On Improved Cryptanalytic Results against ChaCha for Reduced Rounds ≥ 7

Nitin Kumar Sharma[1], Sabyasachi Dey[1], Santanu Sarkar[2], Subhamoy Maitra[3]

[1] Department of Mathematics, Birla Institute of Technology and Science Pilani, Jawahar Nagar, Hyderabad 500078, India.
sharmanitinkumar685@gmail.com,sabya.ndp@gmail.com

[2] Department of Mathematics, Indian Institute of Technology Madras, Chennai 600 036, India
sarkar.santanu.bir1@gmail.com

[3] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India,
subho@isical.ac.in,maitra.subhamoy@gmail.com

Abstract. In this paper, we analyze the subtle issues of complexity estimates related to state-of-the-art cryptanalytic efforts on ChaCha. In this regard, we demonstrate that the currently best-known cryptanalytic result on 7-round ChaCha with time $2^{189.7}$ and data $2^{102.63}$ [Xu et al., ToSC 2024] can be estimated as $2^{178.12}$ for time and $2^{101.09}$ for data complexity. We improve the best-known result for the 7.25 round by obtaining an improved set of Probabilistic Neutral Bits and considering our revised estimation. Our result with time complexity $2^{212.43}$ and data complexity $2^{100.56}$ improves the result of Xu et al., where they could achieve time and data complexity $2^{223.9}$ and $2^{100.80}$, respectively. For both the 7 and 7.25 rounds, we can show an improvement of the order of 2^{11} in the time complexity. For 7.5-round, we improve the result of Dey [IEEE-IT 2024], which reports the time and data complexity of $2^{255.24}$ and $2^{32.64}$, respectively. By applying the formula of the same paper and incorporating additional PNBs, we obtain improved time and data complexity of $2^{253.23}$ and $2^{34.47}$, respectively. Thus, this paper describes the currently best-known cryptanalytic results against reduced round ChaCha. Our results do not affect the security claims of the complete algorithm with 20 rounds. Also, we provide a rebuttal of the Work by Wang et al. [WDL+25] and analyze their claim about the error in the "Divide-and-Conquer" Approach.

Keywords: Biases, ChaCha, Conditional Probability, Differential-Linear Cryptanalysis, Probabilistic Neutral Bits.

1 Introduction

ChaCha [Ber08] was introduced by Bernstein in 2008 following a similar design strategy of Salsa, i.e., ARX, with modified round functions. ChaCha is a more diffused and secure version of Salsa. ChaCha is now widely used in many algorithms and protocols by different organizations, including Google [Goo]. ChaCha combined with MAC algorithm Poly1305 [Ber05] to improve the software performance with no hardware acceleration, and it works faster than AES-GCM [NL18]. As noted from the public domain information, ChaCha20-Poly1305 is used in protocols like IPsec, SSH, TLS 1.2, DTLS 1.2, TLS 1.3, WireGuard, S/MIME 4.0, OTRv4, etc. [Dep].

One must note that the first paper [AFK+08] on cryptanalysis of ChaCha presented the broad idea of the meet-in-the-middle technique, where the attacker moves forward to a specific round and then returns to that round from a later one. Interestingly, to date, all the cryptanalytic techniques exploit that broad idea from [AFK+08] with different sophisticated and clever tweaks. In 2024, Dey [Dey24] introduced a new technique to obtain the Probabilistic Neutral Bits (PNB set) for the linear combination of multiple bits of the output difference position and formulate the new time complexity value for the differential-linear attack. Recently, in ToSC 2024 [XXTQ24], Xu et al. improved the correlation value of the four-round differential-linear distinguisher proposed in [BGG+23]. In this work, we have incorporated these ideas, revised the estimation of complexities, and identified improved PNB sets to obtain the currently best-known parameters for cryptanalysis of reduced round ChaCha.

1.1 Related Works

Soon after the introduction of ChaCha in 2008, cryptanalysts provided a series of results on the cipher. In FSE 2008, Aumasson et al. [AFK+08] presented the first-ever attack on 6 and 7-round ChaCha with time complexity 2^{139} and 2^{248}, respectively. The authors presented a new idea for obtaining the Probabilistic Neutral Bits (PNBs) in this work. Till now, most works have used this technique to improve the attack on ChaCha. Ishiguro et al. [IKM11] then constructed double-bit differentials to improve the attack against ChaCha. In ICISC 2012, Shi et al. [SZFW13] improved the time complexities for 6-round and 7-rounds, and they were shown to be 2^{136} and $2^{246.5}$, respectively. In 2016, Maitra [Mai16] reduced the time complexity for 7-round ChaCha to $2^{238.94}$. In this work, Maitra introduced the concept of finding the key-$\mathcal{IV}$ pair to obtain a higher bias value for the differential-linear distinguisher.

Choudhuri et al. [CM17] further provided an attack on the R-round ChaCha ($4 \leq R \leq 7$) for the 256-bit key version. The authors proposed linear extension techniques by identifying certain approximation values and introducing new high-round differential-linear distinguishers. These techniques reduce the time complexity of 7-round ChaCha to $2^{237.7}$. In 2017, Dey et al. [DS17] improved the idea of finding PNBs to improve the attack on 7-round ChaCha with the time complexity of $2^{235.2}$. In Crypto 2020, Beierle et al. [BLT20] introduced multiple linear approximations and partitioning techniques to improve the time complexity for 7-round ChaCha to $2^{230.86}$. The authors obtained a new 3.5-round single-bit differential-linear distinguisher in the process.

In 2021, cryptanalysts introduced further ideas on the cryptanalysis of ChaCha. In Eurocrypt 2021, Coutinho et al. [CSN21] provided the first explicitly derived linear approximations for 3 and 4 rounds of ChaCha. However, Dey et al. [DDSM22] revisited this work and showed that the claimed distinguisher is incorrect, invalidating the improvement. Miyashita et al. [MIM22] provided the PNB-based differential attack on ChaCha with time complexity $2^{231.63}$, but that was not better than the attack by [BLT20]. In Eurocrypt 2022, Dey et al. [DGSS22] provided a 3-step technique to obtain the PNB-set. This technique reduced the time complexity for 7-round ChaCha to $2^{221.95}$. Moreover, the complexity calculation formula given by Aumasson et al. [AFK+08] was improved in this work. Dey et al. [DGSS23] then provided a novel technique for creating a collection of PNBs and assigning values to the PNBs while guessing the keys. This technique reduced the time complexity for 7-round ChaCha to $2^{218.92}$. In FSE 2023, Dey et al. [DGM23] introduced the divide and conquer approach and reduced the time complexity of 6-round ChaCha to $2^{99.48}$. In Crypto 2023, Wang et al. [WLHL23] introduced the syncopation technique for the PNB-based approximation in the backward direction to reduce the complexity for 7-round ChaCha to $2^{210.3}$. Recently, Bellini et al. [BGG+23] used the MILP technique to obtain better linear trails, reducing the time complexity to $2^{206.8}$. Recently, Sharma et al.

[SD24] explained estimating key recovery probability in ChaCha differential attacks using empirical trials and statistics.

Table 1: Comparison of Our Attack Complexities With the Existing Attacks on ChaCha for 256-Bit Key Having 7, 7.25, and 7.5 Rounds.

7-round		
Time	Data	Ref.
2^{248}	2^{27}	[AFK+08]
$2^{246.5}$	2^{27}	[SZFW13]
$2^{238.9}$	2^{96}	[Mai16]
$2^{237.7}$	2^{96}	[CM17]
$2^{235.22}$	-	[DS17]
$2^{230.86}$	$2^{48.83}$	[BLT20]
$2^{221.95}$	$2^{90.20}$	[DGSS22]
$2^{218.92}$	$2^{87.18}$	[DGSS23]
$2^{210.3}$	$2^{103.3}$	[WLHL23]
$2^{206.8}$	$2^{110.81}$	[BGG+23] ⋆
$2^{192.89}$	$2^{93.79}$	[Dey24]
$2^{189.7}$	$2^{102.63}$	[XXTQ24]
$2^{178.12}$	$2^{101.09}$	Section 4

7.25-round		
Time	Data	Ref.
$2^{244.85}$	$2^{93.24}$	[DGSS23]
$2^{238.34}$	$2^{122.34}$	[BGG+23]⋆
$2^{228.24}$	$2^{100.90}$	[Dey24]
$2^{223.9}$	$2^{100.80}$	[XXTQ24]
$2^{212.43}$	$2^{100.56}$	Subsection 5.1

7.5-round		
Time	Data	Ref.
$2^{255.24}$	$2^{32.64}$	[Dey24]
$2^{253.23}$	$2^{34.47}$	Subsection 5.2

Very recently, in 2024, two state-of-the-art works have been published that further improved the time complexity for 7-round. [Dey24] improved the attack by sharpening the idea of PNBs. The author obtained the PNBs for each bit in the output difference and the entire linear combination of the Output Difference position ($\mathcal{OD}$) bits. This technique reduces the time and data complexities to $2^{192.89}$ and $2^{93.79}$, respectively. Further, this work revisits the technique of [BGG+23] and shows that the data complexity claimed in their attack is more than the maximum available data in their approach, making the attack infeasible. All the infeasible results are marked ⋆ in Table 1. Recently, in ToSC 2024 [XXTQ24], Xu et al. improved the correlation value of the four-round differential-linear distinguisher, earlier proposed in [BGG+23], from $2^{-34.15}$ to $2^{-32.2}$, hence reducing the time and data complexities to $2^{189.7}$ and $2^{102.63}$, respectively. The works of [Dey24, XXTQ24] also presented cryptanalysis on 7.25 and 7.5 rounds. In this work, we incorporated these two independent ideas. Considering our revised complexity-related formula and using certain revised PNB sets, we can improve the best results by a good margin.

1.2 Contributions & Organization

As we have already presented, the complexities of the existing attacks and our improvements are summarized in Table 1. The paper is organized as follows.

Section 2 describes the specification of ChaCha and the preliminary techniques required to explain our attack against $\geq$ 7-rounds of ChaCha. In Table 2, we mention the basic notations used in the paper. Next, in Section 3, we discuss the previous attack ideas combined with our proposed algorithm for PNBs. In Section 4, we describe our result on 7-round ChaCha, showing improvement in time complexity by a good margin compared to [XXTQ24], with the revised formula that we explain in Equation 5. Then, in Section 5, we present the improved cryptanalytic results for 7.25 and 7.5 rounds from [XXTQ24, Dey24]. Subsection 5.1, with our revised formula and improved PNB sets, explains the improvement

for the 7.25 round. Further, Subsection 5.2 presents the improvement for the 7.5 round with certain new PNBs, using the formula presented in the work of [Dey24] itself. In Section 6, we discuss the work of Wang et al. [WDL+25] and discuss their claim of finding errors in the works of IEEE TIT [Dey24] and INDOCRYPT 2024 [SDSM25]. Finally, Section 7 concludes the paper.

Table 2: Table for Notations.

Notation	Meaning
X	The state matrix of the cipher consisting of 16 words
X_i	i-th word of the state matrix
$X^{(r)}$	State matrix after r rounds
R-round ChaCha	ChaCha reduced to R rounds
F^{-1}	Reverse round function
$\mathcal{ID}$	Input Difference position
$\mathcal{OD}$	Output Difference position
$\mathcal{ID} - \mathcal{OD}$	Input- Output Difference position
$X_i^{(r)}[j]$	Value of j-th bit of the i-th word of $X^{(r)}$
$\Delta X_i^{(r)}[j]$	Difference at the j-th bit of the i-th word of states $X^{(r)}$ and $X'^{(r)}$
ϵ_d	Bias obtained after r rounds (forward bias)
ϵ_a	Bias obtained after $(R-r)$ rounds in backward direction
$\mathcal{OD}_i$	i-th bit of the Output Difference position
ϵ_i	Bias obtained obtained for each $\mathcal{OD}_i$ position
$\Pr_{fa}$	Probability of False Alarm Error
$\Pr_{nd}$	Probability of Non-detection Error

2 Specifications, Preliminaries & Background

Here, we describe the cipher in Subsection 2.1, and Subsection 2.2 explains the differential-linear cryptanalysis technique. [AFK+08] introduced the idea of Probabilistic Neutral Bits; we explain this technique in Subsection 2.3. The idea of finding a Key-$\mathcal{IV}$ pair and probabilistically independent bits are explained in Subsection 2.4 and Subsection 2.5, respectively.

2.1 The Design of ChaCha

ChaCha is a steam cipher introduced by Daniel J. Bernstein [Ber08] in 2008. The initial state is a 4×4 matrix of 32-bit words. That is, the cipher has a 512-bit state. The state is initialized with a 128-bit constant, a 256-bit key, a 32-bit counter, and a 96-bit nonce. The constant words (c_0, c_1, c_2, c_3) are placed in the first row. Keywords $(k_0, k_1, \ldots, k_7)$ are arranged in the second and third rows of the matrix. Nonce t_0 and counter words (v_1, v_2, v_3) are positioned in the fourth row. The constants words (c_0, c_1, c_2, c_3) for the 256-bit key structure are $c_0 =$ `0x61707865`, $c_1 =$ `0x3320646e`, $c_2 =$ `0x79622d32`, $c_3 =$ `0x6b206574`. The initial matrix looks as follows:

$$X = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} & X_{11} \\ X_{12} & X_{13} & X_{14} & X_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ \mathbf{k_0} & \mathbf{k_1} & \mathbf{k_2} & \mathbf{k_3} \\ \mathbf{k_4} & \mathbf{k_5} & \mathbf{k_6} & \mathbf{k_7} \\ t_0 & v_0 & v_1 & v_2 \end{pmatrix}.$$

The round function consists of four quarterround functions. Let F denote the round

function. In this round function, the vector (a, b, c, d) transforms into a vector (a'', b'', c'', d'') as shown below:

$$\begin{aligned} a' &= a \boxplus b; & d' &= ((d \oplus a') \lll 16); \\ c' &= c \boxplus d'; & b' &= ((b \oplus c') \lll 12); \\ a'' &= a' \boxplus b'; & d'' &= ((d' \oplus a'') \lll 8); \\ c'' &= c' \boxplus d''; & b'' &= ((b' \oplus c'') \lll 7). \end{aligned} \tag{1}$$

During odd rounds, the round function is applied to each column, and these rounds are referred to as the column-round function. The four columns are (X_0, X_4, X_8, X_{12}), (X_1, X_5, X_9, X_{13}), $(X_2, X_6, X_{10}, X_{14})$ and $(X_3, X_7, X_{11}, X_{15})$. On the other hand, during even rounds, the function is applied to the diagonals, and these rounds are known as the diagonal-round function. The four diagonals are constructed as follows: $(X_0, X_5, X_{10}, X_{15})$, $(X_1, X_6, X_{11}, X_{12})$, (X_2, X_7, X_8, X_{13}) and (X_3, X_4, X_9, X_{14}). X being the initial state is added with $X^{(n)}$ (the state after n rounds of X) to obtain the keystream block Z as $Z = X \boxplus X^{(n)}$. With $n = 20$ for ChaCha and the latest state-of-the-art results.

It should be noted that the ChaCha round function is reversible. In reverse round function (F^{-1}) the vector (a'', b'', c'', d'') acts as initial vector and changes into vector (a, b, c, d) as follows:

$$\begin{aligned} b' &= ((b'' \ggg 7) \oplus c''); & c' &= c'' \boxminus d''; \\ d' &= ((d'' \ggg 8) \oplus a''); & a' &= a'' \boxminus b'; \\ b &= ((b' \ggg 12) \oplus c'); & c &= c' \boxminus d'; \\ d &= ((d' \ggg 16) \oplus a'); & a &= a' \boxminus b; \end{aligned} \tag{2}$$

2.2 Differential-Linear Cryptanalysis

Differential and Linear cryptanalysis are the primary attack methods against symmetric ciphers. Differential cryptanalysis [BS91] was introduced by Biham and Shamir in 1990, specifically targeting DES. In 1992, Matsui developed linear cryptanalysis [MY93], initially applied to FEAL, where the attacker attempts to establish linear relationships between bits of the plaintext, ciphertext, and key using the XOR operation. By 1994, Langford and Hellman combined these two methods, creating differential-linear cryptanalysis [LH94]. This hybrid approach was first applied to DES and extended to stream ciphers. The technique involves splitting the cipher E into two subciphers, E_1 and E_2, where E_1 is analyzed using differential cryptanalysis, and E_2 is subjected to linear approximation.

To illustrate the differential-linear cryptanalysis on ChaCha, let X represent the initial state matrix, and X' denote the state matrix with a single bit difference from X at input difference position ($\mathcal{ID}$). Here, $X_i[j]$ refers to the j-th bit of the i-th word in matrix X. After introducing the difference in the initial stage, the output difference after r rounds is observed, denoted as $\Delta X_p^{(r)}[q] = X_p^{(r)}[q] \oplus X_p'^{(r)}[q]$. The position where the output difference $\Delta X_p^{(r)}[q] = 0$ occurs with high probability is identified as the output difference position ($\mathcal{OD}$). The probability value is expressed as $\frac{1}{2}(1 + \epsilon_d)$, where ϵ_d represents the bias value for the difference obtained after r rounds, also called forward bias. Following the observation of the r-round output difference, a linear relation between the output differential after r rounds and R rounds is sought. The bias for this linear approximation is denoted by ϵ_l, and since the linear relationship applies to both X and X', the differential-linear bias for R rounds is given by $\epsilon_d \cdot \epsilon_l^2$. Therefore, the total bias value obtained for the differential-linear distinguisher used in the key-recovery attack is $\epsilon_d \cdot \epsilon_l^2$.

2.3 Probabilistic Neutral Bits (PNBs)

In this section, we explain the concept of Probabilistic Neutral Bits (PNBs), also known as non-significant key bits, and outline the procedure for identifying these key bits. Probabilistic Neutral Bits have a low probability of affecting the output difference obtained after the completion of r rounds. Using this technique, the attacker need not search through all 2^{256} possible key combinations (for a 256-bit key). The focus should be on the m bits, the non-PNBs, reducing the search space to 2^m possibilities. After identifying these m bits, an exhaustive search can determine the remaining Probabilistic Neutral Bits (PNBs).

Let's formally define Probabilistic Neutral Bits (PNBs). Consider an initial state matrix X. By introducing an appropriate non-zero input difference ($\mathcal{ID}$), we obtain a new state X'. After running X and X' for r rounds, we observe the output difference ($\mathcal{OD}$), denoted by $\Delta X_p^{(r)}[q]$. The bias observed is denoted by ϵ_d. Upon completing R rounds, the final states $X^{(R)}$ and $X'^{(R)}$ are obtained, which are then used along with the initial states X and X' to generate keystream blocks Z and Z', i.e., $Z = X \boxplus X^{(R)}$ and $Z' = X' \boxplus X'^{(R)}$ respectively.

To find the Probabilistically Neutral Bits (PNBs), we alter one key bit, say l, among the total key bits (256) in the initial states X and X', resulting in new altered states Y and Y'. If we apply reverse round function F^{-1} on $Z-Y$ and $Z'-Y'$ for $(R-r)$ rounds, we obtain the state matrices M and M', where $M = F^{-(R-r)}((Z-Y)$ and $M' = F^{-(R-r)}(Z'-Y')$. For a key bit to be PNB, the probability that $\Delta M_p[q] = \Delta X_p^{(r)}[q]$ is expected to be high. We denote γ_l as the bias of this event, expressed as:

$$\Pr\left[\Delta X_p^{(r)}[q] \oplus \Delta M_p[q] = 0 \mid \Delta X = 1\right] = \frac{1}{2}(1+\gamma_l).$$

To construct the set of Probabilistically Neutral Bits (PNBs), as mentioned in the work of [AFK+08], we keep a threshold value γ. Key bits for which $\gamma_i \geq \gamma$ are classified as Probabilistically Neutral Bits. The optimal threshold value γ can be chosen in order to get a set of Probabilistically Neutral Bits, which helps in improving the time complexity.

2.4 Finding Right Pair

To find the right pair for the attack against ChaCha, we use the modified differential-linear cryptanalysis idea by [BLT20]. In this procedure, instead of dividing the cipher E into two subciphers, the cipher is divided into three parts E_1, E_m and E_2, where $E_m \circ E_1$ works on the differential part. E_1 subcipher is used to find the right Key-$\mathcal{IV}$ pair using the input difference ($\mathcal{ID}$). After the completion of E_1 part, the difference between the two states $E_1(X)$ and $E_1(X')$ is observed, and the number of differences after the first round (E_1 part) is δ. The probability p of obtaining a right pair is given by $\Pr\left[E_1(X) \oplus E_1(X') = \delta\right] = p$.

The states (X, X') for which the output difference is δ are considered right pairs. We require p^{-1} pairs on average to obtain a right pair. Hence, the attack needs to multiply p^{-2} with the complexity values obtained for the key-recovery attack, as explained in Section 5.2 of [BBC+22]. The technique to improve the complexity [BBC+22] introduced the concept of finding the Probabilistically Independent $\mathcal{IV}$ Bits, as explained in the next section.

2.5 Probabilistically Independent $\mathcal{IV}$ Bits

As mentioned in Subsection 2.4, the complexity values will be multiplied by p^{-2} using the right pair technique. In order to reduce this term, a new idea was introduced [BLT20], in which a single bit x among the $\mathcal{IV}$ bits is considered and checked that change in the

value of the single bit x will not affect the encryption process. Such bits will be chosen so that the probability of no change p_1 is approximately 1. Pr $[E_1(X+x) \oplus E_1(X'+x) = E_1(X) \oplus E_1(X')] = p_1$. Such values of x are called probabilistically independent $\mathcal{IV}$ bits. These bits are then collected together to form a set. For some pairs of initial states (X, X'), this set is observed, and the probability of the initial states (X, X') satisfy the conditional $E_1(X+x) \oplus E_1(X'+x) = E_1(X) \oplus E_1(X')$ is counted, where x denotes a single bit. Let that probability value be p', which is much larger than the p. Hence, using the concept, the factor p^{-2} will become $p^{-1} \times p'$, which reduces the complexity values by a fair margin.

3 Techniques Used in this Work

This section explains how a certain combination of existing and novel techniques can be exploited to improve state-of-the-art results. Subsection 3.1, explains the improvement in the correlation value of the four-round differential-linear distinguisher, which is an existing technique. In Subsection 3.2, we introduce novel methods for identifying PNBs. Using this newly developed PNB algorithm, we successfully determine the PNB for multiple $\mathcal{OD}$ bits—a significant advancement. The procedure for obtaining such PNBs is explained thoroughly. Finally, the modified data and time complexity values are calculated using the defined methodology mentioned in Subsection 3.3. Hence, the newly introduced PNB algorithm (Subsection 3.2), combined with the recalculated time complexity values, provides a comprehensive solution that enhances the overall efficiency of the process.

3.1 Differential-Linear Hull Technique Proposed in [XXTQ24]

In ToSC 2024, Xu et al. [XXTQ24] explained that the four-round differential-linear distinguisher proposed by Bellini et al. [BGG+23] can be improved using the differential-linear hull method. Using this method, they modified the correlation value of the two-round linear approximation $(\Delta X_2^{(3)}[4,3,0] \oplus \Delta X_7^{(3)}[20,4,0] \oplus \Delta X_8^{(3)}[20,19] \oplus \Delta X_{13}^{(3)}[4]) \longrightarrow (\Delta X_2^{(5)}[0] \oplus \Delta X_6^{(5)}[7] \oplus \Delta X_6^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0])$ to $2^{-2.05}$ over the theoretically obtained value 2^{-4}. They partition the output difference obtained after three rounds into two parts $\Delta X_7^{(3)}[20,4,0]$ and $(\Delta X_2^{(3)}[4,3,0] \oplus \Delta X_8^{(3)}[20,19] \oplus \Delta X_{13}^{(3)}[4])$ and obtain the correlation value separately for both partitions and upon applying the pilling -up lemma showed that the correlation value of the 4-round differential-linear distinguisher is improved from $2^{-34.15}$ to $2^{-32.2}$. This improved correlation value of the four-round distinguisher is used in this work to provide the differential-linear attack against 7-round ChaCha.

3.2 Improved Algorithm for PNBs: Use of Conditional Probability

In [DS17], an improved algorithm to find the probabilistically neutral bits was proposed. It was based on a Greedy approach where, in each iteration, the key bit providing the best neutrality measure along with the existing set is included as the next member in the set of PNBs. The limitation of this approach was that, as the size of the PNB set increases, the bias decreases. Therefore, identifying the best candidate in each iteration becomes very difficult in the end, and therefore, the best set might not be achieved. In this work, we propose a different idea to obtain further improvement in constructing a set of probabilistically neutral bits. This approach is also iterative but based on conditional probability. The procedure of obtaining the PNB set is divided into two parts: pre-processing and post-processing. In the pre-processing stage, we obtain the PNB set using the Algorithm 1. After collecting PNBs, we added more PNBs to the set, which provided a good bias. This addition of PNBs is explained in the post-processing part.

3.2.1 Pre-Processing

The entire process is presented in Algorithm 1.

Algorithm 1: Improved Algorithm to find the set of PNBs.

Input: $LOOP$: the number of iterations to be performed, t: the number of PNBs to be selected.

for $i = 1$ *to* t **do**
  for K_0 *to* K_{255} *such that* $K_j \notin PNB$; **do**
    $counter = 0$.
    for $loop = 1$ *to* $LOOP$ **do**
      Initialize state matrix X and generate matrix X'.
      After R quarter rounds, X generates Z, and X' generates Z'.
      while *true* **do**
        Construct $\tilde{X}, \tilde{X}'$ by put random values at PNBs of X, X'.
        $\tilde{M} = F^{-(R-r)}(Z - \tilde{X})$, $\tilde{M}' = F^{-(R-r)}(Z - \tilde{X}')$.
        if $(\tilde{M}_p[q] \oplus \tilde{M}'_p[q] = X^{(r)} \oplus X^{(r)'})$ **then**
          break
        end
      end
      $\hat{X} = \tilde{X} \oplus K_j, \hat{X}' = \tilde{X}' \oplus K_j$.
      $\hat{M} = F^{-(R-r)}(Z - \hat{X})$, $\hat{M}' = F^{-(R-r)}(Z - \hat{X}')$.
      if $(\hat{M}_p[q] \oplus \hat{M}'_p[q] = X^{(r)} \oplus X^{(r)'})$ **then**
        $counter = counter + 1$.
      end
      $Prob(K_j) = \frac{counter}{LOOP}$.
    end
    Max = Choose the K_j with the highest $Prob(K_j)$.
  end
  $PNB_{i+1} = PNB_i \cup Max$.
end

To explain more, let us denote the PNB set constructed after the i-th iteration by PNB_i. It contains i elements. In the $(i+1)$-th iteration, we do the following for all the key bits K_j such that $K_j \notin PNB_i$.

- From the matrices X and X', by assigning arbitrary values in the key bits belonging to PNB_i, we prepare the matrices $\tilde{X}, \tilde{X}'$.
- Next, we apply the reverse round function F^{-1} on $Z - \tilde{X}, Z' - \tilde{X}'$ by $(R-r)$ rounds. Let us denote the obtained matrices by $\tilde{M}, \tilde{M}'$.
- Consequently, we observe the difference $\tilde{M}_p[q] \oplus \tilde{M}'_p[q] = X^{(r)} \oplus X^{(r)'}$.
- If this difference is zero, we assign a random value at the key bit K_j of both $\tilde{X}, \tilde{X}'$, and call the new matrices $\hat{X}, \hat{X}'$.
- Then, similarly as above, we apply the reverse round function F^{-1} on $Z - \hat{X}, Z' - \hat{X}'$, and obtain $\hat{M}, \hat{M}'$.
- Further, observe the difference $\hat{M}_p[q] \oplus \hat{M}'_p[q] = X^{(r)} \oplus X^{(r)'}$.
- Repeating this process for a sufficient number of X, X', we find out the conditional probability $\Pr(\Delta\hat{M}_p[q] = \Delta X^{(r)} \mid \Delta\tilde{M}_p[q] = \Delta X^{(r)})$.

- Among all the key bits $K_j \notin PNB_i$, the one for which the probability $\Pr(\Delta\hat{M}_p[q] = \Delta X^{(r)} \mid \Delta\tilde{M}_p[q] = \Delta X^{(r)})$ is maximum is considered as the $(i+1)$-th PNB.

As mentioned above, the bias decreases as the number of PNBs increases. Therefore, identifying the best candidates for the PNB set becomes computationally intensive towards the final stages, making it exceedingly challenging. However, our approach successfully mitigates this issue. Consequently, our algorithm performs optimally when obtaining large PNB sets. LOOP represents the number of iterations the algorithm performs when testing each candidate bit. The counter value tracks how often a candidate bit behaves like a PNB by counting how many times it satisfies the condition during testing. We calculate the ratio $\frac{counter}{LOOP}$ for a particular key K_j to find the probability. If the $\frac{counter}{LOOP}$ value will be high for K_j and satisfy the conditions outlined in the algorithm, then K_j is considered PNB. In our algorithm, the bit K_j will be added to the PNB set if its probability value is high, taking into account the PNBs already selected.

3.2.2 Post-Processing

In this part, we focus on expanding the PNB set by adding carefully selected candidates that enhance the backward bias value. These candidates are chosen through a trial-and-error approach, utilizing PNB elements not initially identified by our improved algorithm. When combined with other PNBs, these candidates often yield better results. This method significantly boosts the quality and size of the PNB set. We apply this process to increase the PNB count across all rounds of ChaCha, as discussed in Section 4 and Section 5.

3.3 Improved Complexity Calculation as Explained in [Dey24]

In our attack against 7-round ChaCha, we use a recent idea proposed in [Dey24]. The attack procedure is applied on a multi-bit output difference $\mathcal{OD}$ position. Suppose the output difference is observed as a linear combination of multiple bits, $\mathcal{OD}_1, \mathcal{OD}_2, \ldots, \mathcal{OD}_k$. Then, first, the PNB set is constructed by the usual approach for the linear combination of $\mathcal{OD}$ bits. Then, for the $\mathcal{OD}_i$'s, a separate PNB set is constructed by adding extra PNBs corresponding to $\mathcal{OD}_i$ only. Therefore, we have a PNB set for each $\mathcal{OD}_i$. Now, during the actual attack, for each of the $\mathcal{OD}_i$, random values are assigned to PNB bits in the initial states, and the states $\bar{X}_i$ and $\bar{X}'_i$ are constructed corresponding to each $\mathcal{OD}_i$ position. From Z, Z', we compute $Z - \bar{X}_i, Z' - \bar{X}'_i$, and on applying the reverse round function F^{-1}, we obtain two states $\bar{M}_i$ and $\bar{M}'_i$.

We perform this step for N pairs of Z and Z', storing the difference as an N-tuple. The remaining non PNBs corresponding to each $\mathcal{OD}_i$ are guessed in the next step. After guessing the non PNBs (denoted by m_i), we find the correlation between the states $\Delta\bar{M}_i$ and $\Delta\bar{X}_i$ for each $\mathcal{OD}_i$ bit position. This correlation value is denoted by ϵ_i. We find the XOR of the N-tuple for each $\mathcal{OD}_i$. We also obtain the correlation corresponding to the linear combination of multiple bits $\mathcal{OD}_i$'s. The correlation value is denoted by ϵ_a. According to Proposition 1 in [Dey24], the total correlation is ϵ and is given by $\epsilon = \epsilon_d \times \epsilon_a \times (\prod_{i=1}^{k} \epsilon_i)$. The formula for finding the data complexity value for the fixed value of $\Pr_{nd} = 1.3 \times 10^{-3}$ is the same as mentioned in the previous works (for example [AFK+08]), using the Neyman-Pearson lemma, for $\Pr_{fa} = 2^{-\alpha}$ and $\Pr_{nd} = 1.3 \times 10^{-3}$.

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

The time complexity value mentioned in [Dey24] is given as

$$\sum_{i=1}^{k} 2^{m_i} \cdot N + 2^m \cdot N \times \frac{k-1}{2^{10} \times (R-r)} + 2^{256-\alpha} + 2^{256-m}, \tag{4}$$

where m denotes the non PNBs corresponding to the multi-bit output difference $\mathcal{OD}$ position and m_i denotes the non-PNBs corresponding to each $\mathcal{OD}_i$ position. In this paper, we will revisit this formula in a disciplined manner and will present certain modifications.

In [Dey24], the author mentioned that during the complexity calculation, the same set of operations is performed for k $\mathcal{OD}_i$ positions, which can be reduced by considering these operations as one unit of complexity as mentioned in section V-B [Dey24]. As we know, 16 additions ($\boxplus$) and 16 XORs ($\oplus$) are involved in one round of ChaCha. These operations are applied to 32-bit numbers. Hence, 32 operations (16 additions and 16 XORs) were repeated for $(R-r)$ rounds. Also, the procedure is repeated for both X and X', so the number of operations becomes equivalent to $\mathbf{32} \times \mathbf{32} \times \mathbf{2} \times (\mathbf{R}-\mathbf{r}) = \mathbf{2^{11}} \times (\mathbf{R}-\mathbf{r})$. In [Dey24], the author missed counting the 16 addition operations that will be repeated during computation. Since there are 32 operations that are functional for one round, hence instead of 16, the factor should be 32 in the computation, which will increase the denominator. The modified formula based on this observation is given below:

$$C = \sum_{i=1}^{k} 2^{m_i} \cdot N + 2^m \cdot N \times \frac{k-1}{\mathbf{2^{11}} \times (\mathbf{R}-\mathbf{r})} + 2^{256-\alpha} + 2^{256-m}. \tag{5}$$

The final data and time complexity values are $p^{-1} \times N$ and $p^{-1} \times C$, where p is the differential probability for E_1 subcipher as explained in Subsection 2.4.

4 Description of our Cryptanalysis on 7-round ChaCha

In this section, we explain the attack procedure against 7-round ChaCha based on the approaches discussed in Section 3, which improves the time complexity by a reasonable margin over previous attacks.

In the timeline of the cryptanalysis against 7-round ChaCha, the introduction of differential-linear distinguisher ($\mathcal{ID} - \mathcal{OD}$ pair) played an important role in building different attack techniques. Since 2020, most recent works have used the 3.5-round $\mathcal{ID} - \mathcal{OD}$ pair obtained by Beierle et al. [BLT20]. However, in 2023, Bellini et al. [BGG+23] found a 5-round differential-linear distinguisher using the MILP tool starting with 2-bit differences instead of 1-bit difference in the differential part. In this attack against 7-round ChaCha, we have to come back by 2 rounds during the reverse direction to compare the difference at 5-round. This leads to different variations of cryptanalysis towards building various attack ideas using this 5-round differential-linear distinguisher. As explained thoroughly in Section 3, the parallel works by [Dey24] and [XXTQ24] improved the attack procedure of [BGG+23]. The 5-round differential-linear distinguisher is given below.

$$\begin{aligned} &\mathcal{ID} : X_{15}^{(0)}[9], X_{15}^{(0)}[29] \\ &\mathcal{OD} : (\Delta X_2^{(5)}[0] \oplus \Delta X_6^{(5)}[7] \oplus \Delta X_6^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0]). \end{aligned} \tag{6}$$

4.1 Results of [Dey24] and [XXTQ24]

In [Dey24], the author uses the same $\mathcal{ID} - \mathcal{OD}$ pair mentioned above in Equation 6 with bias $\epsilon_d = 2^{-34.15}$ and uses the 3-step process of finding PNBs given in [DGSS23] and obtain the set of 156 PNBs mentioned below. The PNB bits obtained are listed in the descending order of the bias value γ_l.

31, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 95, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 1, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 191, 218, 219, 220, 221, 222, 223, 255, 2, 3, 4, 103, 32, 104, 64, 105, 123, 106, 65, 5, 33, 244, 71, 77, 124, 125, 83, 245, 204, 224, 126, 127, 34, 6, 66, 107, 246, 225, 84, 72, 205, 78, 192, 198, 210, 247, 89, 108, 199, 67, 35, 7, 8, 226, 128, 206, 79, 85, 73, 193, 109, 129, 99, 90, 211, 140, 19, 20, 141, 227, 9, 200, 194, 142, 21, 110, 68, 130, 36, 74, 231, 91, 100, 86, 14, 201, 212, 207, 80.

The correlation value for these bits is $\epsilon_a = 0.0026$. Next, in this work, the author introduces a new technique of finding PNBs for each $\mathcal{OD}$ bit of the $\mathcal{OD}$ position obtained after 5 rounds. This technique improves the time complexity to $2^{192.89}$.

At the same time, Xu et al.[XXTQ24] worked on improving the bias ϵ_d value for the 5-round differential-linear distinguisher. They successfully improved the value from $2^{-34.15}$ to $2^{-32.2}$ using the differential-linear hull technique. Xu et al. also introduce a two-step technique to find the PNBs. They obtained 169 PNBs listed below. The PNBs are arranged according to the PNB algorithm mentioned in their work. The correlation value $\epsilon_a = 0.00027$ for these PNBs.

0, 1, 2, 3, 4, 5, 6, 7, 8, 19, 20, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 71, 72, 73, 74, 77, 78, 79, 80, 83, 84, 85, 86, 89, 90, 95, 99, 100, 103, 104, 105, 106, 107, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 198, 199, 200, 204, 205, 206, 207, 210, 211, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 244, 245, 246, 247, 255, 248, 9, 130, 142, 21, 91, 212, 110, 231, 22, 143, 232, 111, 228, 10, 201, 249, 115, 147, 14, 81, 26.

4.2 Our Cryptanalytic Steps

Comparing both recent works, we observe further scope for improvement in the value of time complexity. We modified the attack in the following steps.

1. Firstly, we use the improved bias value obtained using the differential-linear hull technique for the $\mathcal{ID} - \mathcal{OD}$ pair given by [XXTQ24]. Following the differential-linear hull technique by Xu et al. [XXTQ24], the improved forward bias is $\epsilon_d = 2^{-32.2}$.
2. We use the Algorithm 1 proposed in Subsection 3.2 to construct the set of Probabilistic Neutral Bits for the $\mathcal{ID} - \mathcal{OD}$ pair and obtained bias values for different sets of PNBs and selected the best set to provide the attack against 7-round ChaCha. The comparison of complexity values for different sets of PNBs is given in Table 3.
3. Next, we find PNBs for each $\mathcal{OD}$ position mentioned in [Dey24] and apply the data and time complexity formulae for the same attack procedure.
4. Further, we improve the probability value p of obtaining a right pair as explained by Bellini et al. [BGG+23]. For this input difference, the right pair produces exactly

8 differences after the first round. Hence, they obtained that the probability of achieving a right-pair by randomly choosing the $\mathcal{IV}$ is $p = 2^{-7}$. Both [Dey24] and [XXTQ24] use the same value of p.

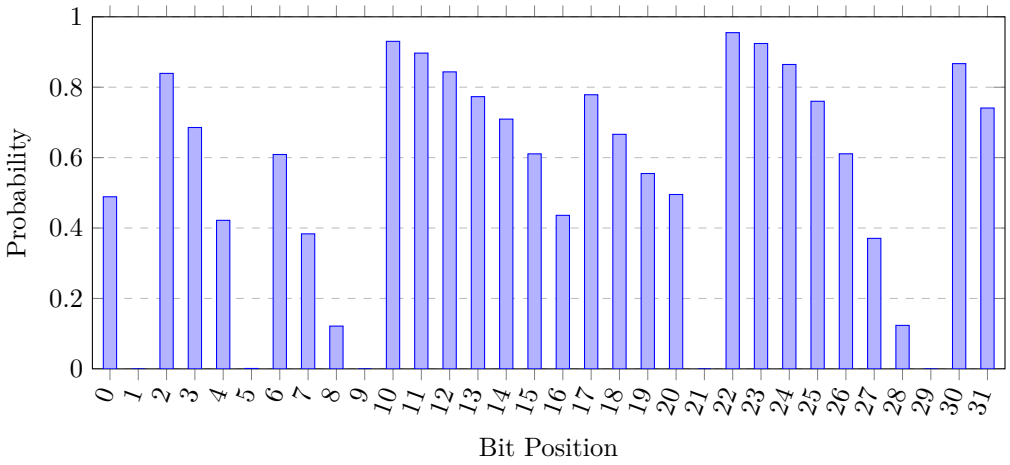

Figure 1: Distribution of Probability Value for 32 Bit Positions of word v_2.

We observed that the value of p can be reduced experimentally. According to our experiment, among all keys, approximately for 63% keys, the right pair is available, and the probability of achieving a suitable $\mathcal{IV}$ is $p = 2^{-5.2}$.

After improving the value of p over the previous attacks, we use the idea of Probabilistically Independent $\mathcal{IV}$'s mentioned in [BLT20]. We find the probability for each $\mathcal{IV}$ bit. One must note that except the $\mathcal{IV}$ word in the input difference column, for all the other $\mathcal{IV}$ bits, the probability is 1 since the criterion for the right pair is in the first round only. The distribution of the 'probability' across all 32 bits of v_2 is illustrated in Figure 1. There are four positions, namely 10, 11, 22, and 23, where the probability exceeds 0.9. We select 96 bits with probability 1 for the set of probabilistically independent $\mathcal{IV}$ bits, hence $p' = 1$. Therefore, the data and time complexity values will be multiplied by $2^{-5.2}$ instead of 2^{-7}.

4.3 Experimental Results

In this process, we used our proposed Algorithm 1 to find a better set of PNBs compared to the sets provided by [Dey24] and [XXTQ24] and used Equation 3 and Equation 5 to determine the complexity values for the key-recovery attack. The improved forward bias $\epsilon_d = 2^{-32.2}$ is used in computing the complexity values. Finding the value of $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{5} \epsilon_i$ and substituting $k = 5$ and $R - r = 7 - 5 = 2$ in Equation 5, we obtain the complexity values for different PNB sets. The outcomes are shown below, with details in Table 3.

1. Firstly, we obtain a set of 156 PNBs with bias $\epsilon_a = 0.0030$ using our PNB algorithm (Algorithm 1). This set gives better complexity in comparison to [Dey24]. After obtaining the bias values for 5 $\mathcal{OD}$ bits positions, for $\alpha = 83$, we get $N = 2^{88.87}$ and using Equation 5, $C = 2^{178.90}$ for the improved set of PNBs. After multiplying with $p^{-1} = 2^{5.2}$, the final data, and time complexity values are $2^{94.07}$ and $2^{184.10}$, respectively.

2. Running Algorithm 1 for $t = 163$, we increased the PNB set size to 163 PNBs. The bias value ϵ_a for these bits is 0.0010. Using the technique of finding PNBs for all

$\mathcal{OD}$ positions mentioned in [Dey24], for $\alpha = 87$, we get $N = 2^{92.11}$ and $C = 2^{175.13}$. The final data and time complexity values are $2^{97.31}$ and $2^{180.33}$, respectively after multiplying both values with p^{-1}.

3. Using the post-processing technique mentioned in Subsubsection 3.2.2, we added 4 Bits {26, 115, 147, 14} and achieved 167 PNBs. Similarly, for the set of 167 PNBs, we get the values N and C as $2^{94.13}$ and $2^{173.15}$, respectively, by keeping $\alpha = 89$. Upon multiplying with $p^{-1} = 2^{5.2}$, the final data, and time complexity values are $2^{99.33}$ and $2^{178.35}$, respectively.

4. We obtain a set of 169 PNBs after including {249, 81} in the set of 167 PNBs. This PNB set of 169 PNBs mentioned below is the same as the PNB set mentioned in [XXTQ24]. These PNBs are arranged in ascending order to assign the values to PNBs as explained in [DGSS23]. The consecutive PNBs are assigned $100\ldots00$, and the non-consecutive PNBs are assigned random values. The backward bias obtained is $\epsilon_a = 0.00027$.

> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 19, 20, 21, 22, 26, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 71, 72, 73, 74, 77, 78, 79, 80, 81, 83, 84, 85, 86, 89, 90, 91, 95, 99, 100, 103, 104, 105, 106, 107, 108, 109, 110, 111, 115, 123, 124, 125, 126, 127, 128, 129, 130, 140, 141, 142, 143, 147, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 198, 199, 200, 201, 204, 205, 206, 207, 210, 211, 212, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 231, 232, 244, 245, 246, 247, 248, 249, 255.

Applying the technique proposed in [Dey24], we find the PNBs and the corresponding bias ϵ_i for the 5 individual $\mathcal{OD}$ bits separately. The 5 $\mathcal{OD}_i$ positions are $X_2^{(5)}[0], X_6^{(5)}[7], X_6^{(5)}[19], X_{10}^{(5)}[12]$ and $X_{14}^{(5)}[0]$. The count of PNB sets for these five $\mathcal{OD}$ positions are $41, 58, 41, 29$, and 37, and the biases ϵ_i corresponding to these positions are $0.983, 0.99, 0.996, 0.993$, and 0.982, respectively. We provide this PNB list for five $\mathcal{OD}$ bits below.

PNB Set for 7-round

$\mathcal{OD}_1$ ($X_2^{(5)}[0]$) 11, 12, 13, 15, 16, 17, 18, 23, 24, 25, 27, 28, 29, 30, 87, 88, 92, 93, 94, 102, 112, 113, 122, 131, 144, 145, 146, 148, 149, 150, 151, 195, 208, 209, 213, 214, 215, 216, 217, 233, 243. **Count = 41**, **Bias = 0.983**.

$\mathcal{OD}_2$ ($X_6^5[7]$) 15, 16, 17, 18, 23, 24, 25, 27, 28, 29, 30, 37, 38, 82, 87, 88, 92, 93, 94, 96, 97, 98, 101, 102, 112, 113, 114, 116, 117, 118, 119, 120, 121, 122, 135, 136, 137, 138, 139, 144, 145, 146, 148, 149, 150, 151, 202, 203, 208, 209, 233, 234, 243, 250, 251, 252, 253, 254. **Count = 58**, **Bias = 0.99**.

$\mathcal{OD}_3$ ($X_6^{(5)}[19]$) 27, 28, 29, 30, 37, 38, 69, 70, 75, 76, 92, 93, 94, 96, 97, 98, 101, 102, 112, 113, 114, 116, 117, 118, 119, 120, 121, 122, 131, 148, 149, 150, 151, 213, 214, 215, 216, 217, 229, 230, 243. **Count = 41**, **Bias = 0.996**.

$\mathcal{OD}_4$ ($X_{10}^{(5)}[12]$) 37, 38, 69, 70, 75, 76, 82, 87, 88, 92, 93, 94, 96, 97, 98, 101, 195, 208, 209, 213, 214, 215, 216, 217, 250, 251, 252, 253, 254. **Count = 29**, **Bias = 0.993**.

$\mathcal{OD}_5$ $(X_{14}^{(5)}[0])$ 11, 12, 13, 15, 16, 17, 27, 28, 29, 30, 37, 38, 69, 70, 75, 76, 82, 119, 120, 135, 136, 137, 138, 148, 149, 150, 151, 195, 197, 202, 203, 216, 217, 240, 252, 253, 254. **Count = 37**, **Bias = 0.982**.

For $\alpha = 89$, and $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{5} \epsilon_i = 2^{-32.2} \times 0.00027 \times 0.983 \times 0.99 \times 0.996 \times 0.993 \times 0.982 = 2^{-44.13}$, we obtain $N = 2^{95.89}$ for $\alpha = 89$ using Equation 3. The m_i values $(1 \leq i \leq 5)$ are $46, 29, 46, 58$ and 50 respectively for the 5 $\mathcal{OD}$ bits. By fixing $k = 5$, $R = 7$, $r = 5$ and $m = 87$ in Equation 5, we get $C = 2^{172.92}$. The final data and time complexity values are $p^{-1} \times 2^{95.89} = 2^{101.09}$ and $p^{-1} \times 2^{172.92} = 2^{178.12}$ respectively, with $p^{-1} = 2^{5.2}$.

5. Using the post-processing method, we add a few potential bits to the PNB set of 169 bits mentioned in [XXTQ24]. However, the bias value ϵ_a we obtain for 170 bits increased the value of N beyond 2^{96}, which gives an infeasible attack. Therefore, we use the PNB sets of 169 bits and, through the computational techniques thoroughly explained in Section 3, we achieve the optimal complexity value.

In Table 3, we compare the complexity values for different sizes of PNB sets, specifying the corresponding bias value ϵ_a for each observation.

Table 3: Complexity for Different Sizes of PNB set for 7-round ChaCha

# PNBs	Bias ϵ_a	Bias ϵ	Time Complexity	N
156	0.0030	$2^{-40.66}$	$2^{184.10}$	$2^{88.87}$
163	0.0010	$2^{-42.25}$	$2^{180.33}$	$2^{92.11}$
167	0.0005	$2^{-43.25}$	$2^{178.35}$	$2^{94.13}$
169	0.00027	$2^{-44.13}$	$2^{178.12}$	$2^{95.89}$
170	0.00018	$2^{-44.72}$	-	$> 2^{96}$

5 Cryptanalysis of ChaCha beyond 7 rounds

In this section, we will explain the cryptanalysis of ChaCha for 7.25 and 7.5 rounds. Recently, there have been two important works [XXTQ24] and [Dey24] that mentioned the cryptanalysis of ChaCha beyond 7 rounds. A detailed discussion about the techniques and the complexity values in both works are first presented. In the next two subsections, we provide our results for 7.25 and 7.5 rounds, claiming the best result over the previous attacks.

In [XXTQ24], Xu et al. provided the attack on two round versions of ChaCha 7.25 and $7.5^{\oplus}$. For the 7.25-round, they obtained 133 PNBs and data complexity as $2^{100.8}$ and time complexity as $2^{223.9}$. They mentioned the same data and time complexity for $7.5^{\oplus}$-round. This observation is because there is no difference between the 7.25 and $7.5^{\oplus}$ round versions of ChaCha. The $7.5^{\oplus}$ version is obtained by adding 4 addition operations ($\boxplus$) to the 7.25 round version, which does not affect the PNB set. Since both versions of ChaCha have the same security against the PNB-based differential-linear attack, Xu et al. claimed that the complexity values for both versions are the same. Hence, in an actual sense, Xu et al. provided the results only up to 7.25 rounds. They have not extended their technique for 7.5 rounds. For the PNB-based differential-linear attack, the authors used the same $\mathcal{ID} - \mathcal{OD}$ as they used for the 7-round ChaCha.

The first-ever key recovery attack on 7.5-round ChaCha is presented in [Dey24]. In this work, the cryptanalysis of both 7.25 and 7.5 rounds has been explained thoroughly. However, the major difference in the attacks is the differential-linear distinguisher ($\mathcal{ID} - \mathcal{OD}$). For the attack against 7.25-round, the author used the 5-round $\mathcal{ID} - \mathcal{OD}$ given by Bellini et al. [BGG+23], whereas for the attack against 7.5-round, the 4-round $\mathcal{ID} - \mathcal{OD}$ given by [BLT20] is used. The 4-round differential has the bias value $\epsilon_d = 0.0032$, which is much higher than the bias value for the 5-round differential. Moreover, the author has also modified the error probability value $\Pr_{nd}$ so that the time complexity value is below 2^{256}. The attack is only applicable for 79% of all keys, as the author did not use the concept of right pair in the attack against 7.5-round ChaCha. Hence, this result can be improved by making it applicable to all the keys.

5.1 Analysis of Attacks on 7.25-round ChaCha

This subsection presents the attacks on the 7.25-round ChaCha and highlights the key findings. Firstly, we discuss our improvements over the works of [Dey24] and [XXTQ24] and in subsection 5.1.3 we provided our results in detail.

5.1.1 Improvement over work [Dey24]

In this work, the attack on 7.25-round is observed by finding 121 PNBs with bias value $\epsilon_a = 0.0020$. This reduces the data complexity to $2^{100.90}$ and time complexity to $2^{228.24}$. We use our PNB algorithm (Algorithm 1) to obtain a set of 121 PNBs and the bias value $\epsilon_a = 0.0039$. Using this result, we can improve the work of [Dey24], but the count of PNBs should be increased in order to provide a major improvement over the attack over the recent work [XXTQ24]. We also obtained the improved PNB sets for 5 $\mathcal{OD}$ positions $X_2^{(5)}[0], X_6^{(5)}[7], X_6^{(5)}[19], X_{10}^{(5)}[12]$ and $X_{14}^{(5)}[0]$. Combining all bias values, $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{5} \epsilon_i = 2^{-32.2} \times 0.0039 \times 0.968 \times 0.978 \times 0.972 \times 0.983 \times 0.96 = 2^{-40.4}$, we obtain $N = 2^{87.78}$ for $\alpha = 49$ using Equation 3 and $C = 2^{212.64}$ by fixing $k = 5$, $R = 7.25$, $r = 5$ and $m = 135$ in Equation 5. The final data and time complexity values for $p^{-1} \times 2^{87.78} = 2^{92.98}$ and $p^{-1} \times 2^{212.64} = 2^{217.84}$ respectively for this computation.

5.1.2 Improvement over work [XXTQ24]

The work of [XXTQ24] provided an improvement of around 2^4 over [Dey24]. The set of 133 PNBs was obtained using their PNB algorithm, which reduces the complexity value. The data complexity is $2^{100.8}$ and time complexity is $2^{223.9}$. This complexity value can be improved using the mathematical formulation discussed for the time complexity value mentioned in [Dey24]. In the next section, we have discussed our cryptanalysis against a 7.25-round ChaCha in detail.

5.1.3 Our Result

We get an improved PNB set for 7.25-round using Algorithm 1, but to increase the PNB count, we have to add some more bits to the PNB set, as mentioned in the post-processing step. To improve the recent attacks against 7.25-round ChaCha, we use the PNB set of 133 bits mentioned in [XXTQ24]. The bias $\epsilon_a = 2^{-11.25}$ for these 133 bits. Adding bit $\{195\}$ in the PNB, using the post-processing step mentioned in Subsubsection 3.2.2, we add 1 more PNB in the set. The bias value $\epsilon_a = 0.00025$ for the set of 134 PNBs. Using the technique mentioned in [Dey24], we obtain the PNB set for the 5 $\mathcal{OD}$ positions. Also, to compute the complexity values, we use the bias value $\epsilon_d = 2^{-32.2}$ improved by using

the differential-linear hull technique. The 134 bits are arranged in the ascending order as given below:

0, 7, 8, 20, 21, 22, 31, 35, 44, 45, 46, 47, 48, 51, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 71, 72, 73, 74, 77, 80, 81, 83, 84, 85, 86, 89, 90, 91, 95, 99, 100, 108, 109, 110, 111, 123, 124, 125, 126, 127, 128, 129, 130, 140, 141, 142, 143, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, **195**, 198, 199, 200, 204, 205, 206, 207, 210, 211, 212, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 231, 232, 244, 245, 246, 247, 248, 249, 255.

The 5 $\mathcal{OD}_i$ positions are $X_2^{(5)}[0], X_6^{(5)}[7], X_6^{(5)}[19], X_{10}^{(5)}[12]$ and $X_{14}^{(5)}[0]$. The count of PNB sets for these five $\mathcal{OD}$ positions are $66, 71, 67, 34$, and 65, and the bias value ϵ_i corresponding to these positions are $0.986, 0.978, 0.972, 0.983$, and 0.960, respectively. We provide this PNB list for five $\mathcal{OD}$ bits below.

PNB Set for 7.25-round

$\mathcal{OD}_1$ $(X_2^{(5)}[0])$ 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26, 27, 28, 29, 30, 49, 50, 52, 53, 54, 87, 88, 92, 93, 94, 102, 103, 104, 105, 106, 107, 112, 113, 114, 122, 131, 132, 144, 145, 146, 147, 148, 149, 150, 151, 196, 208, 209, 213, 214, 215, 216, 217, 233, 234, 243.
Count = 66, **Bias = 0.968**.

$\mathcal{OD}_2$ $(X_6^5[7])$ 19, 23, 24, 25, 26, 27, 28, 29, 30, 39, 40, 41, 42, 43, 69, 78, 79, 82, 87, 88, 92, 93, 94, 96, 97, 98, 101, 102, 103, 104, 105, 106, 107, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 135, 136, 137, 138, 139, 144, 145, 146, 147, 148, 149, 150, 151, 201, 202, 203, 208, 209, 233, 234, 235, 243, 250, 251, 252, 253, 254.
Count = 71, **Bias = 0.978**.

$\mathcal{OD}_3$ $(X_6^{(5)}[19])$ 1, 2, 3, 4, 5, 6, 36, 37, 38, 39, 40, 52, 53, 54, 69, 70, 75, 76, 78, 79, 92, 93, 94, 96, 97, 98, 101, 102, 103, 104, 105, 106, 107, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 131, 132, 133, 134, 147, 148, 149, 150, 151, 201, 202, 213, 214, 215, 216, 217, 228, 229, 230, 233, 234, 235, 243. **Count = 67**, **Bias = 0.972**.

$\mathcal{OD}_4$ $(X_{10}^{(5)}[12])$ 32, 33, 49, 50, 52, 53, 54, 69, 70, 75, 76, 78, 79, 82, 87, 88, 92, 93, 94, 96, 97, 196, 208, 209, 213, 214, 215, 216, 217, 250, 251, 252, 253, 254. **Count = 34**, **Bias = 0.983**.

$\mathcal{OD}_5$ $(X_{14}^{(5)}[0])$ 1, 2, 3, 4, 5, 6, 16, 17, 18, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 49, 50, 52, 53, 54, 69, 70, 75, 76, 78, 79, 82, 104, 105, 106, 107, 120, 121, 135, 136, 137, 138, 139, 147, 148, 149, 150, 151, 196, 197, 201, 202, 203, 216, 217, 240, 241, 252, 253, 254.
Count = 65, **Bias = 0.960**.

Therefore, for $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{5} \epsilon_i = 2^{-32.2} \times 0.00025 \times 0.968 \times 0.978 \times 0.972 \times 0.983 \times 0.96 = 2^{-43.95}$, we obtain $N = 2^{95.36}$, keeping $\alpha = 54$ using Equation 3 and $C = 2^{207.23}$ by fixing $k = 5$, $R = 7.25$, $r = 5$ and $m = 122$ in Equation 5. The final data and time complexity values for $p^{-1} \times 2^{95.36} = 2^{100.56}$ and $p^{-1} \times 2^{207.23} = 2^{212.43}$ respectively for this computation.

5.2 Critical Analysis of Attack on 7.5-round

In this section, we describe the cryptanalysis of 7.5-round ChaCha. First, we revisit the attack idea mentioned in [Dey24] and later discuss the improvement we can achieve using our techniques.

The $\mathcal{ID} - \mathcal{OD}$ pair used in attack against 7.5-round ChaCha is $X_{13}^{(0)}[6] - (\Delta X_2^{(4)}[0] \oplus \Delta X_8^{(4)}[0] \oplus \Delta X_7^{(4)}[7])$ which is different from the $\mathcal{ID} - \mathcal{OD}$ pair (Equation 6) we use in the attack against 7 and 7.25 ChaCha. The bias value for this $\mathcal{ID} - \mathcal{OD}$ pair is $\epsilon_d = 0.0032$. The PNB set size used by [Dey24] is 23, with a bias value $\epsilon_a = 0.012$ after using the memory approach introduced in [DGSS23]. In the memory approach, the PNBs from the IV column (X_1, X_5, X_9, X_{13}) are not included in the PNB set. This is done to avoid the multiplication of N and C with p^{-2} to compute the final data and time complexity as explained in Subsection 2.4.

Dey [Dey24] introduced the concept of finding the PNB for $\mathcal{OD}$ bits. The count of PNBs for the 3 $\mathcal{OD}_i$ positions $X_2^{(4)}[0]$, $X_8^{(4)}[0]$ and $X_7^{(4)}[7]$ is 16, 20 and 16 with ϵ_i's 0.66, 0.97 and 0.84 as mentioned in Table VII of [Dey24]. Therefore, $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{3} \epsilon_i = 0.0032 \times 0.012 \times 0.66 \times 0.97 \times 0.84 = 2^{-15.56}$. For the computation of N, the probability of non-detection error is considered $\Pr_{nd} = 0.147$, which is different from the $\Pr_{nd}$ used in most previous works starting from [AFK+08]. Using $\Pr_{nd} = 0.147$, term $3\sqrt{1-\epsilon^2}$ in Equation 3 will be replaced by $-0.8\sqrt{1-\epsilon^2}$ as mentioned in Section VI-C of [Dey24]. The data and time complexity values using the memory approach are $2^{32.64}$ and $2^{255.24}$.

Our Result

To provide the attack against 7.5-round ChaCha, we added 3 PNBs {89, 116, 226} using our post-processing step in the existing PNB set of 23 bits to improve this attack. For the PNB list of 26 PNBs mentioned below, we obtain bias $\epsilon_a = 0.0048$.

> 70, 71, 72, 75, 78, 86, 87, **89**, 95, 103, 104, 105, 106, **116**, 120, 121, 122, 123, 127, 155, 156, 157, 158, 159, **226**, 255.

Also, we improve the PNB count for 3 $\mathcal{OD}_i$ positions $X_2^{(4)}[0]$, $X_8^{(4)}[0]$ and $X_7^{(4)}[7]$ with bias values ϵ_i's 0.848, 0.97 and 0.856 respectively over the work of [Dey24].

PNB Set for 7.5-round

$\mathcal{OD}_1$ ($X_2^{(4)}[0]$) 35, 36, 37, 38, 75, 76, 77, 83, 95, 155, 156, 218, 255.
Count = 13, **Bias = 0.848**.

$\mathcal{OD}_2$ ($X_8^4[0]$) 0, 1, 2, 31, 97, 98, 99, 100, 101, 108, 125, 126, 140, 141, 142, 143, 152, 153, 154. **Count = 19**, **Bias = 0.97**.

$\mathcal{OD}_3$ ($X_7^{(4)}[7]$) 73, 74, 97, 98, 99, 100, 101, 102, 115, 117, 118, 119, 125, 126, 135.
Count = 15, **Bias = 0.856**.

Hence, the bias value $\epsilon = \epsilon_d \times \epsilon_a \times \prod_{i=1}^{3} \epsilon_i = 0.0032 \times 0.0048 \times 0.848 \times 0.97 \times 0.856 = 2^{-16.49}$, keeping $\alpha = 4.5$, we obtain $N = 2^{34.47}$ using the same formulation mentioned in [Dey24] ($\Pr_{nd} = 0.147$ i.e., replace term $3\sqrt{1-\epsilon^2}$ by $-0.8\sqrt{1-\epsilon^2}$ in Equation 3). We improved attack complexity with an increase in the PNB count. Using Equation 5, we obtain

the value of $C = 2^{253.23}$. The data and time complexity values are $2^{34.47}$ and $2^{253.23}$, respectively.

An important observation from using Equation 3 without modification is that the value of C exceeds 2^{256}, making the attack infeasible. However, by applying the memory-based approach, the data and time complexities are significantly reduced to $2^{34.47}$ and $2^{253.23}$, respectively. Notably, this approach successfully identifies the correct pair for only 79% of the keys, limiting the attack's overall effectiveness.

Our analysis of the 7.5-round ChaCha shows that introducing new techniques can further enhance the attack. Using our methodology, we reduced the time complexity to $2^{253.23}$, with the potential for further improvements in future research. However, this requires additional analysis, particularly by experimenting with a toy version of ChaCha [DGM23]. To determine the appropriate value of $\Pr_{nd}$ that would enable the development of more effective and refined techniques, further reducing time complexity and enhancing the overall feasibility of attacking ChaCha.

6 Rebuttal of the Work by Wang et al. [WDL+25]

Recently, in 2025 Wang et al. [WDL+25] presented work on the analysis of Differential-Linear attacks on ChaCha from recent works in IEEE TIT [Dey24] and INDOCRYPT 2024 [SDSM25]. Wang et al. claim to find some errors in the work from IEEE TIT [Dey24] and INDOCRYPT 2024 [SDSM25] and give the cryptanalytic attacks based on their new analysis. Wang et al. assert to modify the attack technique "Divide-and-Conquer" introduced in [Dey24]. However, the attacks based on this modified technique have extremely large time and data complexities compared with the claimed attacks in [Dey24, SDSM25]. They propose that the technique introduced in [Dey24] may not be able to obtain improved differential-linear attacks on ChaCha,

In [WDL+25, Section 3, Page 3], Wang et al. critically examine the prior works on ChaCha from IEEE TIT [Dey24] and INDOCRYPT 2024 [SDSM25] and state

> "The attacker utilizes different thresholds for the linear combination $\mathcal{OD}$ and the output difference bit $\mathcal{OD}_i$."

This above statement comes from the assumption that Theorem 1 is used to obtain the PNB set for both $\mathcal{OD}$ and each output difference bit $\mathcal{OD}_i$, which is not the case. Notably, Wang et al. only referenced a segment of Theorem 1 from [Dey24], despite the fact that neither of the works [Dey24, SDSM25] claims to employ Theorem 1 for identifying the PNBs. Let us compare the cryptanalytic attack on 7-Round ChaCha mentioned in [WDL+25] and the works [Dey24, SDSM25]. This will help us to thoroughly understand the technique introduced by Dey [Dey24], which is further modified by Sharma et al. [SDSM25], and draw a conclusion about the claim of Wang et al. [WDL+25].

Discussion on the Divide-and-Conquer Approach for 7-Round ChaCha

In our research, we determine the PNB set for the $\mathcal{OD}$ position $(\Delta X_2^{(5)}[0] \oplus \Delta X_6^{(5)}[7] \oplus \Delta X_6^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0])$ using the Improved Algorithm for PNBs (Algorithm 1). The initial PNB set consists of 169 elements. We set a threshold of $\theta = 0.30$ to select 163 PNBs, while the remaining six PNBs are included through a post-processing technique, as detailed in Subsection 4.3. As outlined by Dey [Dey24], after identifying the PNBs for the linear combination, the bias value is determined by guessing significant key bits. S denotes the set of significant key bits for the $\mathcal{OD}$ position. The computed bias for guessing

set S is $\epsilon_a = 0.00027$. Let us thoroughly discuss the work of Sharma et al.[SDSM25] based on the analysis by Wang et al. [WDL+25].

- After obtaining the set of PNBs and significant key bits for the $\mathcal{OD}$ position (the linear combination of $\mathcal{OD}_i$ positions), we find the set of PNBs and the remaining significant key bits for each $\mathcal{OD}_i$ position and S_i denotes the set of significant key bits for the $\mathcal{OD}_i$ position. The correlation value ϵ_a is obtained for the correct guess of set S, and by guessing the set S_i correctly, the bias value we get is ϵ_i. Divide-and-Conquer approach [Dey24] explicitly correlates these value values and provides a new attack procedure against ChaCha.
- Importantly, neither of the works [Dey24, SDSM25] suggests that the threshold value for identifying PNBs in the linear combination must be identical to the threshold for individual $\mathcal{OD}_i$ bit. Both studies employ a modified PNB algorithm as an extension of the basic PNB algorithm. In [SDSM25], an initial PNB set is established using a specific threshold, but certain PNBs are subsequently added or removed based on the improved PNB algorithm. Consequently, a single fixed threshold cannot be universally applied to all cases, which is why neither work [Dey24, SDSM25] specifies a threshold value.
- In contrast, Wang et al. adopt a threshold of 0.435 and identify 156 PNBs as mentioned in their work in [WDL+25, Section 4, Page 5]. However, their methodology diverges from the PNB algorithm described in previous works [Dey24, SDSM25]. As a result, their approach is not optimal for identifying PNBs. Additionally, they apply the same algorithm to determine the PNB set for each $\mathcal{OD}_i$, which further raises concerns about the validity of their findings.
- Moreover, the procedure for identifying PNBs for individual $\mathcal{OD}_i$ bits, as outlined in [Dey24], has not been accurately implemented by Wang et al. [WDL+25]. The PNB set for each $\mathcal{OD}_i$ should be derived using the same PNB algorithm as the one applied for the linear combination of $\mathcal{OD}_i$ bits.
- For the attack on 7-round ChaCha, as detailed in [SDSM25], the PNB set for $\mathcal{ID} - \mathcal{OD}$ pair is determined using the improved PNB algorithm. Setting a threshold of $\theta = 0.90$ and applying the post-processing technique, we obtain 210 PNBs. After excluding the 169 PNBs already given in the PNB set mentioned in Subsection 4.3, we derive a final subset of 41 PNBs. These 41 PNBs pertain specifically to $\mathcal{OD}_1$ and not to the linear combination of $\mathcal{OD}_i$ bits. The bias value for these PNBs is 0.983.

  Thus, after identifying the PNBs, we find the set of extra PNBs for each $\mathcal{OD}_i$ by eliminating the PNBs obtained for the $\mathcal{OD}$ position. The corresponding biases ϵ_i are also obtained for each $\mathcal{OD}_i$ position. The final bias value ϵ is computed as explained in Section V [Dey24].

In conclusion, the PNB set for $\mathcal{OD}$ position and k $\mathcal{OD}_i$ positions are obtained separately, and the extra PNBs are extracted for each $\mathcal{OD}_i$ position from their respective PNB set by removing the PNBs obtained for the $\mathcal{OD}$ position. Wang et al.'s [WDL+25] analysis of the Divide-and-Conquer approach lacks a thorough understanding of this PNB selection process. Their assertion that different thresholds are used for the linear combination $\mathcal{OD}$ and individual $\mathcal{OD}_i$ bits is misleading, as both previous works [Dey24, SDSM25] employ a refined PNB algorithm that does not rely on a single fixed threshold. The approach we have outlined, based on the Improved PNB Algorithm, more accurately captures the dependencies within the cipher structure and ensures a more precise bias estimation. Therefore, our findings highlight the importance of correctly applying the PNB algorithm to maintain the integrity of the Divide-and-Conquer framework in the cryptanalysis of ChaCha.

7 Conclusion

In this paper, we propose a novel heuristic to construct improved sets of Probabilistically Neutral Bits (Subsection 3.2) and also revisit the formula (Equation 5) for calculating the complexities of the attacks on ChaCha. Our ideas help in reducing the time complexity value for the cryptanalysis of $7, 7.25$, and 7.5-round ChaCha over the existing attacks (Sections 4, 5). Connecting relevant ideas mentioned in [Dey24, XXTQ24] with our refinements, we improved the time complexities for 7 and 7.25 rounds by order of around 2^{11}. We also provided an improved time complexity value for 7.5-round ChaCha. It is important to see how refinements of previous techniques and possibly new ideas can improve the cryptanalysis against ChaCha. We also examine the study by Wang et al. [WDL+25] and address their assertion of identifying errors in the cryptanalysis presented in IEEE TIT [Dey24] and INDOCRYPT 2024 [SDSM25].

Acknowledgments: The authors would like to thank the anonymous reviewers for their detailed comments that improved the editorial as well as technical presentation of the paper. The last author (Subhamoy Maitra) acknowledges the support of MeitY, Government of India, related to the initiative “Cluster - Cryptography, Information Security Education and Awareness (ISEA) Project Phase - III".

References

[AFK+08] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086, pages 470–488, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-71039-4_30.

[BBC+22] Christof Beierle, Marek Broll, Federico Canale, Nicolas David, Antonio Flórez-Gutiérrez, Gregor Leander, María Naya-Plasencia, and Yosuke Todo. Improved Differential-Linear Attacks with Applications to ARX Ciphers. *J. Cryptol.*, 35(4), oct 2022. URL: https://doi.org/10.1007/s00145-022-09437-z.

[Ber05] Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, pages 32–49, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/11502760_3.

[Ber08] Daniel Bernstein. ChaCha, a variant of Salsa20. *In Workshop Record of SASC*, pages vol. 8, pp. 3–5, 2008. URL: https://cr.yp.to/chacha/chacha-20080120.pdf.

[BGG+23] Emanuele Bellini, David Gerault, Juan Grados, Rusydi H. Makarim, and Thomas Peyrin. Boosting Differential-Linear Cryptanalysis of ChaCha7 with MILP. *IACR Transactions on Symmetric Cryptology*, 2023(2):189–223, 2023. URL: https://doi.org/10.46586/tosc.v2023.i2.189-223.

[BLT20] Christof Beierle, Gregor Leander, and Yosuke Todo. Improved Differential-Linear Attacks with Applications to ARX Ciphers. In *CRYPTO(3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020. URL: https://doi.org/10.1007/978-3-030-56877-1_12.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, volume 537, pages 2–21, Berlin, Heidelberg, 1991.

Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-38424-3_1.

[CM17] Arka Rai Choudhuri and Subhamoy Maitra. Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. *IACR Transactions on Symmetric Cryptology*, 2016(2):261–287, 2017. URL: https://doi.org/10.13154/tosc.v2016.i2.261-287.

[CSN21] Murilo Coutinho and Tertuliano C. Souza Neto. Improved Linear Approximations to ARX Ciphers and Attacks Against ChaCha. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, volume 12696, pages 711–740, Cham, 2021. Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-77870-5_25.

[DDSM22] Sabyasachi Dey, Chandan Dey, Santanu Sarkar, and Willi Meier. Revisiting Cryptanalysis on ChaCha From Crypto 2020 and Eurocrypt 2021. *IEEE Transactions on Information Theory*, 68(9):6114–6133, 2022. URL: https://doi.org/10.1109/TIT.2022.3171865.

[Dep] ChaCha Usage & Deployment. URL: https://ianix.com/pub/chacha-deployment.html.

[Dey24] Sabyasachi Dey. Advancing the idea of probabilistic neutral bits: first key recovery attack on 7.5 round ChaCha. *IEEE Transactions on Information Theory*, 2024. URL: https://doi.org/10.1109/TIT.2024.3389874.

[DGM23] Sabyasachi Dey, Hirendra Kumar Garai, and Subhamoy Maitra. Cryptanalysis of Reduced Round ChaCha- New Attack and Deeper Analysis. *IACR Transactions on Symmetric Cryptology*, page 89–110, Mar. 2023. URL: https://doi.org/10.46586/tosc.v2023.i1.89-110.

[DGSS22] Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Revamped Differential-Linear Cryptanalysis on Reduced Round ChaCha. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 86–114, Cham, 2022. Springer International Publishing. URL: https://doi.org/10.1007/978-3-031-07082-2_4.

[DGSS23] Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Enhanced Differential-Linear Attacks on Reduced Round ChaCha. *IEEE Transactions on Information Theory*, 69(8):5318–5336, 2023. URL: https://doi.org/10.1109/TIT.2023.3269790.

[DS17] Sabyasachi Dey and Santanu Sarkar. Improved analysis for reduced round Salsa and Chacha. *Discrete Applied Mathematics*, 227:58–69, 2017. URL: https://doi.org/10.1016/j.dam.2017.04.034.

[Goo] Google. URL: https://varindia.com/news/for-the-entry-level-smartphones-google-announced-a-new-encryption-solution--adiantum.

[IKM11] Tsukasa Ishiguro, Shinsaku Kiyomoto, and Yutaka Miyake. Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha. In Sihan Qing, Willy Susilo, Guilin Wang, and Dongmei Liu, editors, *Information and Communications Security*, pages 255–266, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-25243-3_21.

[LH94] Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages

17–25, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-48658-5_3.

[Mai16] Subhamoy Maitra. Chosen IV cryptanalysis on reduced round ChaCha and Salsa. *Discrete Applied Mathematics*, 208:88–97, 2016. URL: https://doi.org/10.1016/j.dam.2016.02.020.

[MIM22] Shotaro Miyashita, Ryoma Ito, and Atsuko Miyaji. PNB-Focused Differential Cryptanalysis of ChaCha Stream Cipher. In Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo, editors, *Information Security and Privacy*, pages 46–66, Cham, 2022. Springer International Publishing. URL: https://doi.org/10.1007/978-3-031-22301-3_3.

[MY93] Mitsuru Matsui and Atsuhiro Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, pages 81–91, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-47555-9_7.

[NL18] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 8439, June 2018. URL: https://www.rfc-editor.org/info/rfc8439.

[SD24] Nitin Kumar Sharma and Sabyasachi Dey. Analyzing the probability of key recovery in the differential attacks against chacha. *IEEE Access*, 12:37000–37011, 2024. URL: 10.1109/ACCESS.2024.3372857.

[SDSM25] Nitin Kumar Sharma, Sabyasachi Dey, Santanu Sarkar, and Subhamoy Maitra. On improved cryptanalytic results against chacha for reduced rounds ≥ 7. In Sourav Mukhopadhyay and Pantelimon Stănică, editors, *Progress in Cryptology – INDOCRYPT 2024*, pages 29–52, Cham, 2025. Springer Nature Switzerland.

[SZFW13] Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology – ICISC 2012*, pages 337–351, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-37682-5_24.

[WDL+25] Xinhai Wang, Lin Ding, Zhengting Li, Jiang Wan, and Bin Hu. Revisiting the differential-linear attacks on ChaCha from IEEE TIT and INDOCRYPT 2024 (extended abstract). Cryptology ePrint Archive, Paper 2025/206, 2025. URL: https://eprint.iacr.org/2025/206.

[WLHL23] Shichang Wang, Meicheng Liu, Shiqi Hou, and Dongdai Lin. Moving a Step of ChaCha in Syncopated Rhythm. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 273–304, Cham, 2023. Springer Nature Switzerland. URL: https://doi.org/10.1007/978-3-031-38548-3_10.

[XXTQ24] Zhichao Xu, Hong Xu, Lin Tan, and Wenfeng Qi. Differential-Linear Cryptanalysis of Reduced Round ChaCha. *IACR Transactions on Symmetric Cryptology*, 2024:166–189, 06 2024. URL: https://doi.org/10.46586/tosc.v2024.i2.166-189.