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, Hyderabad, India.

[2]Department of Mathematics, Indian Institute of Technology Madras, Chennai, India

[3]Applied Statistics Unit, Indian Statistical Institute, Kolkata, India

December 19, 2024

Table of Contents

Introduction

ChaCha Design

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 initial matrix looks as follows:

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

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 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 $Z = X \boxplus X^{(n)}$.

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:

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

Differential-Linear Cryptanalysis

Differential and Linear cryptanalysis are the primary attack methods against symmetric ciphers. In 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 denote 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 state, 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 linear bias up to R rounds is ϵ_l^2.

Therefore, the total bias value obtained for the differential-linear distinguisher used in the key-recovery attack is $\epsilon_d \cdot \epsilon_l^2$.

Probabilistic Neutral Bits (PNBs)

In 2008, Aumasson et al. [AFK+08] introduced a method to classify key bits into two groups: Probabilistically Neutral Bits (PNBs) and Non-PNBs. 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).

Consider an initial state matrix X. By introducing an appropriate non-zero input difference ($\mathcal{ID}$), we obtain a new state X'. 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

$$M = F^{-(R-r)}(Z - Y) \quad \text{and} \quad M' = F^{-(R-r)}(Z' - Y').$$

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:

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

To construct the set of Probabilistically Neutral Bits, 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 to get a set of Probabilistically Neutral Bits, which helps improve the time complexity.

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

$$\Pr\ [E_1(X) \oplus E_1(X') = \delta] = p.$$

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].

Techniques Used in this Work

⋆ Differential-Linear Hull Technique Proposed in [XXTQ24]

The 5-round differential-linear distinguisher is given below.

$$X_{15}^{(0)}[9], X_{15}^{(0)}[29]$$
$$\downarrow \ p$$
$$\Delta X_3^{(1)}[25,5] \oplus \Delta X_7^{(1)}[28,12] \oplus \Delta X_{11}^{(1)}[25,21] \oplus \Delta X_{15}^{(1)}[21,13]$$
$$\downarrow \quad \epsilon_d = 2^{-30.15}$$
$$\Delta X_2^{(3)}[4,3,0] \oplus \Delta X_7^{(3)}[20,4,0] \oplus \Delta X_8^{(3)}[20,19] \oplus \Delta X_{13}^{(3)}[4]$$
$$\downarrow \quad {\epsilon_l}^2 = 2^{-4}$$
$$\Delta X_2^{(5)}[0] \oplus \Delta X_6^{(5)}[7] \oplus \Delta X_6^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0].$$

In ToSC 2024, Xu et al. [XXTQ24] explained that the 5-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

$$\Delta X_2^{(3)}[4,3,0] \oplus \Delta X_7^{(3)}[20,4,0] \oplus \Delta X_8^{(3)}[20,19] \oplus \Delta X_{13}^{(3)}[4]$$

$$\downarrow$$

$$\Delta X_2^{(5)}[0] \oplus \Delta X_6^{(5)}[7] \oplus \Delta X_6^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0]$$

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 5-round differential-linear distinguisher is improved from $2^{-34.15}$ to $2^{-32.2}$.

⋆ Improved Algorithm for PNBs

In [DS17], an improved algorithm to find the probabilistically neutral bits was proposed. In this algorithm, 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 Conditional Probability Algorithm.

Conditional Probability Algorithm

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

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. In our Conditional Probability 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.

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 reduced round versions of ChaCha, as discussed in **Results and Comparison With Previous Works**.

⋆ Complexity Calculation as Explained in [Dey24]

In the recent approach proposed by [Dey24], the attack procedure is applied to a multi-bit output difference $\mathcal{OD}$ at specific positions.

Suppose the output difference is observed as a linear combination of multiple bits, $\mathcal{OD}_1, \mathcal{OD}_2, \ldots, \mathcal{OD}_k$. At 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 non-PNBs for 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 $\Pr_{nd} = 1.3 \times 10^{-3}$.

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

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

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

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 work, we revisited this formula in a disciplined manner and presented 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 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 $32 \times 32 \times 2 \times (\mathrm{R} - \mathrm{r}) = 2^{11} \times (\mathrm{R} - \mathrm{r})$. 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 value.

The modified formula based on this observation is given below:

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

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.

Improving the "p" Value

The probability value p of obtaining a right pair is 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.

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. Therefore, the data and time complexity values will be multiplied by $p = 2^{-5.2}$ instead of 2^{-7}.

Results and Comparison With Previous Works

In the timeline of the cryptanalysis against reduced-round versions of 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.

The 5-round differential-linear distinguisher is given below:

$$\mathcal{ID} : X_{15}^{(0)}[9], X_{15}^{(0)}[29].$$

$$\mathcal{OD} : (\Delta X_{2}^{(5)}[0] \oplus \Delta X_{6}^{(5)}[7] \oplus \Delta X_{6}^{(5)}[19] \oplus \Delta X_{10}^{(5)}[12] \oplus \Delta X_{14}^{(5)}[0]).$$

7-round

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.

We obtain a set of 169 PNBs, 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 $\epsilon_a = 0.00027$.

PNB Set for 7-round ChaCha

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, 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 number of PNBs and the biases ϵ_i for these $\mathcal{OD}_i$ positions are mentioned below.

i	1	2	3	4	5
$\mathcal{OD}_i$	$X_2^{(5)}[0]$	$X_6^{(5)}[7]$	$X_6^{(5)}[19]$	$X_{10}^{(5)}[12]$	$X_{14}^{(5)}[0]$
PNBs	41	58	41	29	37
ϵ_i	0.983	0.99	0.996	0.993	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 $N = 2^{95.89}$ 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 $C = 2^{172.92}$.

With $p^{-1} = 2^{5.2}$, 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.

# PNBs	Bias ϵ_a	Bias ϵ	Time Complexity	N
169	0.00027	$2^{-44.13}$	$2^{178.12}$	$2^{95.89}$

7.25-round

For an attack against 7.25-round ChaCha, we have to add more bits to the PNB set using the Conditional Probability Algorithm. In the recent attacks against 7.25-round ChaCha, Xu et al. [XXTQ24] use the PNB set of 133 bits. The bias $\epsilon_a = 2^{-11.25}$ for these 133 bits. We add 1 more bit $\{195\}$ in the PNB set using the post-processing step. The bias value is $\epsilon_a = 0.00025$ for the set of 134 PNBs given below:

PNB Set for 7.25-round ChaCha

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.

Using the technique mentioned in [Dey24], we obtain the PNB set for the 5 $\mathcal{OD}$ positions.

i	1	2	3	4	5
$\mathcal{OD}_i$	$X_2^{(5)}[0]$	$X_6^{(5)}[7]$	$X_6^{(5)}[19]$	$X_{10}^{(5)}[12]$	$X_{14}^{(5)}[0]$
PNBs	66	71	67	34	65
ϵ_i	0.986	0.978	0.972	0.983	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.

7.5-round

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 (5-round differential-linear distinguisher) 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 value with p^{-2} to compute the final data and time complexity.

Dey [Dey24] applied the concept of finding the PNB for $\mathcal{OD}$ bits applied to this $\mathcal{ID} - \mathcal{OD}$ pair. The 3 $\mathcal{OD}_i$ positions are $X_2^{(4)}[0]$, $X_8^{(4)}[0]$ and $X_7^{(4)}[7]$. The count of PNBs for the 3 $\mathcal{OD}_i$ positions 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 $\Pr_{nd} = 1.3 \times 10^{-3}$ 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 $2^{255.24}$.

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 $\epsilon_a = 0.0048$.

PNB Set for 7.5-round ChaCha

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 $\mathcal{OD}_i$'s $X_2^{(4)}[0]$, $X_8^{(4)}[0]$ and $X_7^{(4)}[7]$.

i	1	2	3
$\mathcal{OD}_i$	$X_2^{(4)}[0]$	$X_8^{(4)}[0]$	$X_7^{(4)}[7]$
PNBs	13	19	15
ϵ_i	0.848	0.97	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] ($\Pr_{nd} = 0.147$).

Using Equation 5, we obtain the value of $C = 2^{253.23}$. Therefore, the data and time complexity values are $2^{34.47}$ and $2^{253.23}$, respectively.

Conclusion and Future Works

In this paper, we propose a novel heuristic to construct improved sets of Probabilistically Neutral Bits and 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.

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. Using our methodology, we reduced the time complexity to $2^{253.23}$, with the potential for further improvements in future research. Our analysis of the 7.5-round ChaCha shows that introducing new techniques can further enhance the attack.

In this work, we observe that improvements to existing methods and the introduction of novel approaches are of significant importance in strengthening cryptanalysis against ChaCha.

References


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.


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.


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.


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.


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.


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.


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.


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.


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.


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.

THANK YOU EVERYONE