A New Approach to Improved PNB-based Attacks on Reduced-round ChaCha and Salsa

Goutam Paul, Anup Kumar Kundu, and Sucheta Chakrabarti

Indian Statistical Institute, Kolkata,
203 Barrackpore Trunk Road, Kolkata 700108, INDIA.
goutam.paul@isical.ac.in, anupkundumath@gmail.com,
suchetadrdo@hotmail.com

Abstract. ChaCha and Salsa are two ARX based stream ciphers which are widely used in data encryption including TLS v1.3 standard, VPN software etc. Exploiting Probabilistic Neutral Bits (PNB) is one of the most significant cryptanalysis strategies for reduced-round versions of these ciphers. The seminal work using PNB by Aumasson et al. (FSE 2008) claims that the PNB set mostly depends on the output bit difference occurring in the intermediate round. The subsequent works mainly relied on the differential or differential-linear cryptanalysis, or multiple distinct input-output differentials for which the bias is higher than a threshold in the intermediate round. In this paper, we propose a new PNB set construction based on multiple output bit differences with respect to a single input bit difference only. We exploit the differentials to mount key recovery attacks using a multi-step procedure depending on our new PNB set. Our attack achieves a time complexity of $2^{167.90}$ for 7-round ChaCha and $2^{183.54}$ for 8-round Salsa in 256-bit versions, beating all the existing PNB-based attacks on 7-round ChaCha and 8-round Salsa by a significant margin. Further, both our time and data complexities for 7.5-round ChaCha are better than the latest published works by Flórez-Gutiérrez and Todo (Eurocrypt 2025 and Asiacrypt 2025). For 128-bit versions, our attack procedure results in a time complexity of $2^{115.24}$ for 6.5-round ChaCha and $2^{98.40}$ for 7.5-round Salsa respectively. We have experimentally validated the underlying independence assumptions across all ciphers, alongside our attack procedure on different reduced-round toy versions of ChaCha.

Keywords: Stream Cipher, ChaCha, Salsa, ARX Cipher, Probabilistic Neutral Bit (PNB), Bias, Differential attack, Differential-Linear attack, ToyChaCha.

1 Introduction

Symmetric key cryptography is used in a wide range of hardware and software products, such as data encryption, secure communication, disk encryption, database security, VPN software etc. In the symmetric key paradigm, stream ciphers are faster than block ciphers on application scenarios, and therefore are more widely used in practical applications and hardware implementations.

Salsa [4] is a stream cipher, designed by Bernstein in 2005 and was a candidate in the eStream project [35]. The original version of the cipher has 20 rounds. There are also reduced-round variants of the cipher with 8 and 12 rounds. The version 12-round Salsa was accepted in the phase 3 of the project by receiving the highest votes in 2008. Since the cipher was proposed as a candidate in the eStream project, many attacks [1],[9],[19],[23],[29],[34],[36] have been mounted on the cipher.

ChaCha [5] is another stream cipher which was also proposed by Bernstein in the year 2008. The structure of the cipher is similar to Salsa. The `quarterround` operation of the round function of ChaCha applies on the rows and the diagonals of the state matrix instead of the rows and the columns, as are in Salsa. Due to this, the diffusion in the state bits is also higher in ChaCha than in Salsa.

1.1 Summary of Existing Works

Most of the previously reported cryptanalysis of the above ciphers start by setting an input bit difference in the initial state of the cipher and then finding the corresponding output difference in a bit position at the end of an intermediate round. The next step is to perform a meet-in-the-middle approach by first reaching the reduced final round R of the targeted cipher, and then coming back to the r-th intermediate round, $r < R$, by applying the inverse function $(R - r)$ times. The first attack using this strategy was proposed by Crowley [9] to find a truncated differential for 5 rounds of Salsa by going 3 forward rounds, and then coming back for 2 rounds. Later, Aumasson et al. [1] introduced the idea of Probabilistic Neutral Bit (PNB) and proposed an attack for 8 rounds of Salsa by going 4 forward rounds and coming back for 4 rounds. For ChaCha, the attack [1] worked for 7 rounds with 3 forward and 4 backward rounds, respectively.

Afterwards, many authors explored different variants of PNB set formation and improved the attack complexity [25],[24],[13],[37],[33]. Some other works have used predefined size of PNB [18],[2] as well. Shi et al. [34] extended the previously proposed PNB work by column and row chaining distinguishers. They further proposed new second order differential trails and generalized the notion of probabilistic neutral bits (PNB) to probabilistic neutral vectors (PNV) by flipping more than one input bit simultaneously and improved the attack complexity.

Some PNB-based works [6],[8],[14],[30],[3],[7],[12],[39],[2],[17] explored the combination of the differential and linear cryptanalysis, some of which have penetrated into fractional rounds of the ciphers. Moreover, the last two works ([2],[17]) used an additional technique of the hamming weight ($\mathcal{HW}$) restriction of the state matrix.

In [3], the authors found a new differential-linear distinguisher[1] by giving 2-bit differences at the beginning of the differential part of the differential-linear trail. They further used Mixed Integer Linear Programming (MILP) tool to find

[1] This may also be termed as a differential attack with linear extension.

trails for the linear part. Using this, they found 5 round differential-linear trail and provided a key recovery attack for 7 rounds of 256 bit versions.

In [6], the authors proposed the first multi-bit (using different input-output differences each time) differential-linear strategy to find the forward bias after 3.5, 4, 4.5 rounds of ChaCha and 5 rounds of Salsa cipher. In addition to multiple input-output differentials, the work [11],[22] used a systematic multi-step approach to maximize the size of the PNB set based on neutrality measurement parameter.

In [21], the authors have introduced a novel theoretical methodology, called bit puncturing, as an alternative to the PNB technique against the cipher ChaCha. This bit puncturing approach is based on theoretical analysis of the Walsh spectrum, which aims to construct an approximated key-recovery map by ignoring certain input bits. The purpose is to reduce the number of active input bits from the secret key, so as to simplify the complexity of the key recovery procedure. This yields time complexities of $2^{154.2}$ and $2^{250.2}$ for 7 and 7.5 rounds respectively of 256-bit ChaCha.

A subsequent work [20] by the same authors proposes a divide-and-conquer approach to approximate the key recovery function by decomposing its Walsh spectrum into smaller components. This reduces the time complexities on 7.5 rounds of 256-bit ChaCha to $2^{243.6}$ and on 8.5 rounds of 256-bit Salsa to $2^{219.7}$. Further, for 128-bit Salsa, they report complexities of $2^{86.7}$ and $2^{125.8}$ on 7 and 8 rounds respectively.

Both our data and time complexities on ChaCha 7.5 rounds outperform the latest reported work proposed in [20]. The attack complexities of relevant important existing works are presented in Table 1 in comparison with our results.

1.2 Our Contributions

Here we enlist our main contributions one by one.

1. In general, all the existing PNB based differential / differential-linear cryptanalysis on ChaCha and Salsa utilize only a specific output bit-difference for any single input bit-difference. More specifically, all the existing works either explored single input-output ($\mathcal{ID}$-$\mathcal{OD}$) differential or multiple input-output ($\mathcal{ID}_i$-$\mathcal{OD}_i$) differentials with linear combinations. We first investigate the differential distinguisher in the unexplored domain of single input-multiple output ($\mathcal{ID}$-$\mathcal{OD}_i$) differentials.
2. In the seminal work on the PNB based attacks, the authors mention the following [1, Section 3.5].

   > "The number n of PNB 's for some fixed threshold γ mostly depends on the output difference, but not on the input difference."

   We construct a new PNB set (PNBS) by taking the union of proper choices of PNB sets, each of which is related to separate output bit-differences corresponding to a single input bit-difference. Our construction method is described in Section 3 and Algorithm 2. Thus, the choice of the input bit-difference also plays a significant role in determining the PNB set, contrary to the claim of the seminal work [1] and all subsequent follow-ups.

3. Further, we partition this new type of PNB set into suitably crafted disjoint subsets to mount a divide-and-conquer algorithm on those parts, leading to an improved key recovery attack on reduced-round ChaCha and Salsa. This partitioning method is detailed in Section 4 and Algorithm 3. In particular, we show that with a proper choice of input bit-difference and attack parameters, our method achieves key recovery of 7 round, 7.5 round of ChaCha with a time complexity of $2^{167.90}$, $2^{237.41}$ and that of 8 round Salsa with a time complexity of $2^{183.54}$ for 256-bit versions. For 128-bit versions, the time complexity of our attack becomes $2^{115.24}$ and $2^{98.40}$ for 6.5 rounds of ChaCha and 7.5 rounds of Salsa respectively. The comparison of our results for ChaCha and Salsa with respect to all the previous works, is given in Table 1.
4. An obvious way to compute the backward bias for $\mathcal{ID}$-$\mathcal{OD}_i$ is to put random values in the corresponding PNB set P_i and actual values to the rest. However, as validated from our experiments with ToyChaCha, such assignments do not guarantee the independence of forward and backward biases which is a crucial requirement of the attack model. Rather, our observation establishes that we need to put random bits in the union of all PNBs to compute the backward bias for any $\mathcal{ID}$-$\mathcal{OD}_i$'s (irrespective of the particular P_i). This is a non-trivial generalization to handle multiple $\mathcal{OD}$s from a single $\mathcal{ID}$.
5. Dey et al. [13] proposed a toy version of ChaCha, called ToyChaCha to verify the PNB attacks on ChaCha. We have introduced another two variants of toy versions, namely ToyChaCha48 and ToyChaCha64. We have experimentally verified the independence assumptions and our key recovery attack procedure across all these three toy versions of ChaCha (ToyChaCha, ToyChaCha48 and ToyChaCha64).
6. We have also used both differential and differential-linear strategies, to apply our attack on 7.5-round ChaCha. We have observed that, by using only differential approach (without any additional linear or intermediate condition) we can attack 7.5-round ChaCha. This has not been done in any of the previous works to the best of our knowledge. The approaches are discussed in Section 5.2. For both the approaches, the data and the time complexities become $2^{53.41}$ and $2^{237.41}$ using our attack procedure. Both our data and time complexities of 7.5-round ChaCha outperform the latest reported work proposed in [20].

Advantage of Our Work. The advantage of our new construction procedure is derived from the following observations.

- The cardinality of our new PNB set is larger than that of the single input-output bit-difference case.
- It creates more partitions (≥ 2) of the key space.
- The key recovery procedure uses the partitions in step by step manner. This reduces the overall time complexity.

The advantage of choosing multiple output differences from the same input difference is that, in this case, we can reuse the same generated keystream pairs to recover the key bits in the key recovery attack. Hence the key recovery attack

Cipher	Rounds	Attack Strategy	Data	Time	References	Year
256-bit version of ChaCha	7	Differential	2^{27}	2^{248}	[1]	2008
			$2^{23.89}$	$2^{238.94}$	[24]	2016
			$2^{103.3}$	$2^{210.3}$	[37]	2023
			$\mathbf{2^{68.90}}$	$\mathbf{2^{167.90}}$	**This work**	2026
		Bit Puncturing	$2^{127.7}$	$2^{148.2}$	[21]	2025
		Differential-Linear	2^{28}	2^{233}	[6]	2016
			$2^{32.87}$	$2^{235.2}$	[18]	2017
			2^{224}	2^{224}	[8]	2021
			$2^{48.83}$	$2^{230.86}$	[2]	2022
			$2^{90.20}$	$2^{221.95}$	[14]	
			$2^{87.18}$	$2^{218.92}$	[15]	2023
			$2^{110.81}$	$2^{206.8}$	[3]	
			$2^{93.79}$	$2^{192.89}$	[12]	2024
			$2^{102.63}$	$2^{189.7}$	[39]	
			$2^{101.09}$	$2^{178.12}$	[33]	
			$2^{101.15}$	$2^{192.15}$	[30]	2025
	7.5	Differential	$\mathbf{2^{53.41}}$	$\mathbf{2^{237.41}}$	This work	2026
		Differential-Linear	$2^{125.8}$	$2^{242.9}$	[37]	2023
			$2^{32.64}$	$2^{255.24}$	[12]	2024
			$2^{34.47}$	$2^{253.23}$	[33]	
			$2^{100.8}$	$2^{223.9}$	[39]	
			$\mathbf{2^{53.41}}$	$\mathbf{2^{237.41}}$	This work	2026
		Bit Puncturing	$2^{127.1}$	$2^{250.2}$	[21]	2025
			$2^{127.1}$	$2^{243.6}$	[20]	
256-bit version of Salsa	8	Differential	2^{31}	2^{251}	[1]	2008
			$2^{32.11}$	$2^{247.2}$	[25]	2015
			$2^{22.46}$	$2^{245.52}$	[24]	2016
			$\mathbf{2^{61.54}}$	$\mathbf{2^{183.54}}$	This work	2026
		Differential-Linear	2^{96}	$2^{244.9}$	[6]	2016
			$2^{30.86}$	$2^{243.7}$	[18]	2017
			$2^{113.14}$	$2^{217.14}$	[7]	2023
			$2^{117.29}$	$2^{210.32}$	[11]	2024
			$2^{99.73}$	$2^{186.01}$	[17]	2025
	8.5		$2^{99.47}$	$2^{245.84}$	[17]	
		Bit Puncturing	$2^{126.7}$	$2^{219.7}$	[20]	
128-bit version of ChaCha	6	Differential	2^{30}	2^{107}	[1]	2008
		Differential-Linear	$2^{38.66}$	$2^{84.39}$	[14]	2022
			$2^{40.75}$	$2^{78.77}$	[15]	2023
			$2^{41.63}$	$2^{71.03}$	[12]	2024
	6.5		$2^{66.94}$	$2^{123.04}$	[14]	2022
			$2^{66.29}$	$2^{121.4}$	[15]	2023
			$2^{37.27}$	$2^{113.08}$	[12]	2024
		Differential	$\mathbf{2^{67.24}}$	$\mathbf{2^{115.24}}$	**This work**	2026
128-bit version of Salsa	7	Differential	2^{21}	2^{111}	[1]	2008
		Differential-Linear	$2^{28.77}$	$2^{102.82}$	[16]	2025
			$2^{86.67}$	$2^{98.25}$	[17]	
	7.5		$2^{23.06}$	$2^{124.22}$	[16]	
			$2^{94.16}$	$2^{111.47}$	[17]	
		Differential	$\mathbf{2^{58.40}}$	$\mathbf{2^{98.40}}$	**This work**	2026
	8	Bit Puncturing	$2^{110.3}$	$2^{125.8}$	[20]	2025

Table 1. Data and time complexities of existing PNB and bit puncturing based attacks on reduced-round ChaCha and Salsa.

can be performed more efficiently in a multi-step procedure, which is discussed later.

Organization of the paper. Section 1 presents the introduction, a summary of existing works along with the motivation and contribution of this work. Section 2 revisits the notations used through out the paper, the description of the ciphers Salsa and ChaCha and the original PNB work by Aumasson et al. and discuss the terminologies and definitions used to find PNB set and mount the key recovery attack. Section 3 discusses how we construct our new PNB set to mount an improved attack. Our key recovery attack procedure is presented in Section 4. The final attack parameters and complexity results for all the ciphers, are given in Section 5. The verification of our attack on different toy versions of ChaCha, is presented in Section 6. Finally, we conclude the paper in Section 8.

2 Preliminaries and Background

In this section, first we introduce the notations used throughout the paper, followed by brief descriptions of ChaCha and Salsa. Then we revisit the existing notions, terminologies and PNB construction strategies.

2.1 Notations

All the notations used in the paper are given in the following:

Notation	Description
$X, X^{(r)}$, $X_{i,j}^{(r)}$	Initial state matrix, state matrix after the r-th round, and the j-th bit of its i-th word in the state matrix
$\Delta_{p,q}^{(r)}$	Difference of the states at the q-th bit of the p-th word at round r, defined as $X_{p,q}^{(r)} \oplus X'^{(r)}_{p,q}$
$\mathcal{ID}$, $\mathcal{OD}$	Input difference ($\Delta_{p,q}^{(0)}$) at initial round and intermediate output difference ($\Delta_{p,q}^{(r)}$) at r-th round
$\boxplus_n, \boxminus_n$	Addition, subtraction over $\mathbb{F}_2^n$
$\lll_n$	Circular left rotation over $\mathbb{F}_2^n$
R, r	Total number of full rounds of the cipher, and the intermediate round used to evaluate biases
$\mathcal{B}, v$	Output differential set at the r-th round, and its cardinality $v = \lvert\mathcal{B}\rvert$
P_t, P'_t	PNB set corresponding to the output difference $\Delta_{p_t,q_t}^{(r)} \in \mathcal{B}$, and its mutually exclusive version
$\mathbb{K}, m$	Keyspace of the cipher, and the number of significant key bits with respect to $\bigcup_{t=1}^{v} P_t$
$\epsilon_d^*[t]$, $\epsilon_a^*[t]$	Forward bias of $\Delta_{p_t,q_t}^{(r)}$ at round r, and its backward bias evaluated after assigning random values across union of PNB bits
$Th_u^{(t)}$	Threshold for the union of t PNB sets in our key recovery attack
(N_u, T_u)	Data complexity and time complexity of our attack
$NumIV$, $NumKey$	Number of randomly chosen initialization vectors (IVs) and keys

Table 2. Summary of notations used in this paper

In Table 2, an input difference ($\mathcal{ID}$, denoted by $\Delta_{p,q}^{(0)}$) represents a bit difference is given in the q-th bit of the p-th word in the initial round. Correspondingly,

an output difference ($\mathcal{OD}$, denoted by $\Delta_{p,q}^{(r)}$) at some r-th round indicates a bit difference appears with a certain bias at $X_{p,q}^{(r)}$ in the r-th intermediate round.

2.2 Salsa

Salsa [4] is a stream cipher that consists of R rounds with $R = 20$. It has state size 512 bits. A Salsa state can be represented by a 4×4 matrix, called a state matrix. Each cell of the state matrix consists of 32-bits, called a word. The initial state matrix of the cipher has 4 words of predefined round constants $c = (c_0, c_1, c_2, c_3)$, 8 words of key $k = (k_0, \cdots, k_7)$, and 4 words of initial vector (IV) $v = (v_0, v_1, v_2, v_3)$. The initial state matrix can be represented by the following:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ v_2 & v_3 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}.$$

Depending upon the key sizes, i.e., 128 bits or 256 bits, the cipher has two variants. In the 256-bit version, distinct 256-bit key is given in 8 words of the initial state. In the other one, 128-bit key is given repeatedly in (k_0, k_1, k_2, k_3) and (k_4, k_5, k_6, k_7). The round function of the cipher has `columnround` / `rowround` function depending upon the even / odd round. Each of these functions calls another function, named `quarterround`, which is applied on the 4 words of the columns for `columnround` and rows for `rowround` in the corresponding state. This process is repeated for R rounds to generate the final state. Each `quarterround` function updates 4 words of 32-bit as the following:

$$b = b \oplus ((a \boxplus_{32} d) \lll_{32} 7), \tag{1}$$
$$c = c \oplus ((b \boxplus_{32} a) \lll_{32} 9), \tag{2}$$
$$d = d \oplus ((c \boxplus_{32} b) \lll_{32} 13), \tag{3}$$
$$a = a \oplus ((d \boxplus_{32} c) \lll_{32} 18). \tag{4}$$

Finally after adding the initial state with the final one i.e. $Z = X \boxplus_{32} X^{(20)}$, the keystream bits are generated.

Inverse function. The inverse function of the cipher differs in the `quarterround`. The inverse of the `quarterround` function updates 4 words (a, b, c, d) by applying the Eq. (1), (2), (3), (4) in reverse order. After subtracting the initial state from the generated keystream within modulo 2^{32}, the inverse `rowround` and inverse `columnround` functions are called. Like the encryption procedure (Enc), inverse `columnround` (`rowround`) applies inverse `quarterround` in each column (row) in the state of the cipher. After repeating the process for R rounds, the initial state is generated.

2.3 ChaCha

ChaCha [5] is a variant of Salsa, that contains R rounds with $R = 20$. The structure of the cipher is similar to its predecessor Salsa. Here it takes round constant (c), key (k), and IV (v) cells and forms the initial state matrix as the following:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ v_0 & v_1 & v_2 & v_3 \end{pmatrix}.$$

It has also two versions depending upon 128-bit and 256-bit keys. The difference between these two is similar with the difference between the two versions in Salsa. Irrespective of applying the `rowround` and `columnround`, here the `columnround` and `diagonalround` function is called in the even and odd round respectively, for R rounds to generate the final state. A `quarterround` function is applied on each column in `columnround` and diagonal i.e. (x_0, x_5, x_{10}, x_{15}), (x_1, x_6, x_{11}, x_{12}), (x_2, x_7, x_8, x_{13}), and (x_3, x_4, x_9, x_{14}) in `diagonalround` of the state respectively. The non-linear `quarterround` function updates the words a, b, c, $d \in \mathbb{F}_2^{32}$, as given in the following:

$$a = a \boxplus_{32} b, \qquad d = d \oplus a, \qquad d = d \lll_{32} 16, \tag{5}$$
$$c = c \boxplus_{32} d, \qquad b = b \oplus c, \qquad b = b \lll_{32} 12, \tag{6}$$
$$a = a \boxplus_{32} b, \qquad d = d \oplus a, \qquad d = d \lll_{32} 8, \tag{7}$$
$$c = c \boxplus_{32} d, \qquad b = b \oplus c, \qquad b = b \lll_{32} 7. \tag{8}$$

The keystream bits are generated by adding the initial state with the final one. In [2], [27], [6] authors discuss about $(R + 0.25)$ and $(R + 0.5)$ rounds for some R number of rounds. Here they have taken only Eq. (5) and Eq. (5), (6) respectively in the `quarterround` function.

Inverse function. Similar to Salsa, here also firstly the initial state matrix is subtracted from the final keystream bits. Then inverse `columnround` and inverse `diagonalround` is applied periodically on the state matrix depending on the round number for generating the initial state. These functions mainly consist of inverse `quarterround` function that takes 4 words (a, b, c, d) and apply the following operations:

$$b = b \lll_{32} 25, \qquad b = b \oplus c, \qquad c = c \boxminus_{32} d, \tag{9}$$
$$d = d \lll_{32} 24, \qquad d = d \oplus a, \qquad a = a \boxminus_{32} b, \tag{10}$$
$$b = b \lll_{32} 20, \qquad b = b \oplus c, \qquad c = c \boxminus_{32} d, \tag{11}$$
$$d = d \lll_{32} 16, \qquad d = d \oplus a, \qquad a = a \boxminus_{32} b. \tag{12}$$

Note that, for the inversion of $(R+0.25)$ and $(R+0.5)$ rounds, Eq. (12) and Eq. (12), (11) are used in the inverse `quarterround` function.

2.4 Revisiting Existing Constructions of Probabilistic Neutral Bits (PNB)

In this section, we first revisit the original Probabilistic Neutral Bits (PNB) based attack on reduced round ChaCha and Salsa by Aumasson et al. [1].

The PNB set divides the key bits of the cipher into two halves. One contains the neutral n-bits for a particular number of rounds and the other half contains m-significant bits (m = number of key bits $-n$ bits) of the cipher, so here $m = 256 - n$. Note that, the neutral key bits are probabilistic in nature. Hence the attacker needs to run the experiments for certain number of times to find out the forward bias at the particular intermediate round (r) of the cipher as well as the PNB set. The authors have taken the sample of 2^{24} different nonce and counters for each of the 2^{10} keys in all the experiments carried out [1].

The attack is a meet-in-the-middle type of approach where the attacker initially finds the bits with respect to the median forward bias (ϵ_d^*) in the intermediate round. Then the ϵ_d^* is used to calculate the PNB set from the reduced last round (R) of the cipher.

Forward Biases (ϵ_d). Consider X and X' are two valid input states with an initial input difference ($\mathcal{ID}$) at the j-th bit of the i-th word i.e. $\Delta_{i,j}^{(0)} = 1$. The forward bias (ϵ_d) for a single key after r rounds can be calculated as follows:

$$\Pr(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_d), \tag{13}$$

where $\Delta_{p,q}^{(r)}$ is the output difference ($\mathcal{OD}$) after r-th round at q-th bit of the p-th word. By doing the same experiments for different keys, a set of ϵ_d's can be calculated. Now one can compute ϵ_d^* which is the median forward bias of the set of ϵ_d's. ϵ_d^* is used to find the PNB set and in the key recovery procedure.

PNB and the related biases (γ_k, γ). Suppose Z and Z' are the corresponding output keystream bits of the cipher after R rounds i.e. $Z = X \boxplus_{32} X^{(R)}$ and $Z' = X' \boxplus_{32} X'^{(R)}$. Now to check whether the k-th bit is PNB or not, at first we invert the k-th bit in both X and X' to make the initial states $\overline{X}$ and $\overline{X'}$. Then invert $Z \boxminus_{32} \overline{X}$ and $Z' \boxminus_{32} \overline{X'}$ for $R - r$ rounds. Suppose Y, Y' are the intermediate states after $R - r$ rounds. Now one can compute the neutrality measure of the k-th key bit (γ_k) by considering the following equation:

$$\Pr(\Delta_{p,q}^{(r)} = Y_{p,q} \oplus Y'_{p,q} | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \gamma_k). \tag{14}$$

The encrypted function should run for several random IVs to calculate the bias γ_k for the corresponding key bit k. Now, if $\gamma_k \geq \gamma$, where γ is a predefined high bias, then the k-th bit is included in the PNB set for the related bias γ. In [24] the authors have used 2^{26} random samples of nonce, counters to calculate the neutrality measurement for each key bit and finally generated the PNB set.

Reverse biases (ϵ_a) After recovering the PNB set of the cipher, one has to find the backward bias (ϵ_a) and the median of the backward bias (ϵ_a^*) related to the PNB set for the key recovery attack. Suppose Z and Z' are the keystream bits for the states X and X' respectively with $X' = X \oplus \Delta_{i,j}^{(0)}$. Consider that after setting the $\mathcal{ID}$ at $X_{i,j}^{(0)}$, the $\mathcal{OD}$ appears at $X_{i,j}^{(r)}$ in the r-th intermediate round. Now, generate two states $\hat{X}$ and $\hat{X}'$ by putting random values in the n-PNB bits and the original values in the m-significant key bit positions. Apply the inverse function on the states $Z \boxminus_{32} \hat{X}$ and $Z' \boxminus_{32} \hat{X}'$ for $R - r$ rounds from the R-th round. Suppose this yields the states $\hat{Y}$ and $\hat{Y}'$ and $\Gamma_{p,q}$ is the difference of $\hat{Y}, \hat{Y}'$ at $X_{p,q}^{(r)}$. Now calculate the following probability:

$$\Pr(\Delta_{p,q}^{(r)} = \Gamma_{p,q} | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_a). \tag{15}$$

Here the backward bias ϵ_a is calculated by varying the IVs randomly over $\mathbb{F}_2^{128}$ for a key. This key is also should be varied randomly from the keyspace $\mathbb{K}$ to generate a set of backward biases. The median ϵ_a^* of this set is used in the key recovery attack procedure.

Distinguisher Event for Key Recovery. For the key recovery attack, the event $\Gamma_{p,q} = 1$ is considered condition to the event $\Delta_{i,j}^{(0)} = 1$. If ϵ is the bias of this event, then one can write:

$$\Pr[\Gamma_{p,q} = 1 | \Delta_{i,j}^{(0)} = 1] = \frac{1}{2}(1 + \epsilon). \tag{16}$$

Using the standard probability rule, we have $\Pr(\Gamma_{p,q} = 1 | \Delta_{i,j}^{(0)} = 1) = \Pr(\Gamma_{p,q} = \Delta_{p,q}^{(r)} \wedge \Delta_{p,q}^{(r)} = 1 \mid \Delta_{i,j}^{(0)} = 1) + \Pr(\Gamma_{p,q} \neq \Delta_{p,q}^{(r)} \wedge \Delta_{p,q}^{(r)} = 0 \mid \Delta_{i,j}^{(0)} = 1)$. Under the assumption that given $\Delta_{i,j}^{(0)} = 1$, the events $\Delta_{p,q}^{(r)} = 1$ and $\Gamma_{p,q} = \Delta_{p,q}^{(r)}$ are statistically independent, we have $\frac{1}{2}(1+\epsilon) = \frac{1}{2}(1+\epsilon_d)\cdot(1+\epsilon_a) + \frac{1}{2}(1-\epsilon_d)\cdot(1-\epsilon_a)$, giving:

$$\epsilon = \epsilon_d \cdot \epsilon_a. \tag{17}$$

Therefore, if the independence assumptions hold in practice, then ϵ can be estimated as $\epsilon_d \cdot \epsilon_a$, as also noted in [1]. However, if the independence assumptions do not hold, then one has to empirically calculate ϵ by directly observing the distinguisher event $\Gamma_{p,q} = 1$ condition to $\Delta_{i,j}^{(0)} = 1$, over many iterations.

Data and Time Complexities of the attack (N, T). Let $P_{fa} = 2^{-\alpha}$ be the probability of false alarm (Type I error) and $P_{nd} = 0.0013$ be the probability of non-detection (Type II error). The sample size of the corresponding attack can be calculated by using the following formula [1]:

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1 - \epsilon^2}}{\epsilon} \right)^2. \tag{18}$$

As mentioned in [1], ϵ_d, ϵ_a and ϵ may have wide variability because of infeasibility of processing sufficient number of samples and so for the estimate to be robust, the medians ϵ_a^*, ϵ_d^* and ϵ^* of the above variables may be calculated by randomly varying the keys over the keyspace.

Let m denote the number of significant (non-PNB) key bits. The time complexity of the attack is given by [1]:

$$T = 2^m(N + 2^{256-m} \cdot P_{fa}) = 2^m \cdot N + 2^{256-\alpha}. \tag{19}$$

A correction of the above formula is suggested in [14] as follows:

$$T = 2^m \cdot N + 2^{256-\alpha} + 2^{256-m}. \tag{20}$$

Threshold for Key Recovery. Note that the distinguisher event $\Gamma_{p,q} = 1$ condition to $\Delta_{i,j}^{(0)} = 1$ has the probability $p = \frac{1}{2}(1 + \epsilon)$. Hence, for the correct key guess, the number of samples out of N that would satisfy the distinguisher event (we can say that this is the number of *votes* for the guessed key) would have a mean $\mu = Np$ and a standard deviation $\sigma = \sqrt{Np(1-p)}$. One may consider a suitable threshold Th (as a function of μ and σ) for the votes of the candidate keys, such that if the number of votes of a key guess is greater than Th, then it is included in the set of possible keys. According to the standard *3-sigma* rule of estimation, if one takes $Th = \mu - 3\sigma$, then theoretically more than 99% of the time the correct key should be included in this filtered set. By simple substitution, one can observe that:

$$Th = 0.5N(1 + \epsilon) - 1.5\sqrt{N(1 - \epsilon^2)}, \tag{21}$$

where ϵ may be replaced by $\epsilon_d \cdot \epsilon_a$, provided the independence assumption holds.

3 Our Construction of new PNB Set (PNBS)

Here in our proposed method, we use the union of the PNB sets which are generated from the suitable choice of the multiple $\mathcal{OD}$'s in the intermediate round corresponding to an $\mathcal{ID}$ in the initial round. The construction procedure is done in the following two steps.

3.1 Finding Output Differential Set

Suppose the input difference is given as $\Delta_{i,j}^{(0)}$, i.e., the j-th bit of the i-th word in the initial round. After a chosen intermediate round r, we will calculate all output bit positions $X_{p,q}^{(r)}$ where the difference $\Delta_{p,q}^{(r)}$ appears with a high forward bias ϵ_d^* with respect to a predefined threshold. For this, we randomly choose a key k and an initial vector IV. Now, depending upon key and IV, we construct the initial state matrix X of the cipher. We also construct another state matrix X' after giving the $\mathcal{ID}$ at $X_{i,j}^{(0)}$. Now we run both the states independently

for r rounds and calculate the output difference $\Delta_{p,q}^{(r)}$ at any arbitrary but fixed position $X_{p,q}^{(r)}$, with $0 \leq p \leq 15$, $0 \leq q \leq 31$ of the state. If $\Delta_{p,q}^{(r)} = 1$, then there is a difference in that position in the intermediate r-th round. For each randomly chosen key, by varying IV for $NumIV$ times, we calculate the probability of forward bias event (ϵ_d) by Eq. (13). Repeat this process for $NumKey$ times different keys and generate a set of ϵ_d values. Then after taking the median of these values, we compute ϵ_d^* for each output bit $X_{p,q}^{(r)}$ in the intermediate round.

Algorithm 1: FINDING_OUTPUT_DIFFERENTIAL_SET

Input: ϵ_d' : Threshold for forward bias; $\Delta_{i,j}^{(0)}$: $\mathcal{ID}$ at initial round.
Output: $\mathcal{B}$, a output differential set at the r-th round.

$\mathcal{B} \leftarrow \{\}$;
for *p =0 to 15, q =0 to 31* **do**
  $\epsilon_d \leftarrow \{\}$;
  for $k_{idx} = 0$ *to* $NumKey$ **do**
    $k \xleftarrow{\$} \{0,1\}^{256}$; `// Taking random key`
    $ctr = 0$;
    for $iv_{idx} = 0$ *to* $NumIV$ **do**
      $iv \xleftarrow{\$} \{0,1\}^{128}$;
      $X \leftarrow$ Initialize with k, iv;
      $X' \leftarrow$ Initialize with k, $iv \oplus \Delta_{i,j}^{(0)}$;
      $Z, Z' \leftarrow Enc^{(r)}(X), Enc^{(r)}(X')$; `// r round Encryption`
      $\Delta_{p,q}^{(r)} \leftarrow Z \oplus Z'$ at $X_{p,q}^{(r)}$;
      if $\Delta_{p,q}^{(r)} = 1$ **then**
        $ctr \leftarrow ctr + 1$;
    $Pr \leftarrow ctr/NumIV$; `// calculating the probability`
    $\epsilon_d \leftarrow \epsilon_d \cup \{2 \times (Pr - \frac{1}{2})\}$;
  $\epsilon_d^* =$ median of ϵ_d;
  if $|\epsilon_d^*| > \epsilon_d'$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\Delta_{p,q}^{(r)}\}$
return $\mathcal{B}$

For random states, ϵ_d^* should be approximately 0. If it is above a certain chosen threshold ϵ_d', then we call $\Delta_{p,q}^{(r)}$ as $\mathcal{OD}_l$ for some index number l, corresponding to the $\mathcal{ID}$ at $\Delta_{i,j}^{(0)}$. We include $\Delta_{p,q}^{(r)}$, where the high bias appears in an output differential set $\mathcal{B}$. To compute all the biases for ChaCha and Salsa in our experiment, we have considered $NumKey = 2^{10}$ and $NumIV = 2^{35}$ samples. The pseudocode of the process is given in Algorithm 1.

3.2 Finding the PNBS

In the previous works, a key bit is declared as PNB corresponding to a fixed $\mathcal{OD}$ at the r-th intermediate round, if the value of γ_k is greater than a certain predefined threshold γ. Here in our approach, we check whether or not the value of γ_k is greater than the threshold corresponding to any $\mathcal{OD}_t \in \mathcal{B}$. If it occurs, then we include the corresponding key bit k in the new PNB set, PNBS. The construction procedure of the new PNB set using our method is given in Algorithm 2. The schematic diagrams for the previously proposed and our new construction procedure are given in Figure 1 and Figure 2 respectively.

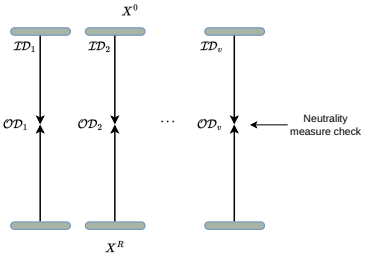

Fig. 1. Previous Construction Procedures of PNB sets

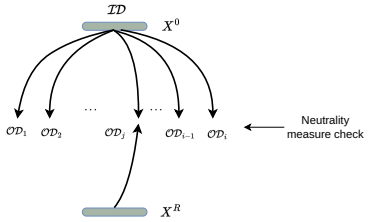

Fig. 2. Our Construction Procedure of PNB sets

Suppose for a fixed $\mathcal{ID}$ $\Delta_{i,j}^{(0)}$ there are v many $\mathcal{OD}$ positions, namely $\mathcal{OD}_1$, $\cdots$, $\mathcal{OD}_v$. According to Algorithm 2, if we take $\mathcal{B} = \{\mathcal{OD}_l : 1 \leq l \leq v\}$, the PNBS is the set P. Here P is the new PNB set corresponding to $\mathcal{ID}-\cup_{l=1}^{v}\mathcal{OD}_l$. It is easy to observe that, $P = \cup_{l=1}^{v} P_l$ where P_l is the PNB set corresponding to $\mathcal{ID}$-$\mathcal{OD}_l$. So, if the individual PNB sets P_l's are already pre-computed, then one may use the above result to construct the PNBS.

The experimental results are given in Table 21 (Appendix A) and Table 22 (Appendix B) for 7-round ChaCha and 8-round Salsa respectively. To select suitable PNB sets for PNBS, we need a collection of PNB sets with various cardinalities. Hence we adopt a smaller threshold of $\epsilon_d' = 0.005$ and $\gamma = 0.07$ compared to earlier studies in [1],[24]. The experiments for PNBS are performed by taking $NumIV = 2^{34}$ times random initial vectors. Here we have observed that, the best case appears with $|\mathcal{B}| = 47$ and $|\mathcal{B}| = 25$, after applying Algorithm 1 and giving the initial input difference at $\Delta_{13,13}^{(0)}$ and $\Delta_{7,31}^{(0)}$ for ChaCha and Salsa respectively. These $\mathcal{ID}$s are also considered in most of the previous works, so we take these specific $\mathcal{ID}$s in all our experiments. This results in 218 and 222 key bits after applying Algorithm 2 for the union of output bit differences, i.e., $\cup_{l=1}^{|\mathcal{B}|}\mathcal{OD}_l$, which are generated from the same $\mathcal{ID}$ for ChaCha and Salsa, respectively.

Note that the PNB property is a statistical property over IVs for a fixed key i.e. flipping a neutral bit should not significantly affect the output differ-

Algorithm 2: FINDING_PNBS

Input: $\Delta_{i,j}^{(0)}$: $\mathcal{ID}$ at initial round; $\mathcal{B} = \{\Delta_{p_1,q_1}^{(r)}, \cdots, \Delta_{p_v,q_v}^{(r)}\}$: Output difference set at r-th round; γ: threshold of the Neutrality measurement

Output: P: PNBS with respect to $\mathcal{B}$

$P \leftarrow \{\}$;
for $k = 0$ *to* 256 **do**
  // Each key bit
  $\gamma_{k_t} = 0$, for $t \in \mathcal{B}$;
  $key \xleftarrow{\$} \{0,1\}^{256}$;
  $ctr_t = 0$, for $t \in \mathcal{B}$;
  for $iv_{idx} = 0$ *to* $NumIV$ **do**
    $iv \xleftarrow{\$} \{0,1\}^{128}$;
    $iv' \leftarrow iv \oplus \Delta_{i,j}^{(0)}$; // Giving initial input difference
    $X \leftarrow$ Initialize state with key, iv;
    $X' \leftarrow$ Initialize state with key, iv';
    $\Delta^{(r)} \leftarrow Enc^{(r)}(X) \oplus Enc^{(r)}(X')$; // Storing the difference at r-th round
    $Z, Z' \leftarrow X + Enc^{(R)}(X), X' + Enc^{(R)}(X')$;
    $key' \leftarrow$ Flip the k-th bit in key;
    $\overline{X} \leftarrow$ Initialize state with key', iv;
    $\overline{X'} \leftarrow$ Initialize state with key', iv';
    $Y \leftarrow Inv^{(R-r)}(Z - \overline{X})$ using key', iv; // $Inv^{(R-r)} :=$ Inverse of $R-r$ rounds
    $Y' \leftarrow Inv^{(R-r)}(Z' - \overline{X'})$ using key', iv';
    $\Gamma \leftarrow Y \oplus Y'$;
    for $t = 1$ *to* v **do**
      if $\Delta_{p_t,q_t}^{(r)} = \Gamma_{p_t,q_t}$ **then**
        // forward difference equals with backward difference at $X_{p_t,q_t}^{(r)}$
        $ctr_t \leftarrow ctr_t + 1$;
  for $t = 1$ *to* v **do**
    $Pr \leftarrow ctr_t / NumIV$; // Calculating the probability
    $\gamma_{k_t} \leftarrow 2 \times (Pr - \frac{1}{2})$;
    if $\gamma_{k_t} > \gamma$ **then**
      $P \leftarrow P \cup \{k\}$;
      break;
Return P

ence regardless of which specific key is used. Our experiments confirm that the PNB sets obtained using a single random key with $NumIV = 2^{34}$ match those obtained using multiple IVs and keys ($NumIV = 2^{26}$ and $NumKey = 2^{10}$). In fact, repeating Algorithm 2 with many independent random keys, IVs and

computing the neutrality measure for each key bit, we find that the resulting PNB sets are identical to those reported in the paper. Furthermore, our derived PNB sets for single $\mathcal{ID}$-$\mathcal{OD}$ match those reported in prior works [1], [6], [13], [17], [22], [24], [25].

4 Our Key Recovery Method

In the previous attack procedures [1], [25], [24], [18] the whole keyspace is divided into two parts (shown in Figure 3). The key recovery attack in this setup has been discussed in Section 2.4. We perform divide and conquer method by partitioning the keyspace into multiple smaller parts by means of multiple $\mathcal{OD}$'s with respect to a single $\mathcal{ID}$ (shown in Figure 4).

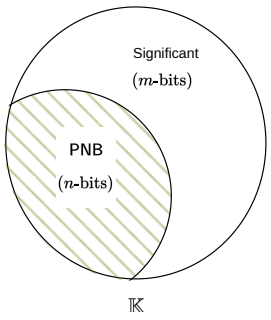

Fig. 3. Previous Key Recovery Attack

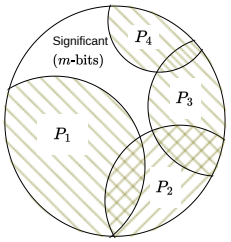

Fig. 4. New Key Recovery attack
$P_t \coloneqq$ PNB set corresponding to $\mathcal{OD}_t$
$m \coloneqq$ significant bits with respect to $\cup_{t=1}^{4} P_t$

4.1 Partitioning the Keyspace

We devise our new multi-step key recovery attack procedure based on PNBS with respect to $\mathcal{B}' \subset \mathcal{B}$ with $|\mathcal{B}'| = v'$ and we label the $\mathcal{OD} \in \mathcal{B}'$ as $\mathcal{OD}_t = \Delta^{(r)}_{p_t,q_t}$, $1 \leq t \leq v'$. This helps in dividing the keyspace into $v'+1$ parts, of which v' parts correspond to the PNB bits corresponding to the v' $\mathcal{OD}$'s and one part is the significant key bits. This, in turn, reduces the total key space into smaller sub-key spaces that can be efficiently searched in a step by step manner. The selection criteria for proper choice of $\mathcal{B}'$ is discussed below and v' is discussed in Section 5.1.

Choice of the set $\mathcal{B}'$. For a fixed v', the time complexity of the attack depends upon the cardinality of the PNB sets P_t, $t \in \{1, \cdots, v'\}$ with respect to $\mathcal{OD}_t$ $= \mathcal{OD}_l \in \mathcal{B}$, for some $l \in \{1, \cdots, v\}$ and S, the significant bits corresponding

to $\cup_{t=1}^{v'} P_t$. The time complexity becomes optimal if each PNB set P_t contains $256/(v'+1)$ bits. In order to avoid redundancy in search, we need to divide the keyspace into mutually exclusive partitions. The selection process of the set $\mathcal{B}'$ and the corresponding mutually exclusive versions P'_t of the PNB sets P_t, is given in Algorithm 3.

Algorithm 3: Selection of $\mathcal{B}'$

Input: $\{P_l : 1 \leq l \leq |\mathcal{B}|\}$: P_l is PNB set corresponding to $\mathcal{OD}_l \in \mathcal{B}$; v': A given number.
Output: $\mathcal{B}'$ and the corresponding mutually exclusive sets P'_t's.

$\mathcal{B}' \leftarrow \{\}$
for $t = 1$ *to* v' **do**
  Set $P'_t = P_l$ such that $|P_l|$ is nearest to $256/(v'+1)$
  $\mathcal{B}' \leftarrow \mathcal{B}' \cup \{\mathcal{OD}_l\}$
  $P_s = P_s \setminus P'_t,\ \forall\ s \leq |\mathcal{B}|$
Return $\mathcal{B}'$ and the corresponding mutually exclusive sets P'_t.

4.2 New Distinguisher Event

In all earlier differential cryptanalysis of ChaCha and Salsa, the key recovery is based on the distinguisher event $\Gamma_{p,q} = 1$ corresponding to a single $\mathcal{ID}$-$\mathcal{OD}$ condition to $\Delta^{(0)}_{i,j} = 1$ (given in Eq. (16)). In our approach, we consider single $\mathcal{ID}$ and union of v' many $\mathcal{OD}$s, with $|\mathcal{B}'| = v'$. The bit $\Gamma_{p,q}$ corresponding to the t-th $\mathcal{ID}$-$\mathcal{OD}_t$ is denoted by Γ_{p_t,q_t} for $1 \leq t \leq v'$. Our distinguisher event is that at least one of the Γ_{p_t,q_t}'s is 1 condition to $\Delta^{(0)}_{i,j} = 1$. The probability of our distinguisher $\left(\bigvee_{t=1}^{v'} \Gamma_{p_t,q_t} = 1 | \Delta^{(0)}_{i,j} = 1\right)$ is denoted by $\Pr_u$. This can be written as:

$$\Pr_u = \Pr\left(\bigvee_{t=1}^{v'} (\Gamma_{p_t,q_t} = 1) | \Delta^{(0)}_{i,j} = 1\right) = p_{rand}(1 + \epsilon_u), \tag{22}$$

where $p_{rand} = 1 - \frac{1}{2^{v'}}$ is the probability that any one of v' many uniformly distributed variables takes the value 1, and ϵ_u denotes the bias of the distinguisher. Later on (Eq. (23)), we will see how to find the expression of ϵ_u.

In the key recovery part of a PNB-based attack, the bias ϵ (discussed in Eq. (17)) plays the crucial role. As already explained in Section 2.4, one can approximate ϵ by the product of $\epsilon_d \cdot \epsilon_a$ under the independence assumption of the events $(\Delta^{(r)}_{p,q} = 1 | \Delta^{(0)}_{i,j} = 1)$ and $(\Gamma_{p,q} = \Delta^{(r)}_{p,q} | \Delta^{(0)}_{i,j} = 1)$ for a single $\mathcal{ID}$-$\mathcal{OD}$. In our distinguishing event, we have considered PNBS, the union of the PNB sets for multiple $\mathcal{OD}$ bits in an intermediate round. The forward bias of each individual $\mathcal{OD}$ bit can be computed using the same process used for single event. Consider

ϵ_{d_t} as the forward bias of the t-th event $\Gamma_{p_t,q_t} = 1$ corresponding to $\mathcal{ID}$-$\mathcal{OD}_t$, which can be derived from Eq. (13). Now, to calculate the backward bias, the question appears that, in which key bits one should give the random values so that the independent assumption still holds for our union event. For computation of ϵ_u, the independence of the events $\Gamma_{p_t,q_t} = 1$ for $1 \leq t \leq v'$, is achieved under the common setting of all the key bits $\cup_t P'_t$ being random. The experimentation on the toy versions (as will be demonstrated in Section 6) also validates this. Consider ϵ_{a_t} as the backward bias of the event $(\Delta^{(r)}_{p_t,q_t} = \Gamma_{p_t,q_t} | \Delta^{(0)}_{i,j} = 1)$, using Eq. (15) for each $\mathcal{ID}$-$\mathcal{OD}_t$.

Let $\Pr(\Gamma_{p_t,q_t} = 1 | \Delta^{(0)}_{i,j} = 1)$ be denoted by $\Pr_t = \frac{1}{2}(1 + \epsilon_t)$, where each bias ϵ_t can be approximated after putting random values in the underlying set $\cup_t P_t$. Observe that, $\Pr_u$ (given in Eq. (22)) can also be approximated as $1 - \prod_{t=1}^{v'}(1 - \Pr_t) = 1 - \frac{1}{2^{v'}} \prod_{t=1}^{v'}(1 - \epsilon_t)$. Hence the bias ϵ_u can be estimated as a function of the ϵ_{d_t} and ϵ_{a_t} values, $1 \leq t \leq v'$, as follows:

$$\epsilon_u \approx \frac{1 - \prod_{t=1}^{v'}(1 - \epsilon_{d_t} \cdot \epsilon_{a_t})}{2^{v'} - 1}. \tag{23}$$

Verification of Independence. We have verified the independence assumptions considered in the product of Eq. (23). In particular, we have examined the forward independence of ChaCha and Salsa (using Eq. (13)), as well as both forward and backward independence (using Eq. (13) and Eq. (15) after putting random values in $\cup_t P'_t$) across all the toy versions of ChaCha discussed in Section 6, with respect to the $\mathcal{OD}$ bits selected in this work. The complete results of these independence verification experiments are presented in Section 7. Such assumptions are common in many attack strategies (such as linear cryptanalysis [26], DFA [31]) in the literature. The assumption of statistical independence for the distinguishers is reasonable provided that all output masks are linearly independent. This approach follows prior established work, such as the work [7] on ChaCha and Salsa, which also relied on the assumption of statistical independence in the Bidirectional Linear Expansions (BLE) technique.

4.3 Data Complexity

Consider a collection of n data samples and an event A with $\Pr(A) = p$. Now, take a variable:

$$X_i = \begin{cases} 1: & \text{if } A \text{ occurs in the } i\text{-th sample with } 1 \leq i \leq n, \\ 0: & \text{otherwise.} \end{cases}$$

Observe that, $\Pr(X_i = 1) = p$ for any i-th sample. This follows each $X_i \sim$ Bernoulli(p) and $\sum_{i=1}^{n} X_i \sim$ Binomial(n, p) as we have n samples. Now if each X_i is independent and identically distributed and n is sufficiently large, then by Central Limit Theorem (CLT) [32] we get:

$$\sum_{i=1}^{n} X_i \sim \mathcal{N}(np, np(1-p)),$$

where $\mathcal{N}$ is the normal distribution. Here the null hypothesis is $H_0 : p = p_0$, where p_0 is the success probability of the Bernoulli random variable under H_0 and the alternative hypothesis is $H_1 : p = p_0(1+\epsilon)$ with $\epsilon > 0$. Here our objective is to distinguish the null hypothesis (H_0) from the alternative hypothesis (H_1). In other words, we have to find a threshold $c \in [np_0, np_0(1+\epsilon)]$ such that $\Pr(\sum_{i=1}^{n} X_i \leq c \mid H_0) \leq 2^{-\alpha}$ and $\Pr(\sum_{i=1}^{n} X_i > c \mid H_1) \leq \beta$, where $2^{-\alpha}$ and β are the probabilities of Type I and Type II errors respectively.

Now for such a c to exist, the condition, $np_0(1+\epsilon) - np_0 > \kappa_1\sigma_1 + \kappa_2\sigma_2$ has to be satisfied, where $\sigma_1^2 = np_0(1+\epsilon)\cdot(1-p_0(1+\epsilon))$, $\sigma_2^2 = np_0(1-p_0)$, $\Phi(\kappa_1) = 2^{-\alpha}$, and $\Phi(\kappa_2) = 1-\beta$. This gives,

$$n > \frac{(\kappa_1\sqrt{1-p_0} + \kappa_2\sqrt{(1+\epsilon)\cdot(1-p_0(1+\epsilon))})^2}{p_0\epsilon^2}. \tag{24}$$

For $P_{fa} = 2^{-\alpha}$, $\kappa_1 \approx \sqrt{\alpha \log 4}$. To detect the bias of the event with probability more than 99% we take $\beta = 0.0013$, which gives $\kappa_2 = 3$.

Here for our PNB based key recovery attack, the success probability of the Bernoulli random variable under the null hypothesis (H_0) is p_{rand} (defined in Section 4.2) and ϵ_u (given in Eq. (23)) is the bias of our new distinguisher event. Hence from Eq. (24), the required data complexity for our new distinguisher event becomes,

$$N_u \approx \left(\frac{\sqrt{\left(\frac{1-p_{rand}}{p_{rand}}\right)\alpha \log 4} + 3\sqrt{\frac{(1+\epsilon_u)(1-p_{rand}(1+\epsilon_u))}{p_{rand}}}}{\epsilon_u} \right)^2. \tag{25}$$

We will use this Eq. (25) for the data complexity of our key recovery attack through out the paper.

4.4 Modified Threshold for Key Recovery

Since we have proposed a new distinguisher event for key recovery, the threshold for the key votes need to be generalized accordingly. The analysis is similar to Section 2.4, except that Eq. (21) has to be modified with $\mu = N_u \Pr_u$ and $\sigma = \sqrt{N_u \Pr_u(1-\Pr_u)}$. With necessary substitutions, the new threshold for union of t PNB sets becomes:

$$Th_u^{(t)} = N_u p_{rand}(1+\epsilon_u) \pm 3\sqrt{N_u p_{rand}(1+\epsilon_u)(1-p_{rand}(1+\epsilon_u))}. \tag{26}$$

Through out the paper, by $Th_u^{(t)}$, we mean $Th_u^{(t)}$ is a function of u and t both.

4.5 Attack Procedure

Let Z and Z' be the keystream and the faulty keystream respectively. We require N_u number of such pairs (Z, Z') with random IVs, after giving the $\mathcal{ID}$ at $\Delta_{i,j}^{(0)}$, where N_u is given in Eq. (25). Suppose $Th_u^{(t)}$ is the threshold for the union of t PNB sets. Next, we discuss the steps of the key recovery process.

Step 1. Suppose m is the number of significant bits corresponding to the output difference set $\mathcal{B}'$. We put random values in $\cup_{t=1}^{v'} P_t'$ and all possible values in m bit positions and construct two initial states X and X'. After subtracting the constructed initial states from the generated keystream Z and the faulty keystream Z', we invert for $R-r$ rounds and calculate the number of times the event $\bigvee_{t=1}^{v'} \Gamma_{p_t,q_t} = 1$ occurs. If the number satisfies the condition in Step 28 or Step 30 of Algorithm 4, then we include the guessed key as a possible candidate key for the m significant bits. If the probability of false alarm is $2^{-\alpha}$, then the keyspace of m bits is reduced from 2^m to $2^{m-\alpha}$.

Step 2. In this step, we consider the output differential set $\mathcal{B}'$ with $v'-1$ number of $\mathcal{OD}$s and the corresponding PNBS $\cup_{t=1}^{v'-1} P_t'$. Hence in this case, for each $2^{m-\alpha}$ possible keys, we put all possible values in $P_{v'}'$ bits and random values in other key bits. Now we calculate the number of times the event $\bigvee_{t=1}^{v'-1} \Gamma_{p_t,q_t} = 1$ occurs. If its greater than $Th_u^{(v'-1)}$ then we include it in the possible key set. Similarly as above, the search space reduces to $2^{m-\alpha+|P_{v'}'|-\alpha}$.

Step 3. Repeat the above Step 2 for the union of $v'-2$ to 1 number of PNB sets.

Therefore using our attack strategy, the keyspace is divided in to $v'+1$ parts and we recover the probable key bits of each part by using the corresponding thresholds at each time. The key recovery attack procedure is discussed in Algorithm 4.

In the following, we discuss the step by step reduction of key space using the above procedure with an example. Suppose we have $v'=4$ $\mathcal{OD}$ bits and the corresponding PNB sets are P_1, P_2, P_3, P_4. Now, the corresponding mutually exclusive PNB sets become $P_1' = P_1$, $P_2' = P_2 \setminus P_1$, $P_3' = P_3 \setminus (P_1 \cup P_2)$ and $P_4' = P_4 \setminus (P_1 \cup P_2 \cup P_3)$. Let $P = \cup_{t=1}^{4} P_t' = \cup_{t=1}^{4} P_t$, m be the number of significant bits (i.e $m = |\text{entire key bits} \setminus P|$) and D as the distinguisher. First, we give random values to the PNB bits in P bits and filter the m significant bits using the distinguisher of all 4 $\mathcal{OD}$ events. Next, we give random values to $P_1' \cup P_2' \cup P_3'$ bits and filter the P_4' key bits using the distinguisher of 3 $\mathcal{OD}$ events. Then, we give random values to $P_1' \cup P_2'$ bits and filter the P_3' key bits using the distinguisher of 2 $\mathcal{OD}$ events. After this, we give random values to P_1' and filter the P_2' key bits using the distinguisher of 1 $\mathcal{OD}$ event. Finally, we try all $2^{|P_1'|}$ possible values of P_1' bits. The key recovery steps for the above example are

Algorithm 4: KEY_RECOVERY

Input: Output differential set $\mathcal{B}' : \{\mathcal{OD}_t = \Delta^{(r)}_{p_t,q_t} : 1 \le t \le v'\}$; Threshold: $Th^{(t)}_u$, Mutually exclusive PNB sets: P'_t, Data complexity: $N^{(t)}_u$ for union of t $\mathcal{OD}$s, $1 \le t \le v'$; List of (Keystream, Faulty keystream, IV) tuples: (Z, Z', iv) of size $N^{(v')}_u$; $\mathcal{ID}$ at initial round: $\Delta^{(0)}_{i,j}$.

Output: A set of possible keys $\mathcal{K}$

$\mathcal{K} \leftarrow \{(0, \cdots, 0)\}$; // Initializing with zero key
for $t = v'$ *down to 1* **do**
  $P \leftarrow \cup_{l=1}^{t} P'_l$; // Union of t PNB sets
  if $t = v'$ **then**
    $S \leftarrow$ Set of all key bits $\setminus \bigcup_{l=1}^{v'} P'_l$;
  else
    $S \leftarrow P'_{t+1}$;
  $\mathcal{K}_{new} \leftarrow \{\}$;
  $p_{rand} \leftarrow 1 - \frac{1}{2^t}$;
  for $k \in \mathcal{K}$ **do**
    for k_{sig} *= 0 to* $2^{|S|} - 1$ **do**
      $k_{tmp} \leftarrow (0, \cdots, 0)$;
      Set the value of S bit positions of k_{tmp} to k_{sig};
      $k_{tmp} \leftarrow k_{tmp} \mid k$; // generate a guessed key by logical OR
      $ctr \leftarrow 0$;
      for $l = 0$ *to* $N^{(t)}_u - 1$ **do**
        $k_{test} \leftarrow k_{tmp}$; // Copy candidate key for sample l
        $rand \xleftarrow{\$} \{0,1\}^P$;
        Set the value of P bit positions of k_{test} to $rand$;
        $iv' \leftarrow iv[l] \oplus \Delta^{(0)}_{i,j}$; // faulty IV
        $Y \leftarrow Inv^{(R-r)}(Z[l],\ k_{test},\ iv[l])$;
        $Y' \leftarrow Inv^{(R-r)}(Z'[l],\ k_{test},\ iv')$;
        $\Gamma \leftarrow Y \oplus Y'$;
        for $\tau = 0$ *to* $t - 1$ // $t \leftarrow v'$ down to 1, as per line 2 **do**
          if $\Gamma_{p_\tau,q_\tau} = 1$ **then**
            $ctr \leftarrow ctr + 1$;
            break;
      if $((Th^{(t)}_u < N^{(t)}_u \cdot p_{rand}) \wedge (Th^{(t)}_u > ctr))$ // For $\epsilon_u < 0$ **then**
        $\mathcal{K}_{new} \leftarrow \mathcal{K}_{new} \cup \{k_{tmp}\}$;
      if $((Th^{(t)}_u > N^{(t)}_u \cdot p_{rand}) \wedge (Th^{(t)}_u < ctr))$ // For $\epsilon_u > 0$ **then**
        $\mathcal{K}_{new} \leftarrow \mathcal{K}_{new} \cup \{k_{tmp}\}$;
  $\mathcal{K} \leftarrow \mathcal{K}_{new}$;
return $\mathcal{K}$

summarized in Table 3. As an alternative explanation, at each step, the backward bias is computed with all remaining unrecovered PNB bits set to random. This ensures independence between forward and backward phases used at that step. As the attack progresses and more key bits become known, the number of $\mathcal{OD}$-events in the distinguisher decreases accordingly, but the randomized set also shrinks, improving the backward bias for the remaining events.

Step	**Known Bits**	**Randomized Bits**	**Guessed Bits**	**#$\mathcal{OD}$ events unioned to form D**
1	–	P	S	All 4
2	S	$P'_1 \cup P'_2 \cup P'_3$	P'_4	3
3	$S + P'_4$	$P'_1 \cup P'_2$	P'_3	2
4	$S + P'_4 + P'_3$	P'_1	P'_2	1
5	$S + P'_4 + P'_3 + P'_2$	–	P'_1	–

Table 3. Step by step reduction of key space with $v' = 4$ in key recovery attack

4.6 Time Complexity of the Proposed Attack

In Section 2.4, the time complexity of Eq. (19) is based on partitioning the 256-bit keyspace into two parts consists of m significant bits and $256 - m$ PNB bits. In our approach we partition the keyspace into $v' + 1$ parts of which one is the m many significant key bits, and the other v' parts are the mutually exclusive P'_t's. Since for each of the 2^m possible assignment of the significant key bits we test the distinguisher event in N_u samples, and then sequentially search the partitions P'_t's exhaustively, the final complexity can be written as:

$$T_u \approx 2^m \left(N_u + P_{fa} \left(\sum_{t=1}^{v'} 2^{|P'_t|} \right) \right). \tag{27}$$

However, as Eq. (19) is corrected in Eq. (20) by adding a term corresponding to the entire non-significant key-space, we can also refine T_u by adding v' more correction terms corresponding to the v' PNB partitions, as follows.

$$T_u \approx 2^m \left(N + P_{fa} \left(\sum_{t=1}^{v'} 2^{|P'_t|} \right) \right) + \sum_{t=1}^{v'} 2^{|P'_t|}. \tag{28}$$

A derivation of the time complexity for the proposed attack (Eq. (28)) is provided in Appendix G.

Observe that our Algorithm 3 outputs the mutually exclusive versions (P') of the PNB sets (P) in a certain order. Then the elimination in Step 2 is performed

in the reverse order in the key recovery attack procedure. One may wonder whether an order of choice of mutually exclusive PNB sets different from that in Algorithm 3 would yield a better complexity. However, it can be easily verified from Eq. (28) and Eq. (25) that the alternation of the order does not effect the final time and data complexity of the attack. It has also been verified in ToyChaCha and the results are given in Table 23 (Appendix F).

Remark. It is worth noting that all our above formulations are generalization of (single $\mathcal{ID}$, single $\mathcal{OD}$)-based attack to (single $\mathcal{ID}$, multiple $\mathcal{OD}$) scenario. In particular, with $v' = 1$, Eq. (22), (23), (25), (27), (28) and (26) reduce to Eq. (16), (17), (18), (19), (20) and (21) respectively.

5 Our Attack Parameters and Results

All the previous works ([1], [24], [25], [14], [13], [22], [2], [14] etc.) have used the median ϵ_d^* of ϵ_d and median ϵ_a^* of ϵ_a to compute the complexities of their attacks. In [10], [28], [38], it is shown that, for fluctuating data, median is more "robust" than mean of some data samples. As discussed in Section 4.2, here we have to put random values in PNBS to calculate each backward bias in the union event. Now if the underlying set is $\cup_t P'_t$, then the backward bias (ϵ_a) of one $\mathcal{OD}$ would be affected due to the random values in the PNB sets of the other $\mathcal{OD}$s. This may cause alteration of signs in the calculated backward bias, resulting in a very small magnitude of the median of ϵ_a's. For example, for the set $\{-0.8, -0.75, -0.7, 0.01, 0.6, 0.65, 0.8\}$, of the biases, the median of the set of values becomes 0.01, which is very low comparative to the other bias values in the set. To mitigate this, we consider median of the absolute values to calculate the scenario. With this, the required number of samples becomes $\frac{1}{0.5\times((|\epsilon_a|)^*)^2}$, which leads to 2^{38} samples to calculate the backward bias of one $\mathcal{OD}$ with underlying set $\cup_t P'_t$ in our case. In all our experiments, we have considered 2^{45} samples (2^{35} random IVs for each of the 2^{10} random keys), which is more than sufficient to verify ϵ_a's for the individual $\mathcal{ID}$-$\mathcal{OD}$. Now, due to the alteration of sign, instead of finding the median and using the values of ϵ_{d_t}, ϵ_{a_t} in Eq. (23), we will calculate ϵ_u for each $NumKey$ values and take the median ϵ_u^*. This is used in Eq. (25) to compute the final theoretical data complexity of the attack for the main and the toy cipher given in this paper. The individual biases at $\mathcal{ID}$-$\mathcal{OD}_t$'s are computed using 2^{35} random IVs and 2^{10} random keys. Here we have chosen $\alpha = 60$, so that after each step the key space reduces to 1. The experiments are performed in a HPC with 13 nodes, each having the specification Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with 40 CPUs.

We have given the results of ChaCha in Sections 5.2, 5.4 and Salsa in Sections 5.3, 5.5 in the following. In [17], the authors first presented an attack on 8.5-round Salsa, with the time complexity of $2^{245.84}$. Here, using our proposed method, we have computed the results for 8.5-round Salsa with respect to PNBS, but the time complexity of the key recovery attack becomes exhaustive for the cipher. However, for 7.5-round ChaCha and 8.5-round Salsa, we have only used

our key recovery procedure using PNBS, and not used any additional condition related to $\mathcal{HW}$ in the intermediate round or strategies which are applied in [39],[17]. Though our results on 7.5-round ChaCha beat the result of Flórez-Gutiérrez and Todo [21], the result reported in [39] remains better than both [21] and our current work. All the results of 7-round ChaCha and 8-round Salsa are given in Table 1.

5.1 Choice of Number v' of $\mathcal{OD}$s/PNB Sets

As discussed earlier, the time complexity would decrease if we partition the key space into more parts, i.e., if we increase v'. However, with increase in v', the approximation error between the theoretical and the practical values of ϵ_u is likely to increase, as more events are assumed to be independent. Therefore, we need to consider a trade-off.

Let $\epsilon_{u_{Pr}}$ and $\epsilon_{u_{Th}}$ denote the practical (stated in Eq. (22)) and the theoretical (stated in Eq. (23)) values of ϵ_u respectively. For a given $\epsilon_{u_{Th}}$, $\epsilon_{u_{Pr}}$, the percentage of relative approximation error for the PNBS (union of v' number of PNB sets) is calculated by $\delta = \frac{|\epsilon_{u_{Th}} - \epsilon_{u_{Pr}}|}{|\epsilon_{u_{Pr}}|} \times 100$. The relative error percentage is likely to increase with the number of PNB sets used to construct new PNBS, i.e. with v'. For the 32-bit toy version of ChaCha, namely, ToyChaCha [13] (described in Section 6), we observe δ as 12.62%, 30.70%, 63.19% corresponding to $v' =$ 2, 3, 4 respectively. Here we have taken $\mathcal{ID} = \Delta^{(0)}_{13,0}$ and selected v' $\mathcal{OD}$s from the output differential set $\mathcal{B}$, with $NumIV = 2^{24}$ and $NumKey = 2^8$. For each v', the relative error percentage δ is estimated by averaging over 100 different tuples of v' $\mathcal{OD}$s.

The approximation relies on the independence assumptions (discussed in Section 4.2), which would hold more effectively, if the cipher has more randomness. Since the randomness of a cipher increases with the key size as well as with the number of rounds, the randomness achieved by a 4 times smaller cipher of 3.5-rounds is much less than that of a 4 times larger cipher of 7 or 8 round. Accordingly, the approximation-error in the distinguishing-event is expected to have a smaller growth-rate in the original larger cipher. To justify this, we have constructed two new toy ciphers with larger key size, namely ToyChaCha48 and ToyChaCha64. In ToyChaCha48, the ARX (addition, rotation, and XOR) operations are defined over 12-bit words, with the four key words duplicated across the two middle rows in the same manner as the 128-bit key variant of ChaCha (Details in Section 6.3). ToyChaCha64 is similar to 32-bit ToyChaCha proposed in [13], except that the middle two rows contains distinct 64-bit keys at the initial state matrix. This is similar to the modification that is done for 256-bit version with respect to 128-bit version of ChaCha. For ToyChaCha48, the relative error percentage δ becomes 10.01%, 20.05%, 35.91% and for ToyChaCha64, the relative error percentage δ becomes 10.14%, 15.74%, 31.24% for $v' = 2, 3, 4$ respectively. These results are summarized in Table 4. In case of ToyChaCha48 and ToyChaCha64, we have considered a single $\mathcal{ID} = \Delta^{(0)}_{12,5}$ and $\mathcal{ID} = \Delta^{(0)}_{13,0}$ re-

spectively, and for each v', we have estimated δ by averaging over 100 different tuples of v' $\mathcal{OD}$s, with $NumIV = 2^{24}$ and $NumKey = 2^8$.

Primitive	$v' = 2$	$v' = 3$	$v' = 4$
ToyChaCha	12.62%	30.70%	63.19%
ToyChaCha48	10.01%	20.05%	35.91%
ToyChaCha64	10.14%	15.74%	31.24%

Table 4. Error percentages across different toy primitives and values of v'

ToyChaCha48 consistently shows lower relative error percentage δ than ToyChaCha and ToyChaCha64 consistently shows lower δ than ToyChaCha48 for every value of v' (see Table 4). This confirms the hypothesis that increased state size (more randomness) improves the accuracy of the independence approximation. The trend is monotonically decreasing: as we move from 32-bit to 48-bit to 64-bit to the full 256-bit cipher, the approximation error is expected to decrease further. This supports our choice of $v' = 4$ for the full cipher, where the error is expected to be well below the 31.24% observed for the 64-bit version.

Two further reasons behind our choice of $v' = 4$ is as follows. For $v' = 4$, the target-cardinality of the non-intersecting PNB-sets P' is $256/(4+1) \approx 51$; but for $v' = 3$, it is $256/(3+1) \approx 64$, which is much larger than 51 indicating larger complexity. Moreover, the PNB-sets are different, and the variance of the actual observed cardinalities for $v' = 3$ is much larger than that of $v' = 4$.

So, our natural choice of v' for 256 and 128-bit versions of ChaCha and Salsa becomes 4 with which we perform the attacks and report the results, that are summarized in Table 1. Additionally, to observe the expected improvement from $v' = 3$ to 4, we have also computed the results corresponding to $v' = 3$ for all the ciphers, which are given in Appendix E.

5.2 Results on 256-bit version of ChaCha

We have discussed the result for 7 and 7.5 rounds using our key recovery attack procedure. 7-round ChaCha is exploited using differential approach and 7.5-round ChaCha is exploited using differential and differential-linear approaches, which we will see in the following. Here we have taken $\mathcal{ID} = \Delta^{(0)}_{13,13}$, as it is taken in most of the works to attack the cipher. We find the best result using our attack approach for $\epsilon_d = 0.005$ and $\gamma = 0.07$ for 7-round ChaCha, 7.5-round ChaCha and 8-round Salsa. We will discuss our results depending upon these parameters in the following.

7-round. We have observed output differential set $\mathcal{B}$ at $r = 3$ with $|\mathcal{B}| = 45$ for $\epsilon'_d = 0.005$. This gives $\mathcal{B}' = \{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{11,0}, \Delta^{(3)}_{2,0}\}$ after using Algorithm 3 and $v' = 4$. Here each P'_t is having cardinality $\{54, 40, 47, 16\}$ for $\mathcal{OD}_t \in \mathcal{B}'$.

The mutually exclusive PNB sets P'_t corresponding to each $\mathcal{OD}_t \in \mathcal{B}'$ for $1 \leq t \leq 4$ at the t-th step of the key recovery attack of 7-round ChaCha, is given in Appendix C. This results $\epsilon_u^* = 1.3331 \times 10^{-10}$. This gives the data and time complexities $2^{68.90}$ and $2^{167.90}$ respectively for 7 round of the cipher.

7.5-round. We have applied our attack procedure on 7.5 rounds of ChaCha cipher using differential and differential-linear approaches, which we discuss below. Here we have used approximately 2^{45} (specifically, $2^{44.32}$ samples with $NumIV = 2^{35}$ and $NumKey = 2^{9.32}$) for each approach to calculate the backward bias corresponding to each $\mathcal{OD}$.

Differential Approach. For PNBS consisting of union of 4, the forward bias ϵ_d is calculated after $r = 3$ round, and the backward bias ϵ_a is calculated after applying inverse round function for 4.5 round from $R = 7.5$ round. We get the best result for $\epsilon'_d = 0.005$ and $\gamma = 0.02$. The forward differential set $\mathcal{B}$ is having cardinality 45 for $\epsilon'_d = 0.005$. This gives $\mathcal{B}' = \{\Delta^{(3)}_{11,0}, \Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{2,0}\}$ corresponding to $v' = 4$. We partition the keyspace depending upon this $\mathcal{B}'$ and get $|P'_t| = \{38, 13, 11, 10\}$ (as discussed in Subsection 4.1). Here the final data and time complexity becomes $2^{53.41}$, $2^{237.41}$ respectively using our key recovery attack procedure.

Differential-linear Approach. In this case, to get the forward bias at $r = 3.5$, we use the differential-linear approach proposed in [6]. Consider the case $\epsilon_L = 1$ here. Since the linear approximation holds with probability 1, here the complexity evaluation directly follows from the differential framework given in the previous section. For this case, in our attack procedure, we have used $\mathcal{ID} = \Delta^{(0)}_{13,13}$ and found the $\mathcal{OD}$ bits, $\mathcal{B} = \{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{2,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{11,0}, \Delta^{(3)}_{12,0}, \Delta^{(3)}_{15,0}\}$ corresponding to $\epsilon'_d > 0.005$, which are of the form $\Delta^{(3)}_{p,0}$ for some p-th word. These bits can be expressed as:

$$\left.\begin{aligned}
\Delta^{(3)}_{0,0} &= \Delta^{(3.5)}_{0,0} \oplus \Delta^{(3.5)}_{5,12} \oplus \Delta^{(3.5)}_{10,0},\\
\Delta^{(3)}_{2,0} &= \Delta^{(3.5)}_{2,0} \oplus \Delta^{(3.5)}_{7,12} \oplus \Delta^{(3.5)}_{8,0},\\
\Delta^{(3)}_{3,0} &= \Delta^{(3.5)}_{3,0} \oplus \Delta^{(3.5)}_{4,12} \oplus \Delta^{(3.5)}_{9,0},\\
\Delta^{(3)}_{11,0} &= \Delta^{(3.5)}_{11,0} \oplus \Delta^{(3.5)}_{12,0},\\
\Delta^{(3)}_{12,0} &= \Delta^{(3.5)}_{12,16} \oplus \Delta^{(3.5)}_{1,0},\\
\Delta^{(3)}_{15,0} &= \Delta^{(3.5)}_{15,16} \oplus \Delta^{(3.5)}_{0,0},
\end{aligned}\right\} \quad (29)$$

$$\left.\begin{aligned}
\Delta^{(3)}_{0,0} &= \Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{5,7} \oplus \Delta^{(4)}_{5,19} \oplus \Delta^{(4)}_{10,12} \oplus \Delta^{(4)}_{15,0},\\
\Delta^{(3)}_{2,0} &= \Delta^{(4)}_{2,0} \oplus \Delta^{(4)}_{7,7} \oplus \Delta^{(4)}_{7,19} \oplus \Delta^{(4)}_{8,12} \oplus \Delta^{(4)}_{13,0},\\
\Delta^{(3)}_{3,0} &= \Delta^{(4)}_{3,0} \oplus \Delta^{(4)}_{4,7} \oplus \Delta^{(4)}_{4,19} \oplus \Delta^{(4)}_{9,12} \oplus \Delta^{(4)}_{14,0},\\
\Delta^{(3)}_{11,0} &= \Delta^{(4)}_{12,0} \oplus \Delta^{(4)}_{11,0} \oplus \Delta^{(4)}_{12,8} \oplus \Delta^{(4)}_{1,0},\\
\Delta^{(3)}_{12,0} &= \Delta^{(4)}_{12,24} \oplus \Delta^{(4)}_{1,16} \oplus \Delta^{(4)}_{1,0} \oplus \Delta^{(4)}_{11,0} \oplus \Delta^{(4)}_{6,7},\\
\Delta^{(3)}_{15,0} &= \Delta^{(4)}_{15,24} \oplus \Delta^{(4)}_{0,16} \oplus \Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{10,0} \oplus \Delta^{(4)}_{5,7}.
\end{aligned}\right\} \quad (30)$$

The Eq. (29) are the linear combinations of the $\mathcal{OD}$ bits at 3.5 round with respect to the $\mathcal{OD}$ bits from $r = 3$ round. We have noticed that, these equations generate the same union of the PNB sets as it was generated for three rounds. As we get the best results in these specific intermediate bits only for the differential approach so we will get the best results for this case. Hence, the data and the time complexity will be same with the data and the time complexity of the differential approach for 7.5-round ChaCha, which is $2^{53.41}$ and $2^{237.41}$ respectively. Therefore, using this approach, the differential attack can be viewed as

a differential-linear attack, where one can improve the intermediate round, but the time complexity remains same.

We have also used the Eq. (30) to compute forward bias at $r = 4$ round corresponding to the $\mathcal{OD}$ bits from $r = 3$ round. As discussed in the previous part, here also we will get the same $\mathcal{OD}$ bits and the PNBS also becomes same. Hence, the complexities will also remain the same with the above discussed part.

5.3 Results on 256-bit version of Salsa

Here we have chosen $\mathcal{ID} = \Delta^{(0)}_{7,31}$ in the initial round and the proposed approach in section 3 is used to perform the key recovery attack on 8 rounds of the cipher. The output differential set $\mathcal{B}$ is formed at the intermediate round $r = 4$. This is used to find the PNB sets after the reduced final round $R = 8$. Here also we find the best time complexity for $\epsilon'_d = 0.005$ and $\gamma = 0.07$. This gives $|\mathcal{B}| = 25$ for $\epsilon'_d = 0.005$, and leads to $\mathcal{B}' = \{\Delta^{(4)}_{1,23}, \Delta^{(4)}_{1,4}, \Delta^{(4)}_{1,14}, \Delta^{(4)}_{6,15}\}$ for union of $v' = 4$ PNB sets. Here the mutually exclusive sets P'_t are having cardinality $\{42, 36, 30, 26\}$ corresponding to $\mathcal{OD}_t \in \mathcal{B}'$. The mutually exclusive PNB sets P'_t corresponding to each $\mathcal{OD}_t \in \mathcal{B}'$ for $1 \leq t \leq 4$ at the t-th step of the key recovery attack of 8-round Salsa, is given in Appendix D. This provides the final ϵ^*_u, data, time complexities equals -1.7114×10^{-9}, $2^{61.54}$ and $2^{183.54}$ respectively for $v' = 4$.

5.4 Results on 128-bit version of ChaCha

Here we discuss our results of 6.5-round ChaCha for 128-bit version. We have observed output differential set $\mathcal{B}$ at $r = 3$ with $|\mathcal{B}| = 46$ for $\epsilon'_d = 0.005$. This gives $\mathcal{B}' = \{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{10,16}, \Delta^{(3)}_{11,0}\}$ after using Algorithm 3 and $v' = 4$. Here each P'_t is having cardinality $\{25, 21, 26, 8\}$ for $\mathcal{OD}_t \in \mathcal{B}'$. This results $\epsilon^*_u = 2.3671 \times 10^{-10}$ and gives the data and time complexities $2^{67.24}$ and $2^{115.24}$ respectively for 6.5 round of the cipher.

5.5 Results on 128-bit version of Salsa

Here we discuss our results of 7.5-round Salsa for 128-bit version. We have observed output differential set $\mathcal{B}$ at $r = 3$ with $|\mathcal{B}| = 25$ for $\epsilon'_d = 0.005$. This gives $\mathcal{B}' = \{\Delta^{(4)}_{1,4}, \Delta^{(4)}_{1,23}, \Delta^{(4)}_{6,4}, \Delta^{(4)}_{6,26}\}$ after using Algorithm 3 and $v' = 4$. Here each P'_t is having cardinality $\{26, 26, 21, 15\}$ for $\mathcal{OD}_t \in \mathcal{B}'$. This results $\epsilon^*_u = -5.0693 \times 10^{-9}$ and gives the data and time complexities $2^{58.40}$ and $2^{98.40}$ respectively for 7.5 round of the cipher.

6 Practical Verification of Our Attack on Toy Versions of ChaCha

Since the first PNB attack proposed by Aumasson et al. [1], there have been many works that have used approaches like differential, linear, differential-linear etc.

Using the updated strategies, the time complexities of the attack got reduced by approximately 2^{50} and 2^{40} times for ChaCha and Salsa respectively over almost two decades. Though the practical validation of the attack procedures still can not be done on either cipher.

Dey et al. [13] introduced a reduced toy version of ChaCha (ToyChaCha), to experimentally validate the attacks proposed by Aumasson et al. [1] and Maitra et al. [24], alongside their own attack strategy. In this work, we introduce two additional toy variants, namely ToyChaCha48 and ToyChaCha64. Similar to 128-bit ChaCha, ToyChaCha48 duplicates four key words across both middle rows, while scaling the ARX operations (addition, rotation, and XOR) down to 12-bit words. In contrast, ToyChaCha64 uses the same word size as 32-bit ToyChaCha [13], but features eight distinct key words across the middle rows for a 64-bit key, mirroring the relation between 256-bit and 128-bit ChaCha. In the following sections, we evaluate our attack across all three toy variants (32-bit, 48-bit, and 64-bit). A comparative performance summary between our attack and existing techniques on ToyChaCha is provided in Table 7.

6.1 Common Setup for All Toy Versions

For each toy version, we apply the three-step key recovery procedure described in Section 4.5. For ToyChaCha, we execute the attack with $(r, R) = (2, 3.5)$ rounds using both the union of 2 and 3 $\mathcal{OD}$s. For ToyChaCha48, we set $(r, R) = (2, 3.5)$ to evaluate both Aumasson's PNB attack procedure and our proposed method. For ToyChaCha64, we set $(r, R) = (2, 4)$ to perform the attack using the union of 2 $\mathcal{OD}$s and find the success rate using union of 3 $\mathcal{OD}$s. To determine the data and time complexities across all toy cipher variants, we fix $\alpha = 30$. All practical experiments across each toy version are conducted with $NumIV = 2^{24}$ and $NumKey = 2^8$, evaluating the success rate over 1000 independent random key trials. Here we experimentally evaluate the success rate of the key recovery attacks across each toy variant. Specifically, for a randomly selected key, we execute the attack procedure detailed in Section 4.5 for 1000 times and count the number of instances in which the key successfully satisfies the threshold condition.

Note that across the key recovery attack summary tables (Tables 8, 9, 10, 11, 12, and 13) for the toy variants, the backward bias of $\mathcal{OD}_t$ ($\epsilon_a^*[t]$) is computed after assigning random values to the remaining non-neutral key bits in $\bigcup_{i=1}^{k} P_i$, where $k = 3, 2, 1$ in Steps 1,2, and 3 of Algorithm 4.5, respectively. Furthermore, under the 'Key Space Reduction' column, the 'Before' and 'After' entries of the tables (Tables 8, 9, 10, 11, and12) quantify the significant key candidate space w.r.t. $\bigcup_{i=1}^{k} P_i$ immediately before and after executing that step.

6.2 Attack on ToyChaCha

The structure. The toy cipher, ToyChaCha [13] adopts the same structural design as the 128-bit version of ChaCha. Here, each word consists of 8 bits,

which makes the total state size equal to 128. The key contains 4 words each of 8 bits, which is copied in the next row, following the same pattern of 128-bit version of ChaCha. The four constants of the initial round are chosen as $c_0 = \texttt{0x65}$, $c_1 = \texttt{0x6e}$, $c_2 = \texttt{0x32}$, $c_3 = \texttt{0x74}$. The quarter round of the round function takes 4 words (a, b, c, d) as input and outputs (a', b', c', d'), as follows.

$$\begin{aligned} a' &= a \boxplus_8 b, & d' &= ((d \oplus a') \lll_8 4), \\ c' &= c \boxplus_8 d', & b' &= ((b \oplus c') \lll_8 3), \\ a'' &= a' \boxplus_8 b', & d'' &= ((d' \oplus a'') \lll_8 2), \\ c'' &= c' \boxplus_8 d'', & b'' &= ((b' \oplus c'') \lll_8 1). \end{aligned} \tag{31}$$

Here $\boxplus_8$ and $\lll_8$ denotes the addition and left circular rotation over the field $\mathbb{F}_2^8$ respectively.

Our Attack using Union of 2 $\mathcal{OD}$s. To validate our key recovery attack, we select the input difference $\mathcal{ID} = \Delta_{13,0}^{(0)}$ alongside specific $\mathcal{OD}$s in intermediate round $r = 2$, as detailed in Table 5. In the following we elaborate a single row in detail.

$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	$\lvert P_1 \cup P_2 \rvert$	$\epsilon_{u_{Th}}^*$	$\epsilon_{u_{Pr}}^*$	$N_{u_{Th}}$	$N_{u_{Pr}}$	$T_{u_{Th}}$	$T_{u_{Pr}}$	Success Rate (%)
$\Delta_{13,0}^{(0)}$	$\Delta_{1,5}^{(2)}$	-0.8765	0.002488	21	0.2462	0.2462	$2^{8.52}$	$2^{8.52}$	$2^{19.68}$	$2^{19.68}$	96.2
	$\Delta_{2,0}^{(2)}$	0.7401	1.0								
	$\Delta_{1,6}^{(2)}$	0.9159	0.004039	21	0.2467	0.2467	$2^{8.51}$	$2^{8.51}$	$2^{19.76}$	$2^{19.76}$	96.56
	$\Delta_{2,0}^{(2)}$	0.7392	1.0								
	$\Delta_{1,6}^{(2)}$	0.9159	0.006117	22	0.003195	0.00377	$2^{21.47}$	$2^{20.99}$	$2^{31.47}$	$2^{31.45}$	90.24
	$\Delta_{3,4}^{(2)}$	0.6378	0.006418								
	$\Delta_{2,7}^{(2)}$	-0.2577	0.03725	21	0.2456	0.2461	$2^{10.36}$	$2^{10.37}$	$2^{22.55}$	$2^{22.55}$	98.39
	$\Delta_{2,0}^{(2)}$	0.7392	1.0								
	$\Delta_{14,5}^{(2)}$	-0.2841	0.007886	21	0.2462	0.2459	$2^{10.07}$	$2^{10.06}$	$2^{20.82}$	$2^{20.82}$	99.1
	$\Delta_{2,0}^{(2)}$	0.7392	1.0								

Table 5. Experimental results on ToyChaCha corresponding to union of 2 $\mathcal{OD}$s.

Here we set the input difference $\mathcal{ID} = \Delta_{13,0}^{(0)}$ and the output differential set $\mathcal{B}' = \{\Delta_{1,5}^{(2)}, \Delta_{2,0}^{(2)}\}$. Under thresholds $\epsilon_d' = 0.2$ and $\gamma = 0.7$, we obtain forward biases $\epsilon_d^* \in \{-0.8765, 0.7401\}$ and the corresponding PNB sets:

$$\begin{aligned} P_1 &= \{0, 1, 2, 3, 4, 5, 6, 7, 29, 30, 31\}, \\ P_2 &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, \\ &\quad 12, 13, 14, 15, 23, 28, 29, 30, 31\}, \end{aligned}$$

with sizes $|P_1| = 11$, $|P_2| = 21$, and $|P_1 \cup P_2| = 21$, respectively. In Step 1 of the key recovery attack (detailed in Section 4.5), we utilize $\mathcal{B}'$ and the combined PNB set $P_1 \cup P_2$, which contains $m = 11$ significant bits. In Step 2, we select the pair $(\mathcal{ID}, \mathcal{OD}) = (\Delta^{(0)}_{13,0}, \Delta^{(2)}_{1,5})$ and the PNB set P_1 to recover 10 bits in $P_2 \setminus P_1$. The details about the attack in each of the steps, are given in Table 6. Overall, the key recovery attack achieves a total data complexity of $2^{8.52}$ and an empirical time complexity of $2^{19.68}$, which is equal to the theoretical time complexity computed by Eq. (28). The experimental success rate of the key recovery attack becomes 96.2%.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$	Key Space	
							Before	After
$\Delta^{(0)}_{13,0}$	Step 1 ($t=2$)	$\Delta^{(2)}_{1,5}$	-0.8765	0.002488	366 ($\approx 2^{8.52}$)	328 ($\approx 2^{8.36}$)	2^{11}	1
		$\Delta^{(2)}_{2,0}$	0.7401	1				
	Step 2 ($t=1$)	$\Delta^{(2)}_{1,5}$	-0.8765	1	81 ($\approx 2^{6.34}$)	12 ($\approx 2^{3.52}$)	1×2^{10}	2^1

Table 6. Our key recovery attack of ToyChaCha using union of 2 $\mathcal{OD}$s

In [13], the authors showed keyspace reduced to $2^{24.65}$ using Aumasson's work and the time complexity using their own attack reduced to $2^{16.15}$. Here the original key remains in 99.81% and 99.71% cases for [1] and [13] respectively. The comparison of our work with all the other works in ToyChaCha is given in Table 7.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	$\epsilon^*_{u_{Th}}$	$\epsilon^*_{u_{Pr}}$	N_u	$T_{u_{Pr}}$	$T_{u_{Th}}$	Success Rate (%)	Refs.
$\Delta^{(0)}_{13,0}$	Step 1	$\Delta^{(2)}_{1,5}$	-0.8765	0.002488	0.2462	0.2462	$2^{8.52}$	$2^{19.68}$	$2^{19.68}$	96.2	This work
		$\Delta^{(2)}_{2,0}$	0.7401	1							
	Step 2	$\Delta^{(2)}_{1,5}$	-0.8765	1	-0.8765	-0.8765					
$\Delta^{(0)}_{13,0}$	—	$\Delta^{(2)}_{1,6}$	0.9167	0.377	0.343	0.3652	$2^{8.562}$	$2^{23.60}$	$2^{24.67}$	99.81	[13]
$\Delta^{(0)}_{13,0}$	—	$\Delta^{(2)}_{1,6}$	0.98	0.49	0.4802	0.3814	$2^{7.531}$	$2^{23.50}$	$2^{24.65}$	99.71	
$\Delta^{(0)}_{13,0}$	—	$\Delta^{(2)}_{1,6}$	0.91	1	0.91	—	$2^{6.56}$	$2^{15.10}$	$2^{16.15}$	—	
$\Delta^{(0)}_{14,0}$	—	$\Delta^{(2)}_{2,6}$									
$\Delta^{(0)}_{15,0}$	—	$\Delta^{(2)}_{3,6}$									

Table 7. Summary of key recovery attack procedures applied on ToyChaCha.

Our Attack using union of 3 $\mathcal{OD}$s. For the key recovery attack using union of three $\mathcal{OD}$s, we have set $\mathcal{ID} = \Delta^{(0)}_{13,0}$ and $\mathcal{B}' = \{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{9,6}, \Delta^{(2)}_{12,2}\}$. Under

$\epsilon'_d = 0.2$ and $\gamma = 0.7$, the resulting PNB sets are:

$$P_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 29, 30, 31\},$$
$$P_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15, 17, 18, 30, 31\},$$
$$P_3 = \{6, 7, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31\},$$

with $|P_1| = 11, |P_2| = 17, |P_3| = 12, |P_1 \cup P_2| = 18, |P_1 \cup P_2 \cup P_3| = 25$. The attack parameter across the three steps discussed in Table 8. Overall, this 3-difference variant of the key recovery attack achieves a total data complexity of $2^{20.13}$ and an empirical time complexity of $2^{28.03}$, with an experimental success rate of 92%.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$	Key Space	
							Before	After
$\Delta_{13,0}^{(0)}$	Step 1 ($t=3$)	$\Delta_{1,5}^{(2)}$	-0.8765	0.002578	1148089 ($\approx 2^{20.13}$)	1002293 ($\approx 2^{19.93}$)	2^7	$2^{5.94}$
		$\Delta_{9,6}^{(2)}$	-0.2612	0.07815				
		$\Delta_{12,2}^{(2)}$	-0.2803	0.002558				
	Step 2 ($t=2$)	$\Delta_{1,5}^{(2)}$	-0.8765	0.03221	16149 ($\approx 2^{13.98}$)	11757 ($\approx 2^{13.52}$)	$2^7 \times 2^{5.94}$	$2^{6.17}$
		$\Delta_{9,6}^{(2)}$	-0.2612	0.3804				
	Step 3 ($t=1$)	$\Delta_{1,5}^{(2)}$	-0.8765	1	81 ($\approx 2^{6.34}$)	12 ($\approx 2^{3.52}$)	$2^7 \times 2^{6.17}$	$2^{0.75}$

Table 8. Our key recovery attack of ToyChaCha using union of 3 $\mathcal{OD}$s

6.3 Attack on ToyChaCha48

The Structure. In this work, we introduce ToyChaCha48, a toy cipher variant whose structure is analogously to the 128-bit version of ChaCha. Each word consists of 12 bits, which makes the total state size equal to 192. The key contains 4 words each of 12 bits, which is copied in the next row, following the same pattern of 128-bit version of ChaCha. The four constants of the initial round are chosen as $c_0 = \texttt{0x865}$, $c_1 = \texttt{0x46e}$, $c_2 = \texttt{0xd32}$, $c_3 = \texttt{0x574}$. The quarter round of the round function takes 4 words (a, b, c, d) as input and outputs (a', b', c', d'), as follows.

$$\begin{aligned} a' &= a \boxplus_{12} b, & d' &= ((d \oplus a') \lll_{12} 6), \\ c' &= c \boxplus_{12} d', & b' &= ((b \oplus c') \lll_{12} 4), \\ a'' &= a' \boxplus_{12} b', & d'' &= ((d' \oplus a'') \lll_{12} 2), \\ c'' &= c' \boxplus_{12} d'', & b'' &= ((b' \oplus c'') \lll_{12} 1). \end{aligned} \tag{32}$$

Here $\boxplus_{12}$ and $\lll_{12}$ denotes the addition and left circular rotation over the field $\mathbb{F}_2^{12}$ respectively.

Verification of Aumasson's PNB Attack [1]. We evaluate the attack on 3.5 rounds of ToyChaCha48 using the difference pair $(\mathcal{ID}, \mathcal{OD}) = (\Delta^{(0)}_{12,0}, \Delta^{(2)}_{0,2})$. Over $NumIV = 2^{24}$ and $NumKey = 2^8$, this pair yields a 2-round intermediate forward bias of $\epsilon^*_d = -0.7376$. For a threshold $\gamma = 0.4$, we obtain the PNB set:

$$P = \{2, 7, 8, 11, 19, 20, 21, 22, 23, 26, 27, 28, 29, 30, 31, 32, 33, \\ 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47\},$$

which contains $|P| = 31$ key bit positions. The details about the forward and backward biases in the key recovery attack, are given in Table 9. Setting $\alpha = 30$, Eqn. (18), (21), and (20) yield a data complexity of $N = 546 \approx 2^{9.09}$, a threshold of $Th = 198 \approx 2^{7.63}$, and a theoretical time complexity of $T = 2^{31.05}$. The empirical time complexity of the attack becomes $2^{31.075}$ with an overall success rate of 96%.

$\mathcal{ID}$	$\mathcal{OD}$	ϵ^*_d	ϵ^*_a	N	Th	Key Space	
						Before	After
$\Delta^{(0)}_{12,0}$	$\Delta^{(2)}_{0,2}$	-0.7376	0.5339	546 ($\approx 2^{9.09}$)	198 ($\approx 2^{7.63}$)	2^{17}	1

Table 9. Key recovery using Aumasson's PNB attack procedure on ToyChaCha48

Our Key Recovery Attack using union of 2 $\mathcal{OD}$s. To verify our key recovery attack, we set the input difference $\mathcal{ID} = \Delta^{(0)}_{12,5}$ and the output differential set $\mathcal{B}' = \{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{1,1}\}$. Under thresholds $\epsilon'_d = 0.2$ and $\gamma = 0.7$, we obtain forward biases $\epsilon^*_d \in \{-0.4063, -0.5008\}$ and the corresponding PNB sets:

$$P_1 = \{12, 22, 23, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, \\ 40, 41, 42, 43, 44, 45, 46, 47\},$$
$$P_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 18, 22, 23, 30, 31, 32, \\ 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47\},$$

with sizes $|P_1| = 22$, $|P_2| = 32$, and $|P_1 \cup P_2| = 35$, respectively. In Step 1 of the key recovery attack (detailed in Section 4.5), we utilize $\mathcal{B}'$ and the combined PNB set $P_1 \cup P_2$, which contains $m = 13$ significant bits. In Step 2, we select the pair $(\mathcal{ID}, \mathcal{OD}) = (\Delta^{(0)}_{12,5}, \Delta^{(2)}_{0,5})$ and the PNB set P_1 to recover 13 bits in $P_2 \setminus P_1$. The details about the attack in each of the steps, are given in Table 10. Overall, the key recovery attack achieves a total data complexity of $2^{16.38}$ and an empirical time complexity of $2^{29.45}$, with an experimental success rate of 99.8%.

Our Key Recovery Attack using union of 3 $\mathcal{OD}$s. We also verified the attack on ToyChaCha48 using a union of three output differences. Here, we set

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$	Key Space	
							Before	After
$\Delta_{12,5}^{(0)}$	Step 1 ($t=2$)	$\Delta_{0,5}^{(2)}$	-0.4063	0.002427	85480 ($\approx 2^{16.38}$)	63294 ($\approx 2^{15.95}$)	2^{13}	$2^{1.61}$
		$\Delta_{1,1}^{(2)}$	-0.5008	0.1103				
	Step 2 ($t=1$)	$\Delta_{0,5}^{(2)}$	-0.4063	0.9022	635 ($\approx 2^{9.31}$)	237 ($\approx 2^{7.89}$)	$2^{13} \times 2^{1.61}$	$2^{3.14}$

Table 10. Our key recovery attack of ToyChaCha48 using union of 2 $\mathcal{OD}$s

$\mathcal{ID} = \Delta_{12,5}^{(0)}$ and $\mathcal{B}' = \{\Delta_{0,5}^{(2)}, \Delta_{12,1}^{(2)}, \Delta_{13,9}^{(2)}\}$. Under $\epsilon_d' = 0.2$ and $\gamma = 0.7$, the resulting PNB sets are:

$$
\begin{aligned}
P_1 = \{&12, 22, 23, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,\\
&40, 41, 42, 43, 44, 45, 46, 47\},\\
P_2 = \{&1, 2, 6, 7, 8, 9, 10, 11, 17, 23, 25, 31, 32, 33, 34, 35,\\
&36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47\},\\
P_3 = \{&0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 21, 25,\\
&26, 39, 45, 46, 47\},
\end{aligned}
$$

with $|P_1| = 22, |P_2| = 28, |P_3| = 21, |P_1 \cup P_2| = 32, |P_1 \cup P_2 \cup P_3| = 40$. The forward, backward biases and other details related to the key recovery attack in each of the step, is given in Table 11. Overall, the 3-difference variant of the key recovery attack achieves a total data complexity of $2^{24.79}$ and an empirical time complexity of $2^{32.79}$, with an experimental success rate of 95%.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$	Key Space	
							Before	After
$\Delta_{12,5}^{(0)}$	Step 1 ($t=3$)	$\Delta_{0,5}^{(2)}$	-0.4063	0.00042	29055232 ($\approx 2^{24.79}$)	25411832 ($\approx 2^{24.6}$)	2^8	$2^{3.08}$
		$\Delta_{12,1}^{(2)}$	-0.927	0.0044				
		$\Delta_{13,9}^{(2)}$	-0.2509	0.0013				
	Step 2 ($t=2$)	$\Delta_{0,5}^{(2)}$	-0.4063	0.0082	76902 ($\approx 2^{16.23}$)	56902 ($\approx 2^{15.8}$)	$2^8 \times 2^{3.08}$	$2^{3.413}$
		$\Delta_{12,1}^{(2)}$	-0.927	0.0602				
	Step 3 ($t=1$)	$\Delta_{0,5}^{(2)}$	-0.4063	0.9022	635 ($\approx 2^{9.31}$)	236 ($\approx 2^{7.89}$)	$2^{10} \times 2^{3.413}$	$2^{2.8}$

Table 11. Our key recovery attack of ToyChaCha48 using union of 3 $\mathcal{OD}$s

6.4 Attack on ToyChaCha64

Our Key Recovery Attack using union of 2 $\mathcal{OD}$s. We experimentally verify the key recovery attack on 4 rounds of the cipher. We set the input difference $\mathcal{ID} = \Delta_{13,0}^{(0)}$ and the output difference set $\mathcal{B}' = \{\Delta_{1,4}^{(2)}, \Delta_{2,0}^{(2)}\}$. Under the threshold pairs $(\epsilon_d', \gamma) = (0.2, 0.4)$ for $\mathcal{OD} = \Delta_{1,4}^{(2)}$ and $(0.2, 0.7)$ for $\mathcal{OD} = \Delta_{2,0}^{(2)}$, we obtain

forward biases $\epsilon_d^* \in \{-0.7632, 0.7392\}$ and the corresponding PNB sets:

$$
\begin{aligned}
P_1 = \{&0, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15, 16, 17, 24, 29, 30, 31, 36,\\
&37, 38, 39, 47, 63\},\\
P_2 = \{&1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 21, 22, 23, 28, 29, 30,\\
&31, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 51, 52, 53, 54, 55, 63\}.
\end{aligned}
$$

with cardinalities $|P_1| = 24$, $|P_2| = 43$, and $|P_1 \cup P_2| = 45$, respectively. The forward, backward biases and other details related to the key recovery attack in each of the step, is given in Table 12. Overall, the 4-round attack achieves a total data complexity of $2^{13.20}$ and an empirical time complexity of $2^{32.20}$, with a success rate of 99%.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$	Key Space	
							Before	After
$\Delta_{13,0}^{(0)}$	Step 1 ($t=2$)	$\Delta_{1,4}^{(2)}$	-0.7632	6.85×10^{-5}	9450 ($\approx 2^{13.20}$)	7359 ($\approx 2^{12.85}$)	2^{19}	$2^{1.32}$
		$\Delta_{2,0}^{(2)}$	0.7392	0.2233				
	Step 2 ($t=1$)	$\Delta_{1,4}^{(2)}$	-0.7632	0.3150	1516 ($\approx 2^{10.57}$)	632 ($\approx 2^{9.30}$)	$2^{21} \times 2^{1.32}$	$2^{0.91}$

Table 12. Our key recovery attack of ToyChaCha64 using union of 2 $\mathcal{OD}$s

Success Rate of Our Key Recovery Attack using union of 3 $\mathcal{OD}$s. We have also calculated the success rate of our key recovery attack using a union of three output differences. Here, we set $\mathcal{ID} = \Delta_{13,0}^{(0)}$ and $\mathcal{B}' = \{\Delta_{1,5}^{(2)}, \Delta_{3,4}^{(2)}, \Delta_{12,0}^{(2)}\}$. Under $\epsilon_d' = 0.2$ and $\gamma = 0.7$, the resulting PNB sets are:

$$
\begin{aligned}
P_1 = \{&0, 1, 2, 3, 4, 5, 6, 7, 13, 14, 30, 31, 37, 38, 39\},\\
P_2 = \{&0, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 28, 47,\\
&52, 53, 54, 55, 63\},\\
P_3 = \{&0, 1, 5, 6, 7, 16, 17, 23, 24, 25, 26, 27, 28, 29, 30, 31,\\
&32, 33, 34, 35, 36, 37, 38, 39, 44, 45, 46, 47, 55, 56, 62, 63\}.
\end{aligned}
$$

with $|P_1| = 15, |P_2| = 19, |P_3| = 32, |P_1 \cup P_2| = 31, |P_1 \cup P_2 \cup P_3| = 46$. The forward, backward biases and other details in each of the step, is given in Table 13. Overall, the key recovery attack using union of 3 $\mathcal{OD}$s achieves an experimental success rate of 88% evaluated over 1000 random key trials.

6.5 Summary of Key Recovery Attacks on Toy Versions

We have added the summary of the practically verified key recovery attacks on each of the toy versions in Table 14.

$\mathcal{ID}$	Step	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	N_u	$Th_u^{(t)}$
$\Delta_{13,0}^{(0)}$	Step 1 ($t=3$)	$\Delta_{1,5}^{(2)}$	-0.8628	3.60×10^{-5}	134691741 ($\approx 2^{27.01}$)	117880026 ($\approx 2^{26.81}$)
		$\Delta_{3,4}^{(2)}$	0.6383	-5.76×10^{-5}		
		$\Delta_{12,0}^{(2)}$	0.6573	0.003480		
	Step 2 ($t=2$)	$\Delta_{1,5}^{(2)}$	-0.8628	0.001439	41309610 ($\approx 2^{25.30}$)	31000155 ($\approx 2^{24.89}$)
		$\Delta_{3,4}^{(2)}$	0.6383	0.005926		
	Step 3 ($t=1$)	$\Delta_{1,5}^{(2)}$	-0.8628	0.8233	145 ($\approx 2^{7.18}$)	34 ($\approx 2^{5.08}$)

Table 13. Success rate of our key recovery attack on ToyChaCha64 using union of 3 $\mathcal{OD}$s

Cipher	Union of Attack Variant	Data Complexity (N_u)	Empirical Time Complexity	Success Rate	Ref. Table
ToyChaCha	2 $\mathcal{OD}$s	$2^{8.52}$	$2^{19.68}$	96.2%	Table 6
	3 $\mathcal{OD}$s	$2^{20.13}$	$2^{28.03}$	92%	Table 8
ToyChaCha48	2 $\mathcal{OD}$s	$2^{16.38}$	$2^{29.45}$	99.8%	Table 10
	3 $\mathcal{OD}$s	$2^{24.79}$	$2^{32.79}$	95%	Table 11
ToyChaCha64	2 $\mathcal{OD}$s	$2^{13.20}$	$2^{32.20}$	99%	Table 12

Table 14. Summary of key recovery attacks across all toy cipher versions

To evaluate the effectiveness of our approach, we compare the complexities of the single $\mathcal{ID}$-$\mathcal{OD}$ components used in our attack against those computed using Aumasson et al.'s technique [1]. The attack parameters across all toy variants are maintained identically to those in our key recovery attack (detailed in Sections 6.2, 6.3, and 6.4). The full parameter sets and biases for the single $\mathcal{ID}$-$\mathcal{OD}$s are provided in Table 15. Our analysis highlights a key trade-off that increasing the number of intermediate $\mathcal{OD}$ bits reduces the resulting backward bias for our attack, as random values must be introduced across the union of PNB sets (see Section 4.2). Consequently, for the ciphers with smaller state sizes, our approach may not consistently outperform the traditional single $\mathcal{ID}$-$\mathcal{OD}$ attack [1]. However, as state sizes scale up, our method progressively gains the performance advantage.

Cipher	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d'	γ	$\lvert P_1 \rvert$	ϵ_d^*	ϵ_a^*	ϵ^*	N_u	T_u
ToyChaCha	$\Delta_{13,0}^{(0)}$	$\Delta_{1,5}^{(2)}$	0.2	0.7	11	-0.8765	1.0	-0.8765	$2^{6.34}$	$2^{27.34}$
	$\Delta_{13,0}^{(0)}$	$\Delta_{2,0}^{(2)}$	0.2	0.7	21	0.7401	1.0	0.7401	$2^{7.03}$	$2^{21.17}$
	$\Delta_{13,0}^{(0)}$	$\Delta_{9,6}^{(2)}$	0.2	0.7	17	-0.2612	0.7681	-0.2006	$2^{11.10}$	$2^{26.10}$
	$\Delta_{13,0}^{(0)}$	$\Delta_{12,2}^{(2)}$	0.2	0.7	12	-0.2803	1.0	-0.2803	$2^{10.11}$	$2^{30.11}$
ToyChaCha48	$\Delta_{12,5}^{(0)}$	$\Delta_{0,5}^{(2)}$	0.2	0.7	22	-0.4063	0.9022	-0.3666	$2^{9.32}$	$2^{35.31}$
	$\Delta_{12,5}^{(0)}$	$\Delta_{1,1}^{(2)}$	0.2	0.7	32	-0.5008	0.8828	-0.4421	$2^{8.74}$	$2^{32.01}$
	$\Delta_{12,5}^{(0)}$	$\Delta_{12,1}^{(2)}$	0.2	0.7	28	-0.9270	0.6815	-0.6318	$2^{7.59}$	$2^{35.31}$
	$\Delta_{12,5}^{(0)}$	$\Delta_{13,9}^{(2)}$	0.2	0.7	21	-0.2509	0.6296	-0.1578	$2^{11.79}$	$2^{38.79}$
ToyChaCha64	$\Delta_{13,0}^{(0)}$	$\Delta_{1,4}^{(2)}$	0.2	0.7	24	-0.7632	0.3159	-0.2411	$2^{10.56}$	$2^{50.56}$
	$\Delta_{13,0}^{(0)}$	$\Delta_{2,0}^{(2)}$	0.2	0.7	43	0.7392	0.8806	0.6510	$2^{7.49}$	$2^{43.00}$

Table 15. Parameters in single $\mathcal{ID}$-$\mathcal{OD}$ based PNB attack across the toy variants

This behavior is clearly observable across the toy variants in Table 16:

- **ToyChaCha:** The time complexity of our attack for a union of 2 $\mathcal{OD}$s is less than that of single $\mathcal{ID}$-$\mathcal{OD}$ attack with the technique of [1] by a factor of $2^{1.49}$. Notably, for a union of 3 $\mathcal{OD}$s, the specific pair $(\mathcal{ID}, \mathcal{OD}) = (\Delta^{(0)}_{13,0}, \Delta^{(2)}_{9,6})$ outperforms our union of PNB attack.
- **ToyChaCha48:** Similarly, for unions of 2 and 3 $\mathcal{OD}$s, the time complexities of our attacks become less than by a factor of $2^{2.56}$ and $2^{2.52}$ respectively than the single $\mathcal{ID}$-$\mathcal{OD}$ based attack.
- **ToyChaCha64:** Further, for a union of 2 $\mathcal{OD}$s, our attack outperforms the single $\mathcal{ID}$-$\mathcal{OD}$ attack by a factor of $2^{10.80}$.

The comparison between the complexities of the single $\mathcal{ID}$-$\mathcal{OD}$ attack and our proposed method in the toy variants is given in Table 16.

Cipher	Attack procedure	$\mathcal{ID}$	$\mathcal{B}'$	N_u	T_u
ToyChaCha	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{1,5}$	$2^{6.34}$	$2^{27.34}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{2,0}$	$2^{7.03}$	$2^{21.17}$
	Ours	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{2,0}\}$	$\mathbf{2^{8.52}}$	$\mathbf{2^{19.68}}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{1,5}$	$2^{6.34}$	$2^{27.34}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{9,6}$	$2^{11.10}$	$2^{26.10}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{12,2}$	$2^{10.11}$	$2^{30.11}$
	Ours	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{9,6}, \Delta^{(2)}_{12,2}\}$	$\mathbf{2^{20.13}}$	$\mathbf{2^{28.03}}$
ToyChaCha48	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{12,5}$	$\Delta^{(2)}_{0,5}$	$2^{9.32}$	$2^{35.31}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{12,5}$	$\Delta^{(2)}_{1,1}$	$2^{8.74}$	$2^{32.01}$
	Ours	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{1,1}\}$	$\mathbf{2^{16.38}}$	$\mathbf{2^{29.45}}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{12,5}$	$\Delta^{(2)}_{0,5}$	$2^{9.32}$	$2^{35.31}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{12,5}$	$\Delta^{(2)}_{12,1}$	$2^{7.59}$	$2^{35.31}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{12,5}$	$\Delta^{(2)}_{13,9}$	$2^{11.79}$	$2^{38.79}$
	Ours	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{12,1}, \Delta^{(2)}_{13,9}\}$	$\mathbf{2^{24.79}}$	$\mathbf{2^{32.79}}$
ToyChaCha64	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{1,4}$	$2^{10.56}$	$2^{50.56}$
	Single $\mathcal{ID}$-$\mathcal{OD}$	$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{2,0}$	$2^{7.49}$	$2^{43.00}$
	Ours	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,4}, \Delta^{(2)}_{2,0}\}$	$\mathbf{2^{13.20}}$	$\mathbf{2^{32.20}}$

Table 16. Comparison of our attack with single $\mathcal{ID}$-$\mathcal{OD}$ based attack in toy variants

These findings provide strong justification for selecting $v' = 4$ for the 128-bit and 256-bit ChaCha and Salsa.

7 Verification of Independence of Selected $\mathcal{OD}$s in Forward and Backward Directions

To check the independence between our selected forward output difference events $(\Delta^{(r)}_{p_t,q_t} = 1 | \Delta^{(0)}_{i,j} = 1)$, $1 \leq t \leq v' = 4$ (given in Section 5.2-5.5), we compute

the marginal probabilities $\Pr_t = \Pr(\Delta^{(r)}_{p_t,q_t} = 1 | \Delta^{(0)}_{i,j} = 1)$ with $NumIV = 2^{26}$ and $NumKey = 2^{10}$, and compare their product $\Pr_{prod} = \prod_{t=1}^{v'} \Pr_t$ with the joint probability $\Pr_{joint} = \Pr(\wedge_{t=1}^{v'}(\Delta^{(r)}_{p_t,q_t} = 1) | \Delta^{(0)}_{i,j} = 1)$. The results are given in Table 17. We observe that the precentage of the relative gaps (calculated by $(\frac{|\Pr_{joint} - \Pr_{prod}|}{\Pr_{joint}}) \times 100$) between $\Pr_{prod}$ and $\Pr_{joint}$ for the full-size ciphers are all below 2%.

Cipher	$\mathcal{ID}$	$\mathcal{B}'$	$\Pr_{joint}$	$\Pr_1 \cdot \Pr_2 \cdot \Pr_3 \cdot \Pr_4$	**Relative gap**
ChaCha 256-bit	$\Delta^{(0)}_{13,13}$	$\{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{11,0}, \Delta^{(3)}_{2,0}\}$	0.05747	0.05749	0.004%
ChaCha 128-bit	$\Delta^{(0)}_{13,13}$	$\{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{10,16}, \Delta^{(3)}_{11,0}\}$	0.0567	0.0569	0.36%
Salsa 256-bit	$\Delta^{(0)}_{7,31}$	$\{\Delta^{(4)}_{1,23}, \Delta^{(4)}_{1,4}, \Delta^{(4)}_{1,14}, \Delta^{(4)}_{6,15}\}$	0.0494	0.0486	1.62%
Salsa 128-bit	$\Delta^{(0)}_{7,31}$	$\{\Delta^{(4)}_{1,4}, \Delta^{(4)}_{1,23}, \Delta^{(4)}_{6,4}, \Delta^{(4)}_{6,26}\}$	0.04745	0.04718	0.57%

Table 17. Relative gap in forward bias of ChaCha and Salsa

Further, we randomly sampled 100 additional $\mathcal{OD}$-quadruples (other than our above-mention 4 particular $\mathcal{OD}$ bit positions used in our attack) and found a mean relative gap $\approx 3.4\%$ for each of 256-bit ChaCha and Salsa. We also check the independence of the forward output difference events for the toy variants (given in Section 6) with $NumIV = 2^{24}$ and $NumKey = 2^8$ and found $\Pr_{joint}$ and $\Pr_{prod}$ to be fairly close. Although the gap is occasionally larger in certain cases, such as the relative error $\approx 10\%$ in ToyChaCha, it is expected because the 32-bit toy cipher has far less randomness than the 128/256-bit full ciphers.

In principle, one may similarly check the independence of the backward bias events also. However, since the backward biases ϵ_{a_t}'s are much smaller than the forward biases ϵ_{d_t}'s, the required sample size to check the independence for the full size ciphers becomes impractical. For the toy versions, we check the independence of the backward output difference events with $NumIV = 2^{24}$ and $NumKey = 2^8$, by comparing their joint and product probabilities. The corresponding relative gaps for all the toy versions in both forward and backward directions, for unions of 2 and 3 $\mathcal{OD}$s are summarized in Table 18 and Table 19, respectively.

We have also calculated the theoretical and practical values of ϵ^*_u using Eqn. (23), which depend on both the forward biases $\epsilon^*_{d_t}$ and backward biases $\epsilon^*_{a_t}$. Columns $\epsilon^*_{u_{Th}}$ and $\epsilon^*_{u_{Pr}}$ in Table 20 and Table 5 illustrate that these theoretical and practical values match closely for $v' = 3$ and 2 union of $\mathcal{OD}$s in ToyChaCha respectively. All the biases in this table also are calculated corresponding to $NumIV = 2^{24}$ and $NumKey = 2^{10}$.

8 Conclusion

This work provides a new approach of constructing the PNB set in terms of the union of the output bit differentials ($\mathcal{OD}$) for the same input bit differential

Bias	Cipher	$\mathcal{ID}$	$\mathcal{B}'$	$\mathbf{Pr_1}$	$\mathbf{Pr_2}$	$\mathbf{Pr}_{\text{joint}}$	$\mathbf{Pr_1 \cdot Pr_2}$	Absolute Gap	Relative Gap
Forward	ToyChaCha	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{2,0}\}$	0.06090	0.8702	0.05870	0.05300	0.005706	9.72%
	ToyChaCha48	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{1,1}\}$	0.29687	0.24951	0.07409	0.07407	0.000018	0.02%
	ToyChaCha64	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,4}, \Delta^{(2)}_{2,0}\}$	0.11727	0.87006	0.10103	0.10203	0.001005	0.99%
Backward	ToyChaCha	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{2,0}\}$	0.50120	1	0.50120	0.50120	3.96×10^{-5}	0.0079%
	ToyChaCha48	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{1,1}\}$	0.50125	0.55519	0.27830	0.27829	1.47×10^{-5}	0.005%
	ToyChaCha64	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,4}, \Delta^{(2)}_{2,0}\}$	0.50001	0.61165	0.30572	0.30584	1.13×10^{-4}	0.04%

Table 18. Relative gaps of independence in forward and backward direction for the toy variants using union of 2 $\mathcal{OD}$s.

Bias	Cipher	$\mathcal{ID}$	$\mathcal{B}'$	$\mathbf{Pr_1}$	$\mathbf{Pr_2}$	$\mathbf{Pr_3}$	$\mathbf{Pr}_{\text{joint}}$	$\mathbf{Pr_1 \cdot Pr_2 \cdot Pr_3}$	Absolute Gap	Relative Gap
Forward	ToyChaCha	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{9,6}, \Delta^{(2)}_{12,2}\}$	0.08498	0.38338	0.37575	0.01368	0.01224	0.001439	10.52%
	ToyChaCha48	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{12,1}, \Delta^{(2)}_{13,9}\}$	0.29692	0.03651	0.37440	0.00398	0.00406	0.000080	2.02%
	ToyChaCha64	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{3,4}, \Delta^{(2)}_{12,0}\}$	0.06147	0.81767	0.78881	0.03679	0.03965	0.002861	7.78%
Backward	ToyChaCha	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{9,6}, \Delta^{(2)}_{12,2}\}$	0.50123	0.53891	0.50135	0.13558	0.13543	0.000156	0.12%
	ToyChaCha48	$\Delta^{(0)}_{12,5}$	$\{\Delta^{(2)}_{0,5}, \Delta^{(2)}_{12,1}, \Delta^{(2)}_{13,9}\}$	0.50061	0.50118	0.50031	0.12551	0.12552	0.000011	0.009%
	ToyChaCha64	$\Delta^{(0)}_{13,0}$	$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{3,4}, \Delta^{(2)}_{12,0}\}$	0.49998	0.50000	0.50179	0.12549	0.12544	0.000051	0.04%

Table 19. Relative gaps of independence in forward and backward direction for the toy variants using union of 3 $\mathcal{OD}$s.

($\mathcal{ID}$) at the initial round of the ciphers. This increases the cardinality of the PNB set. Further, the keyspace can be partitioned into smaller parts leading to more efficient key recovery.

One interesting related problem is: how many (v') and which $\mathcal{OD}$s should we choose to perform the attack most efficiently. This depends on partitioning the PNB sets into mutually exclusive subsets such that each subset contains approximately key size/($v' + 1$) key bits. In our work, we have taken $v' = 4$ and four arbitrary but fixed $\mathcal{OD}$s. Due to approximations under independence assumptions, it is expected that the union of more PNBs we take, the relative gap between a joint probability and the product of the corresponding marginal probabilities would also continue increasing. We have verified this fact for all toy versions. Hence, we have to choose the number PNBs to be judicially united. We have shown that if we take 4 $\mathcal{OD}$s for ToyChaCha, then the relative gap

$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d^*	ϵ_a^*	$\epsilon^*_{u_{Th}}$	$\epsilon^*_{u_{Pr}}$
$\Delta^{(0)}_{13,0}$	$\Delta^{(2)}_{1,3}$	-0.5818	0.0008023	0.0002137	0.000228
	$\Delta^{(2)}_{12,5}$	0.5872	0.002579		
	$\Delta^{(2)}_{14,6}$	0.6378	0.0009301		
	$\Delta^{(2)}_{1,4}$	-0.7619	0.0003426	0.0003143	0.0003506
	$\Delta^{(2)}_{2,5}$	0.7142	0.001488		
	$\Delta^{(2)}_{12,5}$	0.5872	0.002562		
	$\Delta^{(2)}_{1,4}$	-0.7619	0.0003746	0.0002513	0.0002473
	$\Delta^{(2)}_{12,5}$	0.5872	0.002566		
	$\Delta^{(2)}_{14,6}$	0.6378	0.0008495		
	$\Delta^{(2)}_{1,5}$	-0.8618	0.0004065	0.0003153	0.000316
	$\Delta^{(2)}_{2,5}$	0.7142	0.001530		
	$\Delta^{(2)}_{12,5}$	0.5872	0.002629		
	$\Delta^{(2)}_{1,5}$	-0.8619	0.0003510	0.0002453	0.0002351
	$\Delta^{(2)}_{12,5}$	0.5872	0.002568		
	$\Delta^{(2)}_{14,6}$	0.6378	0.0008187		
	$\Delta^{(2)}_{1,6}$	0.9159	0.001040	0.0005036	0.0005272
	$\Delta^{(2)}_{2,5}$	0.7142	0.001539		
	$\Delta^{(2)}_{12,5}$	0.5872	0.002642		
	$\Delta^{(2)}_{1,6}$	0.9159	0.001944	0.000373	0.0003968
	$\Delta^{(2)}_{3,4}$	0.6378	0.0002799		
	$\Delta^{(2)}_{12,5}$	0.5872	0.001167		

Table 20. $\epsilon^*_{u_{Th}}$ vs $\epsilon^*_{u_{Pr}}$ for union of 3 $\mathcal{OD}$s of ToyChaCha.

above becomes much higher than that for 3 $\mathcal{OD}$s. Further, if we fix the number of $\mathcal{OD}$'s, then the relative gap decreases monotonically from 32-bit to full-scale ciphers, establishing the reliability of independence models as the state size and randomness increase.

Hence, we have considered the union of 2 and 3 $\mathcal{OD}$s for the key recovery attacks of the toy versions and 4 $\mathcal{OD}$s for original ChaCha and Salsa ciphers.

Our attack achieves a time complexity of $2^{167.90}$ for 7-round ChaCha, $2^{237.41}$ for 7.5-round ChaCha and $2^{183.54}$ for 8-round Salsa, significantly improving all the existing PNB-based attacks for 7 and 8 rounds on 256-bit version of these ciphers. In the 128-bit setting, the proposed attack yields time complexities of $2^{115.24}$ against 6.5-round ChaCha and $2^{98.40}$ against 7.5-round Salsa, respectively. Further, the time complexity of 7.5-round ChaCha using our proposed method is better than the time complexity of the latest published work by Flórez-Gutiérrez and Todo at Eurocrypt 2025 and Asiacrypt 2025 both, which are $2^{250.2}$ and $2^{243.6}$ respectively. As future work, it would be interesting to combine the special techniques like $\mathcal{HW}$ in the intermediate round and other feasible conditions such as bit puncturing with our proposed method using PNBS and investigate whether it improves the time Complexity for the key recovery attack.

Exhaustively enumerating the attack complexities for the other variations of the ciphers and for different fixed parameters such as $\epsilon'_d, \gamma, \alpha$ can also be taken up as another important future work.

References

[1] Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: analysis of Salsa, ChaCha, and Rumba. In: Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers 15. pp. 470–488. Springer (2008)

[2] Beierle, C., Broll, M., Canale, F., David, N., Flórez-Gutiérrez, A., Leander, G., Naya-Plasencia, M., Todo, Y.: Improved differential-linear attacks with applications to arx ciphers. Journal of Cryptology **35**(4), 29 (2022)

[3] Bellini, E., Gérault, D., Grados, J., Makarim, R.H., Peyrin, T.: Boosting differential-linear cryptanalysis of ChaCha7 with MILP. IACR Trans. Symmetric Cryptol. **2023**(2), 189–223 (2023), https://doi.org/10.46586/tosc.v2023.i2.189-223

[4] Bernstein, D.J.: The Salsa20 family of stream ciphers. In: New stream cipher designs: the eSTREAM finalists, pp. 84–97. Springer (2008)

[5] Bernstein, D.J., et al.: ChaCha, a variant of Salsa20. In: Workshop record of SASC. vol. 8, pp. 3–5. Citeseer (2008)

[6] Choudhuri, A.R., Maitra, S.: Significantly improved multi-bit differentials for reduced round Salsa and ChaCha. IACR Transactions on Symmetric Cryptology pp. 261–287 (2016)

[7] Coutinho, M., Passos, I., Vásquez, J.C.G., Sarkar, S., de Mendonça, F.L., de Sousa Jr, R.T., Borges, F.: Latin dances reloaded: Improved cryptanalysis against Salsa and ChaCha, and the proposal of Forró. Journal of Cryptology **36**(3), 18 (2023)

[8] Coutinho, M., Souza Neto, T.C.: Improved linear approximations to arx ciphers and attacks against ChaCha. In: Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40. pp. 711–740. Springer (2021)

[9] Crowley, P.: Truncated differential cryptanalysis of five rounds of Salsa20. Cryptology ePrint Archive (2005)

[10] DeGroot, M.H., Schervish, M.J.: Probability and Statistics. Addison Wesley, 3rd edn. (2002)

[11] Dey, C., Dey, S., Girme, R., Sarkar, S.: Enhancing the key recovery attack on round reduced Salsa. IEEE Access **12**, 31736–31744 (2024), https://doi.org/10.1109/ACCESS.2024.3367797

[12] Dey, S.: Advancing the idea of probabilistic neutral bits: First key recovery attack on 7.5 round ChaCha. IEEE Trans. Inf. Theory **70**(8), 6091–6106 (2024), https://doi.org/10.1109/TIT.2024.3389874

[13] Dey, S., Garai, H.K., Maitra, S.: Cryptanalysis of reduced round ChaCha - new attack & deeper analysis. IACR Trans. Symmetric Cryptol. **2023**(1), 89–110 (2023), https://doi.org/10.46586/tosc.v2023.i1.89-110

[14] Dey, S., Garai, H.K., Sarkar, S., Sharma, N.K.: Revamped differential-linear cryptanalysis on reduced round ChaCha. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 86–114. Springer (2022)

[15] Dey, S., Garai, H.K., Sarkar, S., Sharma, N.K.: Enhanced differential-linear attacks on reduced round ChaCha. IEEE Trans. Inf. Theory **69**(8), 5318–5336 (2023), https://doi.org/10.1109/TIT.2023.3269790

[16] Dey, S., Leander, G., Sharma, N.K.: Improved key recovery attacks on reduced-round salsa20. Designs, Codes and Cryptography **93**(1), 243–262 (2025)

[17] Dey, S., Maitra, S., Sarkar, S., Sharma, N.K.: Significantly improved cryptanalysis of Salsa20 with two-round criteria. IACR Trans. Symmetric Cryptol. **2025**(1), 420–443 (2025)

[18] Dey, S., Sarkar, S.: Improved analysis for reduced round Salsa and Chacha. Discrete Applied Mathematics **227**, 58–69 (2017)

[19] Fischer, S., Meier, W., Berbain, C., Biasse, J.F., Robshaw, M.J.: Non-randomness in estream candidates Salsa20 and TSC-4. In: Progress in Cryptology-INDOCRYPT 2006: 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006. Proceedings 7. pp. 2–16. Springer (2006)

[20] Flórez-Gutiérrez, A., Todo, Y.: Divide-and-conquer trail enumeration puncturing: Application to salsa and chacha. In: ASIACRYPT (1). pp. 35–65. Lecture Notes in Computer Science, Springer (2025)

[21] Flórez-Gutiérrez, A., Todo, Y.: Improved cryptanalysis of ChaCha: Beating pnbs with bit puncturing. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 15601, pp. 427–457. Springer (2025)

[22] Garai, H.K., Dey, S.: A multi-step key recovery attack on reduced round Salsa and ChaCha. Cryptologia pp. 1–16 (2024)

[23] Ishiguro, T., Kiyomoto, S., Miyake, Y.: Latin dances revisited: new analytic results of Salsa20 and ChaCha. In: Information and Communications Security: 13th International Conference, ICICS 2011, Beijing, China, November 23-26, 2011. Proceedings 13. pp. 255–266. Springer (2011)

[24] Maitra, S.: Chosen iv cryptanalysis on reduced round ChaCha and Salsa. Discrete Applied Mathematics **208**, 88–97 (2016)

[25] Maitra, S., Paul, G., Meier, W.: Salsa20 cryptanalysis: new moves and revisiting old styles. Cryptology ePrint Archive (2015)

[26] Matsui, M.: Linear cryptanalysis method for des cipher. In: Advances in Cryptology — EUROCRYPT '93. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33

[27] Miyashita, S., Ito, R., Miyaji, A.: Pnb-focused differential cryptanalysis of ChaCha stream cipher. In: Australasian Conference on Information Security and Privacy. pp. 46–66. Springer (2022)

[28] Moore, D.S., McCabe, G.P., Craig, B.A.: Introduction to the Practice of Statistics. W. H. Freeman, New York, 10 edn. (2021)
[29] Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for arx: Application to Salsa20. Cryptology ePrint Archive (2013)
[30] Sahoo, S., Chakraborty, D., Sarkar, S.: Chasing shadows: Advancements in differential-linear cryptanalysis for chacha. IEEE Trans. Inf. Theory **71**(8), 6451–6469 (2025)
[31] Sakiyama, K., Sasaki, Y., Li, Y.: Security of Block Ciphers - From Algorithm Design to Hardware Implementation. Wiley (2015)
[32] Schervish, M.J., DeGroot, M.H.: Probability and statistics, vol. 563. Pearson Education London, UK: (2014)
[33] Sharma, N.K., Dey, S., Sarkar, S., Maitra, S.: On improved cryptanalytic results against ChaCha for reduced rounds ≥ 7. In: Mukhopadhyay, S., Stanica, P. (eds.) Progress in Cryptology - INDOCRYPT 2024 - 25th International Conference on Cryptology in India, Chennai, India, December 18-21, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 15496, pp. 29–52. Springer (2024), https://doi.org/10.1007/978-3-031-80311-6_2
[34] Shi, Z., Zhang, B., Feng, D., Wu, W.: Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In: International Conference on Information Security and Cryptology. pp. 337–351. Springer (2012)
[35] eSTREAM Portfolio of Stream Ciphers.: The ecrypt stream cipher project. https://www.ecrypt.eu.org/stream/
[36] Tsunoo, Y., Saito, T., Kubo, H., Suzaki, T., Nakashima, H.: Differential cryptanalysis of Salsa20/8. In: Workshop Record of SASC. vol. 28 (2007)
[37] Wang, S., Liu, M., Hou, S., Lin, D.: Moving a step of ChaCha in syncopated rhythm. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 273–304. Springer (2023), https://doi.org/10.1007/978-3-031-38548-3_10
[38] Witte, R., Witte, J.: Statistics. Wiley (2017), https://books.google.co.in/books?id=KcxjDwAAQBAJ
[39] Xu, Z., Xu, H., Tan, L., Qi, W.: Differential-linear cryptanalysis of reduced round ChaCha. IACR Trans. Symmetric Cryptol. **2024**(2), 166–189 (2024)

A Summary of Union of PNBs of ChaCha

$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$
384	45	186	29	416	43	184	27	448	46	193	38	480	44	185	28
385	48	217	91	417	49	217	91	449	47	216	91	481	46	217	91
386	60	221	79	418	63	225	79	450	60	226	79	482	62	223	79
387	65	216	53	419	65	217	47	451	61	211	41	483	67	219	54
388	44	182	24	420	43	182	23	452	40	173	24	484	43	181	24
389	47	207	54	421	45	205	54	453	45	204	54	485	45	204	54
390	56	222	54	422	55	222	54	454	59	221	54	486	56	220	54
391	65	220	41	423	64	220	41	455	66	221	41	487	62	219	41
392	45	184	28	424	45	186	28	456	45	185	28	488	46	186	28
393	49	222	91	425	47	219	91	457	47	217	91	489	48	221	91
394	55	223	79	426	55	223	80	458	55	223	80	490	54	224	80
395	51	204	41	427	50	203	41	459	52	204	41	491	53	214	54
396	40	178	28	428	46	180	27	460	42	178	28	492	42	186	27
397	44	214	91	**429**	**47**	**218**	**91**	461	43	211	91	493	45	212	91
398	58	227	80	430	57	226	80	462	57	226	80	494	59	226	80
399	76	234	80	431	76	234	80	463	77	234	80	495	76	234	80
400	43	180	38	432	43	180	39	464	46	188	38	496	38	167	38
401	45	218	91	433	46	222	91	465	44	217	91	497	44	218	91
402	55	232	80	434	58	233	80	466	56	233	80	498	57	234	80
403	49	209	41	435	54	218	53	467	52	215	41	499	54	215	42
404	47	184	28	436	45	179	26	468	44	178	27	500	43	178	28
405	43	197	54	437	43	197	54	469	44	201	54	501	45	203	54
406	54	225	54	438	55	225	54	470	53	223	55	502	50	219	54
407	50	203	54	439	53	211	54	471	53	210	54	503	49	203	54
408	47	193	42	440	48	195	41	472	46	193	41	504	47	194	41
409	50	215	91	441	49	212	91	473	49	215	91	505	49	212	91
410	56	225	80	442	57	224	80	474	56	225	80	506	57	225	80
411	54	212	54	443	53	211	53	475	57	217	53	507	56	218	55
412	46	190	28	444	44	180	27	476	45	191	28	508	47	193	38
413	45	221	91	445	46	221	91	477	46	221	91	509	46	221	91
414	49	225	80	446	52	225	80	478	49	225	80	510	49	225	80
415	49	211	54	447	47	209	54	479	47	209	54	511	47	209	53

Table 21. Number of union of PNB bits of P_i for $\mathcal{OD}_i$ corresponding to all possible $\mathcal{ID}$ bits for ChaCha. The forward bias and the PNB bits are calculated corresponding to the thresholds $\epsilon'_d = 0.005$ and $\gamma = 0.07$ respectively.

B Summary of Union of PNBs of Salsa

$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$	$\mathcal{ID}$	$\#\mathcal{OD}$	$\lvert\cup_{i\in\mathcal{OD}} P_i\rvert$	$\max(\lvert P_i\rvert)$
192	-	-	-	224	16	195	41	256	15	173	36	288	2	39	33
193	-	-	-	225	15	196	43	257	15	175	37	289	1	28	28
194	-	-	-	226	16	201	43	258	16	176	38	290	1	30	30
195	-	-	-	227	17	199	43	259	17	175	37	291	1	30	30
196	-	-	-	228	17	197	42	260	16	173	37	292	1	29	29
197	-	-	-	229	17	200	43	261	15	176	37	293	1	29	29
198	-	-	-	230	16	193	42	262	16	171	37	294	1	28	28
199	-	-	-	231	15	199	41	263	16	177	36	295	1	28	28
200	-	-	-	232	15	203	42	264	18	191	36	296	2	36	29
201	-	-	-	233	15	203	42	265	18	190	36	297	2	37	30
202	-	-	-	234	15	197	42	266	15	174	36	298	1	29	29
203	-	-	-	235	16	197	42	267	17	177	38	299	1	28	28
204	-	-	-	236	17	214	43	268	17	187	37	300	1	32	32
205	-	-	-	237	17	201	42	269	17	181	37	301	1	32	32
206	-	-	-	238	15	197	43	270	15	179	37	302	2	37	32
207	-	-	-	239	15	198	43	271	15	173	37	303	1	28	28
208	-	-	-	240	16	200	42	272	15	172	37	304	1	28	28
209	-	-	-	241	19	218	42	273	19	191	36	305	1	28	28
210	-	-	-	242	19	221	42	274	18	192	36	306	1	32	32
211	-	-	-	243	16	197	41	275	15	171	36	307	2	38	32
212	-	-	-	244	15	199	42	276	15	173	36	308	2	37	32
213	-	-	-	245	15	200	42	277	17	187	36	309	2	36	29
214	-	-	-	246	17	201	42	278	18	188	37	310	2	36	29
215	-	-	-	247	16	198	40	279	15	168	34	311	2	35	30
216	-	-	-	248	15	196	43	280	15	173	37	312	1	27	27
217	-	-	-	249	15	197	42	281	16	174	37	313	1	28	28
218	-	-	-	250	15	198	43	282	16	176	37	314	1	28	28
219	-	-	-	251	15	201	43	283	16	177	37	315	1	30	30
220	-	-	-	252	15	203	42	284	15	177	37	316	1	30	30
221	-	-	-	253	17	201	42	285	17	178	37	317	1	31	31
222	-	-	-	254	18	215	42	286	18	190	36	318	1	29	29
223	-	-	-	**255**	**25**	**222**	**42**	287	25	202	37	319	2	40	33

Table 22. Number of union of PNB bits of P_i for $\mathcal{OD}_i$ corresponding to all possible $\mathcal{ID}$ bits for Salsa. The forward bias and the PNB bits are calculated with respect to the thresholds $\epsilon'_d = 0.005$ and $\gamma = 0.07$ respectively.

C Union of PNB Sets for ChaCha

The finding PNB bits experiment is done with respect to the $NumIV = 2^{34}$ and the backward bias with respect to $NumIV = 2^{35}$ and $NumKey = 2^{10}$. Here the input difference is given at $\Delta^{(0)}_{13,13}$ in the initial state. The output differential set, $\mathcal{B}$ is generated with $|\mathcal{B}| = 26$. The ϵ'_d and γ parameters are taken as 0.005 and 0.07 as discussed above in the Section 5. The exclusive PNB sets for $\mathcal{B}' = \{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{3,0}, \Delta^{(3)}_{11,0}, \Delta^{(3)}_{2,0}\}$ are given in the following:

$P'_1 = \{0, 1, 2, 3, 6, 7, 15, 16, 19, 22, 23, 27, 31, 32, 35, 38, 39, 40, 41, 42, 51, 52, 55, 56, 59, 60, 61, 62, 63, 66, 78, 90, 95, 103, 115, 123, 140, 152, 155, 156, 159, 160, 161, 162, 163, 180, 181, 182, 191, 199, 219, 223, 224, 244\}$, $|P'_1| = 54$,

$P'_2 = \{128, 129, 130, 131, 8, 9, 10, 20, 148, 149, 150, 24, 28, 29, 30, 34, 167, 46, 58, 187, 192, 71, 83, 212, 91, 96, 97, 98, 99, 102, 236, 127, 111, 112, 118, 119,

248, 251, 252, 255}, $|P'_2| = 40$,

P'_3 = {4, 136, 137, 138, 11, 139, 144, 17, 18, 157, 36, 47, 176, 188, 64, 193, 67, 68, 69, 72, 73, 74, 79, 213, 216, 92, 93, 94, 220, 225, 226, 227, 100, 101, 228, 104, 105, 106, 107, 232, 250, 240, 243, 249, 122, 253, 254}, $|P'_3| = 47$,

P'_4 = {65, 70, 135, 204, 14, 80, 86, 87, 26, 116, 245, 246, 120, 124, 125, 126} , $|P'_4| = 16$.

D Union of PNB Sets for Salsa

Here also the experiments for finding the PNB sets are done with respect to the $NumIV = 2^{34}$ and the backward bias corresponding to $NumIV = 2^{35}$ and $NumKey = 2^{10}$. Here the input difference is given at $\Delta^{(0)}_{7,31}$ in the initial state. The output differential set $\mathcal{B}$, with $|\mathcal{B}| = 17$. The ϵ'_d and γ parameters are taken as 0.005 and 0.07 as discussed above in the Section 5. The exclusive PNB sets for $\mathcal{B}'$ = {$\Delta^{(4)}_{1,23}$, $\Delta^{(4)}_{1,4}$, $\Delta^{(4)}_{1,14}$, $\Delta^{(4)}_{6,15}$} are given in the following:

P'_1 = {3, 4, 5, 6, 7, 80, 81, 82, 97, 98, 99, 100, 117, 157, 158, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 219, 220, 221, 222, 223, 224, 233, 234, 235, 251, 252, 253, 254, 255}, $|P'_1| = 42$,

P'_2 = {138, 139, 16, 17, 18, 19, 20, 160, 161, 162, 163, 164, 165, 166, 167, 187, 188, 189, 190, 191, 200, 201, 202, 203, 93, 94, 95, 232, 236, 237, 110, 111, 112, 113, 246, 247}, $|P'_2| = 36$,

P'_3 = {71, 72, 73, 210, 211, 148, 212, 213, 26, 27, 28, 29, 30, 31, 225, 226, 168, 169, 170, 171, 172, 173, 108, 242, 243, 244, 245, 120, 121, 122}, $|P'_3| = 30$,

P'_4 = {128, 129, 65, 66, 8, 9, 10, 11, 12, 13, 14, 15, 142, 143, 83, 84, 85, 86, 87, 88, 104, 105, 51, 52, 53, 54} , $|P'_4| = 26$.

E Results corresponding to union of three $\mathcal{OD}$s

For 256 bit version of 7-round ChaCha. Here also the output differential set $\mathcal{B}$ remains same with the union of 4 PNB sets. For v'= 3, the reduced output differential set $\mathcal{B}'$ contains {$\Delta^{(3)}_{0,0}$, $\Delta^{(3)}_{11,0}$, $\Delta^{(3)}_{3,0}$} for $\epsilon'_d = 0.005$ and P'_t = {54, 67, 20} for $\mathcal{OD}_t \in \mathcal{B}'$ i.e. total 141 key bits remain in $\cup_t P'_t$. This gives the value of ϵ^*_u, final data, time complexities equals 2.1408×10^{-9}, $2^{61.99}$ and $2^{176.99}$ for 7-round ChaCha.

For 256 bit version of 7.5-round ChaCha. For the union of 3 $\mathcal{OD}$s, using the same procedure, we get B'=$\{\Delta^{(3)}_{11,0}, \Delta^{(3)}_{0,0}, \Delta^{(3)}_{2,0}\}$ for using the differential approach. This results the exclusive PNB sets P'_t= {38, 13, 11} corresponding to union of 3 $\mathcal{OD}$s. In this case, the final time complexities using our attack approach becomes exhaustive i.e. 2^{256}.

For the differential-linear approach, using the union of 3 $\mathcal{OD}$s also, we get the $\mathcal{OD}$ bits and the PNB sets same with the differential approach. Hence for this case also (like $v' = 4$), the data and the time complexities will remain same.

For 256 bit version of 8-round Salsa. For v'=3, we find $\mathcal{B}' = \{\Delta^{(4)}_{1,23}, \Delta^{(4)}_{1,4}, \Delta^{(4)}_{1,14}\}$. This results the final ϵ^*_u, data, time complexities equals 3.1321×10^{-9}, $2^{60.89}$ and $2^{208.89}$ for union of 3 PNB sets.

For 128 bit version of 6.5-round ChaCha. The reduced output differential set $\mathcal{B}'$ corresponding to $\epsilon'_d = 0.005$ becomes $\{\Delta^{(3)}_{0,0}, \Delta^{(3)}_{11,0}, \Delta^{(3)}_{3,0}\}$. This results $P'_t = \{25, 36, 11\}$ for $\mathcal{OD}_t \in \mathcal{B}'$. The value of ϵ^*_u, final data, time complexities becomes 2.1412×10^{-9}, $2^{61.99}$ and $2^{117.99}$ for 6.5-round ChaCha.

For 128 bit version of 7.5-round Salsa. For v'= 3, the reduced output differential set $\mathcal{B}'$ contains $\{\Delta^{(4)}_{1,14}, \Delta^{(4)}_{6,23}, \Delta^{(4)}_{6,15}\}$ for $\epsilon'_d = 0.005$ and $P'_t = \{32, 27, 17\}$ for $\mathcal{OD}_t \in \mathcal{B}'$. This gives the value of ϵ^*_u, final data, time complexities equals -2.8285×10^{-9}, $2^{61.19}$ and $2^{113.19}$ for 7.5-round Salsa.

F Comparison between time Complexities after altering P'_i in ToyChaCha

Step 1				Step 2				Final
$\mathcal{B}'$	PNB set	PNB set size	$\epsilon_{u_{Th}}$	PNB set	ϵ_d	ϵ_a	PNB set size	$T_{u_{Th}}$
$\{\Delta^{(2)}_{1,5}, \Delta^{(2)}_{2,0}\}$	$P_1 \cup P_2$	21	0.2462	P'_1	−0.8619	1	11	$2^{19.68}$
				P'_2	0.7392	1	10	
$\{\Delta^{(2)}_{1,6}, \Delta^{(2)}_{2,0}\}$	$P_1 \cup P_2$	21	0.2467	P'_1	0.9159	1	10	$2^{19.76}$
				P'_2	0.7392	1	11	
$\{\Delta^{(2)}_{1,6}, \Delta^{(2)}_{3,4}\}$	$P_1 \cup P_2$	22	0.003195	P'_1	0.9159	1	10	$2^{31.47}$
				P'_2	0.6378	1	12	
$\{\Delta^{(2)}_{2,7}, \Delta^{(2)}_{2,0}\}$	$P_1 \cup P_2$	21	0.2456	P'_1	−0.2577	1	9	$2^{22.55}$
				P'_2	0.7392	1	12	
$\{\Delta^{(2)}_{14,5}, \Delta^{(2)}_{2,0}\}$	$P_1 \cup P_2$	21	0.2462	P'_1	−0.2841	1	11	$2^{20.82}$
				P'_2	0.7392	1	10	

Table 23. Comparison of attack parameters and resulting time complexities after altering the mutually exclusive PNB sets (P'_1 vs. P'_2).

G Derivation of Time Complexity of the Proposed Attack

In the key recovery attack procedure, if the guessed key bit is correct then the program is run for:

$$N_u + 2^{|P'_v|} + \cdots + 2^{|P'_1|} = N_u + \sum_{t=1}^{v'} 2^{|P'_t|}.$$

If the guessed key bit is incorrect, then from $(2^m - 1)$ wrong keys one will pass the distinguishing event. Now as $P_{fa}(= 2^{-\alpha})$ is the probability of false alarm, so $(2^m - 1) \cdot 2^{-\alpha}$ keys will remain after checking step 1 of key recovery attack procedure. From these many keys, first we have to search through the PNB set $P'_{v'}$. So the computation time for this iteration becomes:

$$\begin{aligned} &(2^m - 1) \cdot N_u + (2^m - 1) \cdot 2^{|P'_{v'}|-\alpha} \\ \approx\ &(2^m - 1) \cdot N_u + 2^{m+|P'_{v'}|-\alpha}, \end{aligned}$$

as $2^{|P'_{v'}|-\alpha}$ is negligible compared to $2^{m+|P'_{v'}|-\alpha}$. Now the time complexity after searching through the PNB set $P'_{v'-1}$ becomes:

$$\begin{aligned} &(2^m - 1) \cdot N_u + 2^{m+|P'_{v'}|-\alpha} + (2^m - 1) \cdot 2^{-\alpha} \cdot 2^{|P'_{v'-1}|} \cdot 2^{-\alpha} \\ &\approx (2^m - 1) \cdot N_u + 2^{m+|P'_{v'}|-\alpha} + 2^{m+|P'_{v'-1}|-2\alpha} \end{aligned}$$

as $2^{|P'_{v'-1}|-2\alpha}$ is negligible compared to $2^{m+|P'_{v'-1}|-2\alpha}$.

$$\leq (2^m - 1) \cdot N_u + 2^{m+|P'_{v'}|-\alpha} + 2^{m+|P'_{v'-1}|-\alpha}$$

since $2^{m+|P'_{v'-1}|-2\alpha} \leq 2^{m+|P'_{v'-1}|-\alpha}$.

$$\leq (2^m - 1) \cdot N_u + 2^m \cdot (2^{|P'_{v'}|} + 2^{|P'_{v'-1}|}) \cdot 2^{-\alpha}$$

By continuing this procedure up to $v' = 1$ times, the complexity estimation becomes:

$$(2^m - 1) \cdot N_u + 2^m \cdot \sum_{t=1}^{v'} 2^{|P'_t|} \cdot 2^{-\alpha}.$$

Hence the total time complexity becomes:

$$\begin{aligned} &T_u = N_u + \sum_{t=1}^{v'} 2^{|P'_t|} + (2^m - 1) \cdot N_u + 2^m \cdot \sum_{t=1}^{v'} 2^{|P'_t|} \cdot 2^{-\alpha} \\ \implies &T_u = 2^m \cdot N_u + 2^m \cdot \sum_{t=1}^{v'} 2^{|P'_t|} \cdot 2^{-\alpha} + \sum_{t=1}^{v'} 2^{|P'_t|} \\ \implies &T_u = 2^m \cdot (N_u + 2^{-\alpha} \cdot \sum_{t=1}^{v'} 2^{|P'_t|}) + \sum_{t=1}^{v'} 2^{|P'_t|}. \end{aligned} \tag{33}$$