar{z}'_{14}[23,22\text{-}18]$, $\bar{z}_3[7,6\text{-}2]$, $v_{10}^{5}[7,6\text{-}0] \leftarrow \{\bar{z}''_{10}[7,6\text{-}2], k_{10}[7,6,5]\}$, $v_4^6[14,13\text{-}9] \leftarrow \{\bar{z}_4[14,13\text{-}9], k_4[14,13]\}$, $v_9^6[7,6\text{-}2] \leftarrow \{\bar{z}_9[7,6\text{-}2], k_9[7,6]\}$	3699840	$2^{-2.31}$



Attack against ChaCha6

f2, $v_0^{5.5}[8]$



Attack against ChaCha6

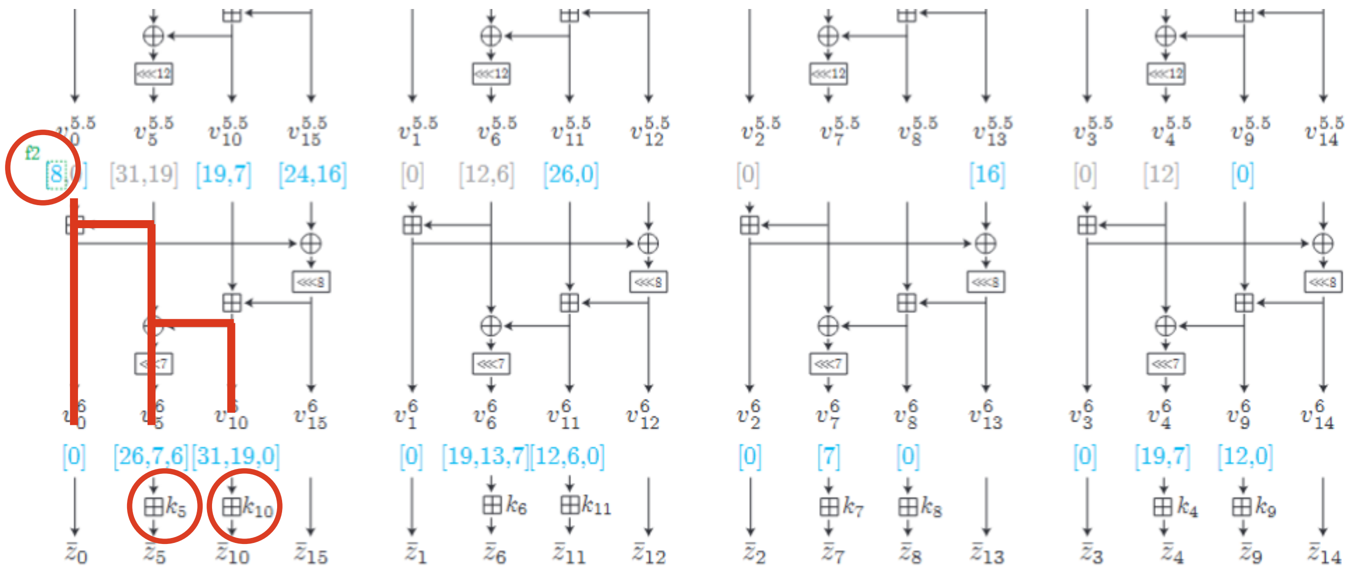
Table 3: Trail enumeration for $f_2 = (-1)^{v_0^{5.5}[8]}$

round	active	#coeffs	ρ^2
6	$v_0^6[8,7\text{-}3]$, $v_5^6[15,14\text{-}10]$, $v_{10}^6[8,7\text{-}3]$	94	$2^{-0.045}$
end	$\bar{z}_0[8,7\text{-}3]$, $v_5^6[15,14\text{-}10] \leftarrow \{\bar{z}_5[15,14\text{-}10], k_5[15,14]\}$, $v_{10}^6[8,7\text{-}3] \leftarrow \{\bar{z}_{10}[8,7\text{-}3], k_{10}[8,7\text{-}5]\}$	18416	$2^{-1.23}$

f2, $v_0^{5.5}[8]$

Guess only 4-bit key.

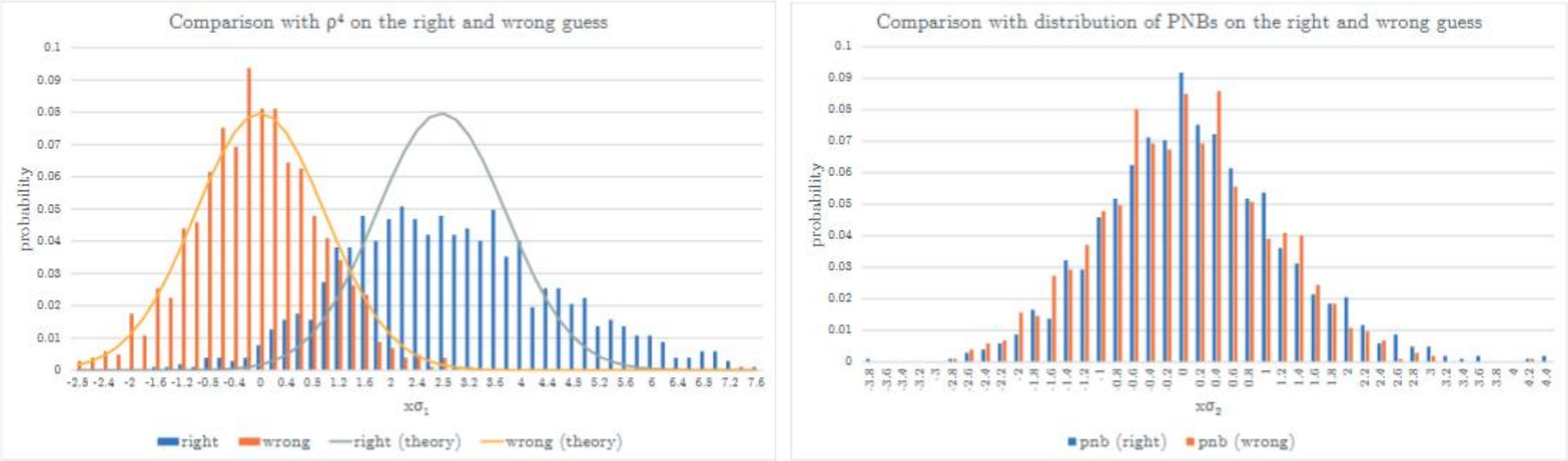
$k_5[14], k_{10}[7,6,5]$



Experiments on 6-round attack

Comparison of backward computation

- 2^{12} samples, 1000 random keys, 6-bit guess for f1 and f2.



(a) Experiments for ρ^4.　(b) Experiments for PNBs.

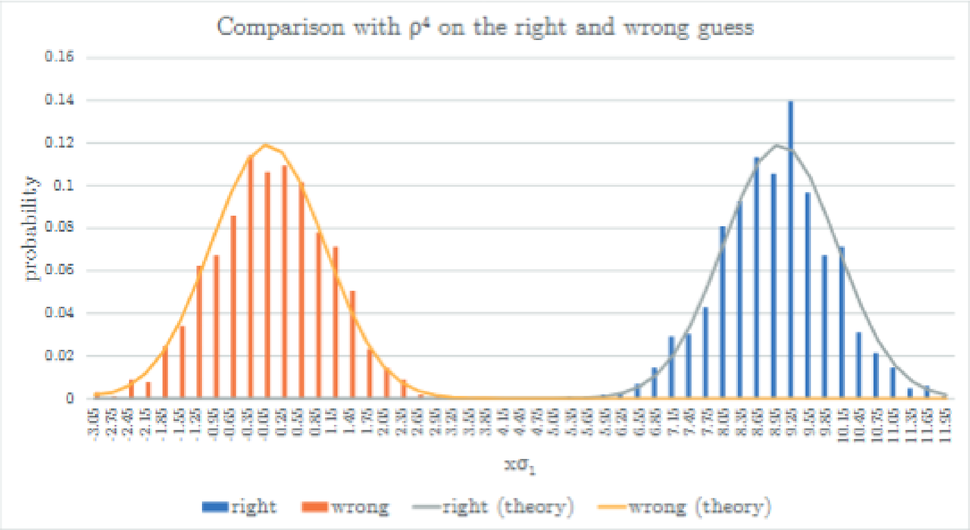
We can easily distinguish both distributions for puncturing but not for PNBs.

Experiments on 6-round attack

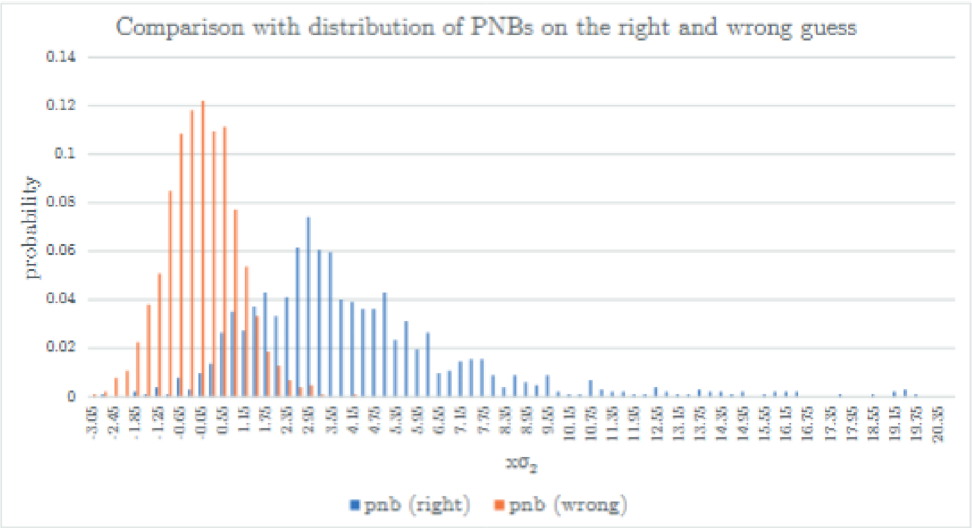
NTT

Comparison of backward computation

- 2^{12} samples, 1000 random keys, 13-bit guess for f1 and f2.



(a) Experiments for ρ^4.



(b) Experiments for PNBs.

PNBs also distinguishable, but clearly, puncturing is better.

Summary

Summary of Results

Round	Data	Time	Note	Ref
6	2^73.7	2^75.5	PNBs w/ syncopation	Wang et al., CRYPTO, 2023
	2^41.6	2^71.0	PNBs w/ linear decomposition	Dey, IEEE-IT, 2024
	2^51,0	**2^61.4**		Ours
	2^55.7	**2^57.4**		Ours
7	2^102.6	2^189.7	DL hull and PNBs	Xu et al., ToSC, 2024
	2^127.7	**2^148.2**		Ours
	2^102.9	**2^154.2**		Ours
7.5	2^32.6	2^255.2	PNBs w/ linear decomposition	Dey, IEEE-IT, 2024
	2^127.1	**2^250.2**		Ours

Summary

- A new tool to analyze ChaCha.
  - No more PNBs. Fully theoretical analysis is possible!!
- Significant improvement for ChaCha.
  - 6 rounds, 2^{71} time → $2^{57.4}$ time
  - 7 rounds, $2^{189.7}$ time → $2^{154.2}$ time
  - 7.5 rounds, $2^{255.2}$ time → $2^{250.2}$ time

Each cost is one table lookup.
We regard the cost is equivalent with one encryption.

- Open problem
  - How to automate and optimize the analysis?
  - So far, choosing parameter is our heuristic.