Significantly Improved Cryptanalysis of Salsa20 With Two-Round Criteria

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

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

[2] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700108, India
subho@isical.ac.in

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

Abstract. Over the past decade and a half, cryptanalytic techniques for Salsa20 have been increasingly refined, largely following the overarching concept of Probabilistically Neutral Bits (PNBs) by Aumasson et al. (FSE 2008). In this paper, we present a novel criterion for choosing key-$\mathcal{IV}$ pairs using certain 2-round criteria and connect that with clever tweaks of existing techniques related to Probabilistically Independent $\mathcal{IV}$ bits (earlier used for ARX ciphers, but not for Salsa20) and well-studied PNBs. Through a detailed examination of the matrix after initial rounds of Salsa20, we introduce the first-ever cryptanalysis of Salsa20 exceeding 8 rounds. Specifically, Salsa20/8.5, consisting of 256 secret key bits, can be cryptanalyzed with a time complexity of $2^{245.84}$ and data amounting to $2^{99.47}$. Further, the sharpness of our attack can be highlighted by showing that Salsa20/8 can be broken with time $2^{186.01}$ and data $2^{99.73}$, which is a significant improvement over the best-known result of Coutinho et al. (Journal of Cryptology, 2023, time $2^{217.14}$ and data $2^{113.14}$). Here, the refinements related to backward biases for PNBs are also instrumental in achieving the improvements. We also provide certain instances of how these ideas improve the cryptanalysis on 128-bit versions. In the process, a few critical points are raised on some existing state-of-the-art works in this direction, and in those cases, their estimates of time and data are revisited to note the correct complexities, revising the incorrect numbers.

Keywords: Salsa20 · Differential-Linear Cryptanalysis · Right Pair · Probabilistically Neutral Bits · Probabilistically Independent Bits.

1 Introduction

Symmetric-key encryption plays a major role in the protection of data by encrypting it using a secure key. Block and stream ciphers are the two main types of symmetric key encryption algorithms. Daniel J. Bernstein designed two very well-known stream ciphers and named them Salsa20 and ChaCha, following the names of Latin dances. Salsa20 [Ber08] was designed in 2005 and submitted to eSTREAM [eST], the ECRYPT Stream Cipher Project, as a candidate. It was selected as an efficient software-suitable candidate. This cipher is used in DNS implementations and messaging software like Viber and Discord. The operating system Chromium uses the scrypt KDF, which is based on Salsa20. Some cryptographic functions like scrypt, scrypt-jane, and nim-csprng exploit Salsa20 in their encryption algorithm. Many other applications of Salsa20 are mentioned in [Sal].

These two ciphers received a lot of interest from the cryptologic community immediately after the designs were proposed. This underlines two points. One, the ciphers received serious interest in being analyzed and are already used commercially. Two, it is becoming more challenging day by day to obtain improved results, the basic idea of the attack being the one presented in [AFK+08] long back. Surprisingly, while the attack complexity could be reduced over the years, the bound of 8 rounds for Salsa20 could not be breached for more than one and a half decades. In this paper, we could show the cryptanalysis for 8.5 rounds for the first time. The underlying technique being the same as in [AFK+08], as is true for all the following works in this direction, we could put together several existing techniques as well as introduce the idea of probabilistically independent $\mathcal{IV}$ bits to mount the attack. For 8 rounds, our results show significant improvements over the existing complexities in terms of time and data. In Table 1, the works [CPGV+22] and [CPV+23] are marked $\star$. The $\star$ mark indicates that the data complexities stated in the referenced papers require re-evaluation for accuracy. The re-evaluation of the data complexity is thoroughly explained in Section 5. Recalculating the $\star$-marked complexities in Section 5 is essential to highlight the extent of improvement achieved by our techniques clearly.

1.1 Cryptanalysis of Salsa20: A Brief History

Salsa20 is an ARX-based stream cipher that is mostly recommended for fast software implementation. ARX comprises three operations: bitwise modular addition, bitwise constant distance rotation, and bitwise XOR. Several authors provided different cryptanalytic techniques on such ciphers based on these operations, and such cryptanalytic efforts build confidence in such designs. Let us now explain in detail the history of such observations. Needless to say, none of the observations breach the designs with the number of recommended rounds. The efforts are to increase the rounds for which attacks are possible and reduce the complexities.

1. First, the 256-bit key version of Salsa20/5 was analysed by Crowley in 2005 [Cro05]. The attack works with an input difference to obtain a bias after 3 rounds and works 2 rounds backward after guessing 160 relevant key bits with time complexity 2^{165}.

2. In Indocrypt 2006, Fischer et al. [FMB+06] described an attack on Salsa20/6 with a time complexity of 2^{177}. This attack provides an input difference to generate a bias after 4 rounds and works 2 rounds backward after guessing 160 relevant key bits. In 2007, Tsunoo et al. [TSK+07] explained differential cryptanalysis on Salsa20/7 having a time complexity of 2^{184}.

3. In FSE 2008, Aumasson et al. [AFK+08] presented the first cryptanalysis of the 256-bit key version of Salsa20/8 by exploiting the 4-round forward differential and working 4 rounds in the backward direction. The time complexity for the attack on Salsa20/8 is 2^{251}. The authors provided a concept of Probabilistic Neutral Bits (PNBs). The idea of PNBs has been widely used in cryptanalytic techniques since then on Salsa20 and ChaCha. This paper is the most fundamental one, and this broad idea is being exploited with different variations to date in all the cryptanalytic results, including our present effort.

4. In 2012, Shi et al. [SZFW13] introduced the concept of Column Chaining Distinguisher (CCD), improving the attack on the 256-bit key version of Salsa20/R ($5 \leq R \leq 8$). The authors reduced the time complexity of Salsa20/7 and Salsa20/8 to 2^{148} and 2^{250}, respectively. In 2015, Maitra et al. [MPM15] improved cryptanalysis

against Salsa20/8 by reducing the time complexity to $2^{247.2}$.

5. In 2015, Maitra [Mai15] introduced a new idea of choosing proper $\mathcal{IV}$s corresponding to the keys. This concept helps reduce the time complexity for Salsa20/8 to $2^{245.5}$. This idea is being exploited with refinements in most of the recent attacks, as evident from [DGSS22].

6. In 2016, Choudhuri et al. [CM16] proposed the idea of multi-bit key differentials and provided an attack on Salsa20/R ($5 \leq R \leq 8$). For Salsa20/7 and Salsa20/8, the authors could reduce the time complexities to 2^{137} and $2^{244.9}$, respectively. In 2017, Dey et al. [DS17] improved the attack techniques for Salsa20/8, reducing the time complexity to $2^{243.7}$.

7. In Asiacrypt 2022, Coutinho et al. [CPGV+22] introduced a new technique called Bidirectional Linear Expansions (BLE) to improve the attack on the Salsa20/7 and Salsa20/8. This technique reduced the time complexity for Salsa20/7 and Salsa20/8 to 2^{137} and 2^{218}, respectively. Later, Coutinho et al. [CPV+23] slightly improved the result for Salsa20/7 and Salsa20/8. They could reduce the time complexities to $2^{125.16}$ and $2^{217.14}$, respectively. They also provided distinguishers for Salsa20/7 and Salsa20/8 with time complexity $2^{108.98}$ and $2^{215.62}$ respectively.

8. Recently, in 2024, Dey et al. [DLS24] introduced the method for creating a differential attack based on the linear combination of variables obtained from many differentials and improved the attack against the Salsa20/8 by reducing the time complexity to $2^{240.62}$.

Let us now revisit the results for the 128-bit secret key. The initial cryptanalysis of the 128-bit key version of Salsa20/5 was provided by Fischer et al. [FMB+06] in 2006. The authors cryptanalysed Salsa20/5 with a time complexity of 2^{81}. In 2007, Tsunoo et al. [TSK+07] provided key guessing results on Salsa20/6 and Salsa20/7. In FSE 2008, Aumasson et al. [AFK+08] used the 4-round differential to provide an attack on the 128-bit key version of Salsa20/7. The time complexity for this attack was 2^{111}. In 2012, Shi et al. [SZFW13] reduced the time complexity of Salsa20/7 to 2^{109}. Later, in 2018, Deepthi et al. [DS18] provided an attack on Salsa20/7 with a time complexity of 2^{107}. In 2024, Dey et al. [DLS24] reduced the time complexity of Salsa20/7 to $2^{102.82}$ and provided the first ever attack on Salsa20/7.5 with time complexity $2^{124.22}$. Works in similar directions are also explored on ChaCha, and many attack ideas mentioned for one cipher can also be applied in the context of other ciphers by cryptanalysts. Table 1 provides an overview of existing key-recovery attacks on the 128 and 256-bit key versions of Salsa20, along with the improvements achieved through our techniques.

1.2 Organization & Contribution

This paper provides improved attacks on Salsa20, introducing several new ideas. Our attack not only improves the existing ones in terms of time and data complexities but also produces an attack on the 256-bit key version of Salsa20/8.5 for the first time. After the attack on Salsa20/8 in 2008, this is the first time a partial round improvement in the key-recovery attack could be produced against the 256-bit key version of Salsa20.

Understanding the attacks requires a detailed explanation of the cipher and is given in Subsection 1.3. We try to deconstruct each small step carefully to identify new techniques. Consequently, we explain the basic understanding of differential-linear cryptanalysis, describe the idea of a right key-$\mathcal{IV}$ pair, Probabilistically Neutral Bits (PNBs), and the existing results related to PNBs in Section 2. These are the necessary background materials to interweave the existing techniques with our novel findings in this paper. Let us now

Table 1: Comparison of Our Attack Complexities With the Existing Attacks on Salsa20. ($\star$ mark indicates the attack is invalid because data complexity is greater than 2^{96})

Key/Round	Time	Data	Ref.
128/7	2^{111}	2^{21}	[AFK+08]
	2^{109}	2^{19}	[SZFW13]
	2^{107}	2^{24}	[DS18]
	$2^{102.82}$	$2^{28.77}$	[DLS24]
	$2^{98.25}$	$2^{86.67}$	Subsection 4.3
128/7.5	$2^{124.22}$	$2^{23.06}$	[DLS24]
	$2^{111.47}$	$2^{94.16}$	Subsection 4.4
256/8	2^{251}	2^{31}	[AFK+08]
	2^{250}	2^{31}	[SZFW13]
	$2^{247.2}$	2^{31}	[MPM15]
	$2^{245.5}$	2^{31}	[Mai15]
	$2^{244.9}$	2^{96}	[CM16]
	$2^{243.7}$	-	[DS17]
	$2^{240.62}$	$2^{27.56}$	[DLS24]
	2^{218}	2^{114}	[CPGV+22]*
	$2^{217.14}$	$2^{113.14}$	[CPV+23]*
	$2^{186.01}$	$2^{99.73}$	Subsection 4.1
256/8.5	$2^{245.84}$	$2^{99.47}$	Subsection 4.2

enumerate the section-wise contributions of this paper.

- In Section 3, we propose two ideas that improve the state-of-the-art attack parameters against Salsa20. The first idea (Subsection 3.1) introduces a criterion in the 2nd round to improve the bias. The second idea (Subsection 3.2) introduces the idea of the probabilistically independent $\mathcal{IV}$ bits, which ensures that sufficient data can be generated to execute the attack using the first idea.
- The key-recovery attacks on 256-bit key version of Salsa20/8 and Salsa20/8.5 and 128-bit key version of Salsa20/7 and Salsa20/7.5 are mentioned in Section 4. For the 256-bit key version of Salsa20/8, our proposed time complexity value is $2^{186.01}$, which improves the previous result [CPV+23] by a significant margin of more than 2^{31}. If we take note of the invalidity of the attack in [CPV+23] due to the data limit as explained in Section 5, the margin is even better. We would like to reiterate that, for the first time after around two decades of research, we could identify cryptanalysis of Salsa20 for more than 8 rounds. We also significantly improve the complexities for the 128-bit key version of Salsa20/7 and Salsa20/7.5.
- Section 5 comprises the critical analyses of data complexity calculation for a few previous attacks on Salsa20 and consequently shows that some attacks are not achievable as claimed.

Finally, we conclude this paper in Section 6.

Table 2: Table of Notations.

Notation	**Meaning**
X	The state matrix of the cipher consisting of 16 words
$X^{(0)}$	Initial state matrix
$X^{(r)}$	State matrix after r rounds
$X_i^{(r)}[j]$	Value of j-th bit of the i-th word of $X^{(r)}$
$\Delta X_i^{(r)}[j]$	XOR difference at the j-th bit of the i-th word of states $X^{(r)}$ and $X'^{(r)}$
Salsa20/R	Salsa20 reduced to R rounds.
$\mathcal{ID}$	Input Difference position
$\mathcal{OD}$	Output Difference position
ϵ_d	Bias obtained after r rounds in forward direction
$\tilde{X}$	State matrix in which significant key bits have fixed values
α	Parameter for finding significance level in hypothesis testing
ϵ_a	Bias obtained after $(R-r)$ rounds in backward direction
$v_i[j]$	Value of j-th bit of the i-th word of $\mathcal{IV}$ v
HW	Hamming weight of the difference matrix $X \oplus X'$
(k, v)	Key-$\mathcal{IV}$ pair

1.3 Description of Salsa20

Salsa20 works on sixteen 32-bit words represented in the form of a 4×4 matrix. The cipher has two versions each, having keys of sizes 128 and 256 bits. The 256-bit key version of the cipher takes 8 key words $(k_0, k_1, \ldots, k_7)$, 4 constants words (c_0, c_1, c_2, c_3), 2 words of nonce (v_0, v_1) and 2 counter words (v_2, v_3) as input and generates a 512-bit output. In this paper, for simplicity, the nonces and counter words are considered as $\mathcal{IV}$s. The constants words (c_0, c_1, c_2, c_3) are slightly different for 128-bit and 256-bit key versions of cipher. The four constants for the 256-bit key structure are

$$c_0 = \texttt{0x61707865}, c_1 = \texttt{0x3320646e}, c_2 = \texttt{0x79622d32}, c_3 = \texttt{0x6b206574}.$$

For the 128-bit key structure, the key words $(k_0, k_1, \ldots, k_7)$ are defined as $k_{i+4} = k_i,\ \forall\ 0 \leq i \leq 3$. There is a slight change in the constants for the 128-bit key structure. The four constants for the 128-bit key structure are

$$c_0 = \texttt{0x61707865}, c_1 = \texttt{0x3120646e}, c_2 = \ \texttt{0x79622d36}, c_3 = \texttt{0x6b206574}.$$

The keywords $(k_0, k_1, \ldots, k_7)$ are arranged in all the rows of the matrix. Nonces (v_0, v_1) and counter words (v_2, v_3) are positioned in the second and third rows respectively. The constants words (c_0, c_1, c_2, c_3) are placed along the diagonals. Based on the discussion above, the initial state matrix looks as follows:

$$X = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} & X_{11} \\ X_{12} & X_{13} & X_{14} & X_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ v_2 & v_3 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}.$$

The round function is a nonlinear operation consisting of ARX operation viz.: addition modulo 2^{32} ($\boxplus$), left cyclic rotation operation ($\lll$), and XOR operation between the bits ($\oplus$). In this round function, the vector (a, b, c, d) transforms into a vector (a', b', c', d') as shown below:

$$\begin{aligned} b' &= b \oplus ((a \boxplus d) \lll 7), \\ c' &= c \oplus ((b' \boxplus a) \lll 9), \\ d' &= d \oplus ((c' \boxplus b') \lll 13), \\ a' &= a \oplus ((d' \boxplus c') \lll 18). \end{aligned} \tag{1}$$

This function described in Equation 1 is known as the quarter round function. For an initial state matrix $X^{(0)}$, after applying the round function r times, we obtain a state matrix $X^{(r)}$. For odd rounds, the round function acts along the columns of the matrix and is called the column-round function. The four columns are (X_0, X_4, X_8, X_{12}), (X_5, X_9, X_{13}, X_1), $(X_{10}, X_{14}, X_2, X_6)$ and $(X_{15}, X_3, X_7, X_{11})$. For even rounds, the round function acts along the rows of the matrix and is called the row-round function. The four rows are (X_0, X_1, X_2, X_3), (X_5, X_6, X_7, X_4), $(X_{10}, X_{11}, X_8, X_9)$ and $(X_{15}, X_{12}, X_{13}, X_{14})$. The keystream block Z is obtained by the addition of matrices $X^{(0)}$ and $X^{(r)}$ as shown:

$$Z = X^{(0)} \boxplus \; X^{(r)},$$

where $X^{(0)}$ is the initial state and $X^{(r)}$ is the state after r rounds of X.

Half of quarter round:

We define a half-round by applying the first two operations of the quarter-round functions on each row/column, i.e.,

$$\begin{aligned} b' &= b \oplus ((a \boxplus d) \lll 7), \\ c' &= c \oplus ((b' \boxplus a) \lll 9). \end{aligned} \tag{2}$$

After implementing Equation 2, only 8 words in the state matrix are updated. The elements b and c of the vector are transformed to b' and c'. The updated words are underlined in the right-side matrix shown below:

$$\text{Salsa20/8} \quad \longrightarrow \quad \text{Salsa20/8.5}$$

$$\begin{pmatrix} X_0^{(8)} & X_1^{(8)} & X_2^{(8)} & X_3^{(8)} \\ X_4^{(8)} & X_5^{(8)} & X_6^{(8)} & X_7^{(8)} \\ X_8^{(8)} & X_9^{(8)} & X_{10}^{(8)} & X_{11}^{(8)} \\ X_{12}^{(8)} & X_{13}^{(8)} & X_{14}^{(8)} & X_{15}^{(8)} \end{pmatrix} \longrightarrow \begin{pmatrix} X_0^{(8.5)} & X_1^{(8.5)} & \underline{X_2^{(8.5)}} & \underline{X_3^{(8.5)}} \\ \underline{X_4^{(8.5)}} & X_5^{(8.5)} & X_6^{(8.5)} & \underline{X_7^{(8.5)}} \\ \underline{X_8^{(8.5)}} & \underline{X_9^{(8.5)}} & X_{10}^{(8.5)} & X_{11}^{(8.5)} \\ X_{12}^{(8.5)} & \underline{X_{13}^{(8.5)}} & \underline{X_{14}^{(8.5)}} & X_{15}^{(8.5)} \end{pmatrix}.$$

Xu et al. [XXTQ24] introduced attacks on the 7.25 and $7.5^{\oplus}$-round versions of ChaCha, where updates are limited to only 8 words of the cipher state (same as our case). The $7.5^{\oplus}$-round version of ChaCha adopts four addition operations over the existing 7.25-round version. In earlier work on Salsa20 and ChaCha, such partial-round attacks have proven advantageous in extending the analysis to full-round attacks later. Moreover, we have observed that there is a possibility to provide an attack on Salsa20/8.75.

2 Existing Cryptanalytic Techniques

In this section, we discuss some existing cryptanalytic techniques that will be used while we build our attacks. Our novel ideas are mentioned in Section 3, which will be connected with these known techniques to mount more efficient attacks.

2.1 Differential-Linear Cryptanalysis

The two main attack techniques against symmetric ciphers are differential and linear cryptanalysis. Biham and Shamir [BS91] introduced the concept of differential cryptanalysis against DES in 1990. Differential cryptanalysis on DES analyzes how chosen differences in plaintext pairs create predictable differences in the resulting ciphertext pairs. By carefully selecting these plaintext pairs and tracking how their differences change through each round, especially through the S-boxes, the attacker can identify patterns that suggest possible encryption key values. In 1992, Matsui [MY93] introduced linear cryptanalysis against FEAL. Using the technique, the attacker constructs one or more linear relation(s) between plaintext, ciphertext, and key bits in this XOR ($\oplus$) operation. In 1994, Langford and Hellman [LH94] combined the differential and linear attack techniques and introduced the concept of differential-linear cryptanalysis. This technique was first applied to DES (a block cipher) but was later successfully extended to cryptanalyse stream ciphers. In this technique, the cipher E is split into two subciphers, E_1 and E_2. The subcipher E_1 is exploited for the differential cryptanalysis, whereas E_2 is considered for the linear approximation.

Let us now recall the differential-linear cryptanalysis on Salsa20. Let X be the initial state matrix and X' be the state matrix different from X only at a single bit ($\mathcal{ID}$). Here, $X_i[j]$ denotes the j-th bit of the i-th word of the matrix X. After employing the difference at the initial stage, we observe the output difference after r rounds. The difference observed is denoted as $\Delta X_p^{(r)}[q] = X_p^{(r)}[q] \oplus X_p'^{(r)}[q]$. The q-th bit of the p-th word, for which the probability of output difference $\Delta X_p^{(r)}[q] = 0$ is high, is considered an output difference position ($\mathcal{OD}$). The probability is given as $\frac{1}{2}(1 + \epsilon_d)$, where ϵ_d denotes the output-difference bias, also known as forward bias.

After noting the r-round output difference, we have to find a linear relationship between the output differential from r rounds of the cipher to R rounds. The bias for linear approximation is given by the notation ϵ_l. Since the linear relation is observed for X and X', the differential-linear bias for R rounds is expressed as $\epsilon_d \cdot \epsilon_l^2$.

2.2 Searching for a Right Pair

In [Mai15], Maitra explained that if a particular value of words in the difference column is chosen, one may obtain a better forward bias ϵ_d. To maximize ϵ_d, the attacker carefully chooses the difference column words so that the propagation of differences after the one-quarter round is controlled and results in a fixed number of differences. This precise selection ensures the desired level of influence on the differential propagation. Let the fixed value of differences after the one-quarter round be h. Instead of choosing any random $\mathcal{IV}$, we can locate a higher bias if we choose specific $\mathcal{IV}$s for which the number of differences after the first quarter round is exactly h.

Let us consider the initial states $(X, X \oplus \Delta_{in})$ such that $\Pr_{X \in \mathbb{F}_2^n}[E_1(X) \oplus E_1(X \oplus \Delta_{in}) = \Delta_m] = p$, where p is the probability of obtaining a right pair when the initial states are chosen at random. The probability for the key is $\Pr_{X_i \in \mathbb{F}_2^n}[\ E_1(X_i) \oplus\ E_1(X_i \oplus \Delta_{in}) = \Delta_m] = p \approx \frac{1}{2}$, i.e., to find a right pair we have to evaluate p^{-1} iterations. We have improved this idea of obtaining a right pair by adding some extra conditions, as mentioned in Subsection 3.1.

2.3 General Idea of Probabilistic Neutral Bits

Aumasson et al. [AFK$^+$08] proposed a technique to divide the key bits into two categories: Probabilistically Neutral Bits (PNBs) and Significant bits. PNBs have a low probability of influencing the output difference bit. Therefore, if out of 256 bits, m bits are significant, and the remaining $(256 - m)$ are probabilistically neutral, we can guess only 2^m values to find the m significant key bits first.

Consider X as the initial state matrix, with X' being a state matrix obtained by introducing an input difference ($\mathcal{ID}$). Upon executing the algorithm on X and X' for r rounds, we observe the difference between the two state matrices at a single bit. This difference is represented as $\Delta X_p^{(r)}[q]$ and termed the output difference ($\mathcal{OD}$). The bias obtained after r rounds is denoted as ϵ_d. Upon completing R rounds, we obtain the final states $X^{(R)}$ and $X'^{(R)}$ respectively. These states are then added with the respective initial states X and X' to generate the keystream blocks Z and Z', respectively.

Now concentrate on a specific key bit, say k_i, out of the total key bits (128 or 256). After changing the value of k_i in the initial states X and X', we obtain the new altered states as $\bar{X}$ and $\bar{X}'$. We obtain the matrices $Z - \bar{X}$ and $Z' - \bar{X}'$ by applying the reverse round function $(R - r)$ times. These two matrices are denoted by $\bar{M}$ and $\bar{M}'$. The key bit k_i will be considered a PNB if the probability of an event $\Delta\bar{M}_p[q] = \Delta X_p^{(r)}[q]$ is high. The bias of this event is denoted by γ_i, also known as the neutrality measure. In [AFK+08], to construct the PNB set, the authors have used a threshold value γ such that a key bit k_i for which $\gamma_i \geq \gamma$ is regarded as probabilistically neutral. Once we have the PNBs, during the actual attack, the attacker assigns random values in the PNBs, guesses the significant bits only, and runs the algorithm backward. When the guess of significant bits is correct, it shows a high bias, which acts as an alarm to identify the correct guess. For a detailed explanation of this process, one may go through [AFK+08] or [Mai15].

3 Our Proposed Ideas for Cryptanalysis

In [BLT20, Mai15], the authors used the idea of minimizing the difference propagation in the first round in order to improve the bias in the distinguisher. This idea was simple and effective because, in the first round, the columns were updated independently. Therefore, random values in the three other columns do not affect the validity of a right pair. While it is natural that minimizing the difference propagation would improve the bias, using this idea in the second round is infeasible because the row-round (followed by the column-round) mixes up the entire matrix. Therefore, a state with minimum difference can be disturbed by any change of bit. So, throughout an attack, such a state can not be maintained. For this reason, improving the bias further by imposing a criterion in the second round has been challenging since 2016.

In this context, we introduce two ideas in Subsection 3.1 and Subsection 3.2 here in order to improve the distinguisher. In the first one, by analyzing the design and the difference propagation, we introduce a small criterion in the second round, which increases the bias significantly. The second idea is regarding probabilistically independent $\mathcal{IV}$s, which is a refinement of an idea proposed in [BLT20] against Chaskey and assures that a sufficient number of $\mathcal{IV}$s can be generated without disturbing the right pair. Thus, an idea from the cryptanalysis of a different cipher is brought in here for cryptanalysing Salsa20.

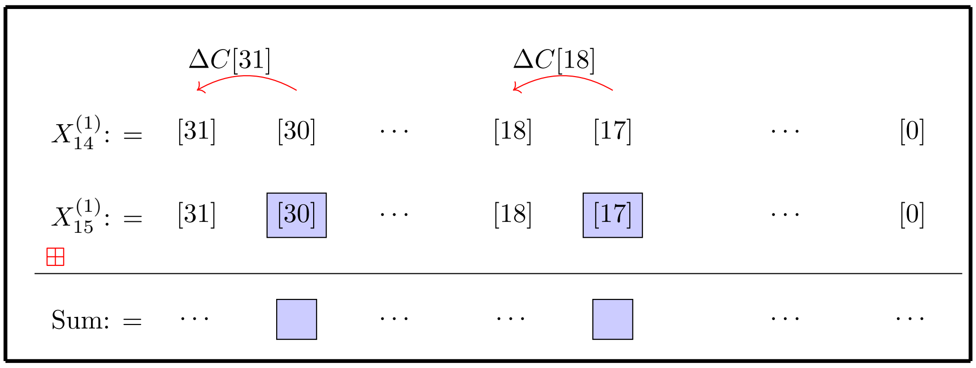
3.1 Introducing a 2-round criterion for Finding a Right Pair by Structural Analysis of Difference Propagation

In this technique, we modify the criterion for a key-$\mathcal{IV}$ pair to be a right pair. In the attack against Salsa20 in [CPV+23], the authors put the input difference at the 31-st bit of X_7 and observed the difference after 5 rounds at $X_4[7]$. To estimate the bias, they expressed $X_4^{(5)}[7]$ as a linear combination of 3 bits, $X_0^{(4)}[0], X_4^{(4)}[7], X_{12}^{(4)}[0]$. Keeping the previously existing condition of four differences after the first round, we introduce one more condition in the second round. In the second round, the first operation in the last row is

$$X_{12}^{(2)} = X_{12}^{(1)} \oplus \Big((X_{14}^{(1)} \boxplus X_{15}^{(1)}) \lll 7 \Big).$$

Note that X_{15} contains two differences after the first round (as shown in Figure 1). Following the operation mentioned above, these differences automatically arrive at $X_{12}^{(2)}$.

Whether these differences would propagate to the adjacent bits is probabilistic. In Figure 1, the terms $\Delta C[31]$ and $\Delta C[18]$ are the differences generated at the carry bits during the addition $X_{14}^{(1)} \boxplus X_{15}^{(1)}$. $\Delta C[i] = 0$ implies that there is no carry difference. Now, whether $\Delta C[18]$ is 0 or 1 depends on the value of $X_{14}[17]$. In order to minimize the difference propagation, we want to make both $\Delta C[18]$ and $\Delta C[31]$ zero. This would result in only two difference bits in $X_{12}^{(2)}$. So, we are interested in only those states where the propagation does not occur. Therefore, the criterion for a key-$\mathcal{IV}$ pair to be a right pair in our attack is to satisfy both of the following criteria:



$\Delta C[31]$
$\Delta C[18]$
$X_{14}^{(1)}: =$ [31] [30] ··· [18] [17] ··· [0]
$X_{15}^{(1)}: =$ [31] [30] ··· [18] [17] ··· [0]
$\boxplus$
Sum: = ··· ··· ··· ··· ···

Figure 1: Difference Propagation in the Addition Operation $X_{14}^{(1)} \boxplus X_{15}^{(1)}$.

1. Exactly 4 differences after the first round, i.e., $HW(X^{(1)} \oplus X'^{(1)}) = 4$.
2. Exactly 2 differences in X_{12} after 2 rounds, i.e., $HW(X_{12}^{(2)} \oplus X_{12}'^{(2)}) = 2$.

A comparison of the biases for all the 32 bits of X_{12} for the 1-round criterion [CPV+23] and the 2-round criterion (this paper) is given in Figure 2. The bias is represented by the intensity of the black color in the box, where full white represents the bias value 1 and full black represents the bias value -1. One can observe that in the older approach (above), from the 24-th bit, the bias gradually becomes less significant as we move towards the left. Bit 25 has a zero bias, and bits on its left have decreasing biases. Further, in bit 6, the bias is 0, and the bias for bit 7 is 1. In our approach (bottom), the difference is at exactly 2 bits, i.e., the biases are -1 at bits 24 and 5, and in the rest of the bits, the bias is $+1$.

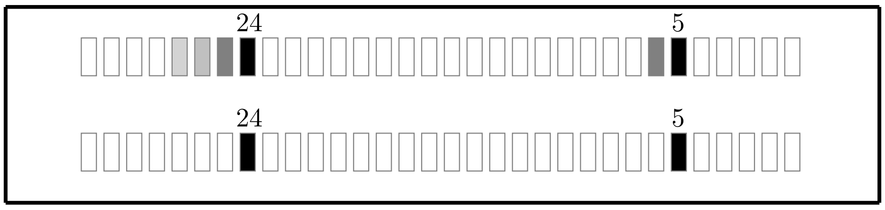


Figure 2: Biases of 32 bits of X_{12} For Both 1-Round (Top) and 2-Round Criterion (Bottom).

This difference of biases between the 1-round criterion and 2-round criterion propagates to other words as the algorithm continues. We find out the bias of $\Delta X_p[q]$ of all 32 bits of each of the words of the 4-th column, (i.e., $q \in \{0, 1, \dots 31\}, p \in \{12, 13, 14, 15\}$) both for the existing 1-round criterion of [CPV+23] and 2-round criterion given by us after completion of 1st and 2nd round respectively. In Figure 3, for each of those words, we provide a side-by-side comparison of the average bias of all 32 bits of that word between the existing 1-round criterion of [CPV+23] and 2-round criterion of us to show the improvement on introducing the 2-round criterion. The output difference is observed at the linear combination of 3 bits. Imposing these two conditions, we observe significant improvement in the forward biases.

Significance of the 2-round Criteria in Estimation of Bias

In [CPV+23, Computational Result 2 Section 4.2], Coutinho et al. estimated the bias ϵ_d of $\Delta X_0^{(4)}[0] \oplus \Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0]$ to be $2^{-42.01}$. However, they mentioned, “In our tests, the observed correlation was always higher than predicted. Therefore, our attack using this correlation is probably better than what we report in this paper". To identify the accurate bias, we need to experiment over a larger sample, which is computationally infeasible. This is why achieving a good estimation using some theoretical approach is important. In this context, our analysis of the 2-round condition is significant for two reasons. Firstly, this gives a significantly improved

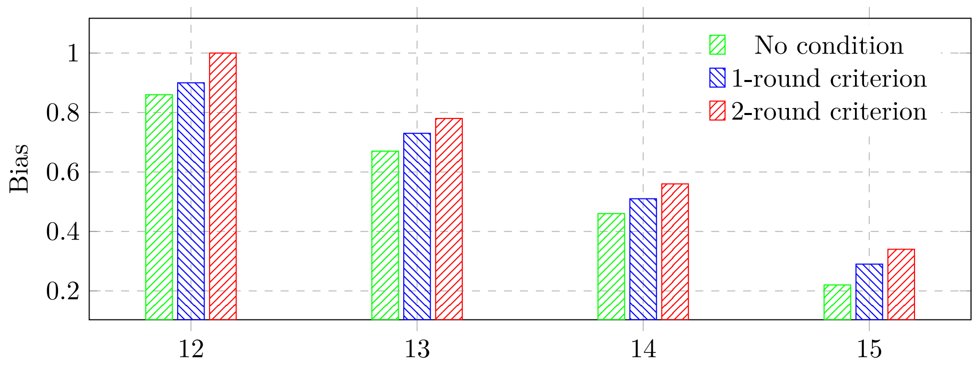


Figure 3: Comparison of average biases over all the 32 bits for each word $X_{12}^{(2)}$, $X_{13}^{(2)}$, $X_{14}^{(2)}$, $X_{15}^{(2)}$ for no condition, 1-round criterion [CPV+23] and 2-round criterion.

bias, estimated to be $2^{-33.75}$. Secondly, based on this finding, we can also estimate the bias for the 1-round condition, which is $2^{-35.75}$. This is also significantly better than the estimation of [CPV+23].

The individual correlation values for the three OD positions $\Delta X_0^{(4)}[0]$, $\Delta X_4^{(4)}[7]$ and $\Delta X_{12}^{(4)}[0]$ were obtained by Coutinho et al. [CPV+23] by implementing the 1-round criteria. Using our 2-round criteria, we provide the correlation/bias values for ϵ_d in these three OD positions. In Table 3, we provide the comparison of the "bias for the 1-round criterion of examining the difference before the completion of the first round" against the "bias obtained for the 2-round criterion of checking the differences before the completion of the first and second rounds. We can use the Piling-Up Lemma to estimate the bias value for the OD position $X_4^{(5)}[7] = \Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] \oplus \Delta X_0^{(4)}[0]$.

Table 3: Comparison of Previous and New Biases Due to the Change in the Condition of Searching for a Right Pair.

i	$\mathcal{OD}$	Bias (ϵ_d)	
		1-round criterion	2-round criterion
1	$\Delta X_0^{(4)}[0]$	$0.00000159/\ 2^{-19.26}$	$0.0000053/2^{-17.52}$
2	$\Delta X_4^{(4)}[7]$	$0.00085/2^{-10.21}$	$0.00105/2^{-9.9}$
3	$\Delta X_{12}^{(4)}[0]$	$0.000167/2^{-12.54}$	$0.00035/2^{-11.48}$
	Bias	$2^{-42.01}$	$2^{-38.90}$

To provide a better estimation of these biases mentioned in Table 3, we obtain the bias of $(\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0)$ and observe the improvement in the overall bias value ϵ_d. Note that Coutinho et al. [CPV+23] mentioned that the correlation values are probably better than what was claimed in their work. We have found the bias values computationally for both 1-round criteria and 2-round criteria, as shown below:

Table 4: Comparison of Bias Value for $\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0]$ Using Piling Up Lemma and Experiment

Conditions	$\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0]$	
	Piling Up Lemma	Experiment
1-round criterion	$0.00000014195/2^{-22.74}$	$0.0000060/2^{-17.34}$
2-round criterion	$0.00000003675/2^{-24.69}$	$0.000013/2^{-16.23}$

Since Salsa20 and ChaCha have similar design algorithms, we also studied some recent works on ChaCha in the context of obtaining improved results. Very recently, in TOSC24, Xu et al. [XXTQ24] provided an improvement in the bias for ChaCha by implementing the differential-linear hull effect on certain existing works on ChaCha. Similarly, we analyzed the work of [CPGV$^+$22, CPV$^+$23] on Salsa20 and improve the biases for the OD positions $\Delta X_0^{(4)}[0]$, $\Delta X_4^{(4)}[7]$ and $\Delta X_{12}^{(4)}[0]$ by providing a better estimation to calculate the bias of $\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0$. In Table 4, we provide the improvement bias for $\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0$ over the existing results, and hence using our techniques, we can improve the complexity values of the work [CPV$^+$23].

We find the bias of $(\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0)$ and $\Delta X_0^{(4)}[0]$ separately by experiment. We use 2^{41} random keys for this and achieved the biases $1.3 \times 10^{-5} \approx 2^{-16.23}$ and $0.0000053 \approx 2^{-17.52}$ respectively. The source code for obtaining the forward bias for $\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0]$ is provided in the GitHub link [Sha24]. Then we use the piling-up lemma to estimate the bias of $(\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0) \oplus \Delta X_0^{(4)}[0]$, leading to an estimated ϵ_d of $2^{-33.75}$. This is around $2^{8.25}$ times higher than the bias in [CPV$^+$23].

Careful Revision of p

The probability of achieving a right pair in the previous approach was $\frac{1}{2}$. Since the restriction has been stronger in this attack, it reduces the probability of certain events. Usually, the probability that a difference would propagate to the next bit is $\frac{1}{2}$, as mentioned in [DDSM22]. Since we have a difference at two bits, an extra fraction of $(\frac{1}{2})^2$ is multiplied by the value of p, making it $\frac{1}{8}$. Hence, the value of p^{-1} changes from 2 to 8. Though this increases the complexity values by some margin, because of the significant increase in the value of ϵ_d, the overall complexity of the attack is reduced by a good margin, which will be shown in Section 4.

Better Estimation of the Bias ϵ_d for 1-Round Criteria

As explained above, the final bias value $\epsilon_d = 2^{-33.75}$ is obtained through the evaluation of the bias for $(\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0)$. Then, we proceed by combining its value with the bias of $\Delta X_0^{(4)}[0]$ by using Pilling-up Lemma. After evaluating the bias value, we note that the same mentioned in [CPV$^+$23] can be improved through our analysis. The bias value is only for an event with probability $\frac{1}{4}$ times what we consider in [CPV$^+$23]. Hence the attack considered in [CPV$^+$23] is bias value can be at least $\epsilon_d = \frac{1}{4} \times 2^{-33.75} = 2^{-35.75}$.

3.2 Probabilistically Independent $\mathcal{IV}$ Bits

As we introduce a new criterion in X_{12} on the second round to form a right pair, by the existing approach, the attacker can change only 64 $\mathcal{IV}$ bits (X_8 and X_9), i.e., the upper bound of N now becomes 2^{64} only. To overcome this bound of data complexity of the attack, we introduce the idea of probabilistically independent $\mathcal{IV}$ bits. This idea is a tweak of an idea used in [BLT20] in order to produce an attack against Chaskey. We refined it based on our requirement for analyzing Salsa20.

For any $\mathcal{IV}$ v, let us denote by $\hat{v}$ the $\mathcal{IV}$ achieved by flipping $v_i[j]$. Consider a right pair (k, v). If we note that $(k, \hat{v})$ is also a right pair with high probability, then we refer to $v_i[j]$ as a probabilistically independent bit. To identify such bits, we assign a threshold probability β (in the similar line of threshold probability in the case of PNBs). Now, for any $\mathcal{IV}$ bit $v_i[j]$, we compute the conditional probability $\Pr[(k, \hat{v}) \text{ is right pair} \mid (k, v) \text{ is right pair}]$ by experimenting over a large number of randomly chosen keys and $\mathcal{IV}$'s. In our case, the number of randomly chosen keys and IVs is 2^{17}. We call this probability the "measure of independence." If the measure of independence is higher than the threshold β, we consider $v_i[j]$ as a probabilistically independent $\mathcal{IV}$ bit.

A formal definition is written as follows, and the detailed procedure is given in Algorithm 1.

Definition 1. For a given threshold β, an $\mathcal{IV}$ bit $v_i[j]$ is probabilistically independent if $\Pr[(k, \hat{v}) \text{ is right pair} \mid (k, v) \text{ is right pair }] > \beta$, where $\hat{v}$ is achieved by flipping the bit $v_i[j]$ of v.

Algorithm 1: Algorithm to Find Probabilistically Independent $\mathcal{IV}$ bits.

```
Input: $\beta$: A threshold probability, $LOOP$: the number of iterations to be
       performed.
for each $\mathcal{IV}$ bit $v_i[j]$ do
    $count_{v_i[j]} = 0$.
    for loop = 1 to LOOP do
        while true do
            take a random matrix $X$.
            prepare $X'$ by putting the difference at the 31-st bit of the 7-th word.
            compute $X^{(1)}$, $X'^{(1)}$, $X^{(2)}$, $X'^{(2)}$.
            if $(HW(\Delta X^{(1)}) = 4$ and $HW(\Delta X^{(2)}_{12}) = 2)$ then
                break
            end
        end
        Prepare $\hat{X}, \hat{X}'$ by complementing $v_i[j]$ in $X$ and $X'$.
        compute $\hat{X^{(1)}}, \hat{X'^{(1)}}, \hat{X^{(2)}}, \hat{X'^{(2)}}$.
        if $(HW(\Delta \hat{X^{(1)}}) = 4$ and $HW(\Delta \hat{X^{(2)}_{12}}) = 2)$ then
            increase counter $count_{v_i[j]}$.
        end
    end
    if $\frac{count_{v_i[j]}}{LOOP} > \beta$ then
        include $v_i[j]$ in the set.
    end
end
```

Modifying the Attack Procedure

During the data collection phase of the attack (keystream generation), for any $\mathcal{IV}$, the attacker generates different $\mathcal{IV}$'s by changing the values of the probabilistically independent bits. For an $\mathcal{IV}$ v, which forms a right pair with a key k, if we randomly change the values of the probabilistically independent bits and produce a new $\mathcal{IV}$ $\tilde{v}$, the probability that $\tilde{v}$ forms a right pair with the same key k is high. If there are n probabilistically independent $\mathcal{IV}$ bits, the attacker can generate 2^n possible $\mathcal{IV}$s, thus 2^n possible keystreams. This improves the restriction on the upper bound of N. Experimentally, we find out the probability $\Pr[(k, \tilde{v}) \text{ is right pair} \mid (k, v) \text{ is right pair}]$. Let us call this probability p_1. Therefore, if we generate N different keystreams by changing the probabilistically independent $\mathcal{IV}$ bits out of N (k, v) pairs, $p_1 \times N$ will form the right pair, the remaining $(1 - p_1) \times N$ will not. Therefore, for these $(1 - p_1) \times N$ pairs, we do not expect any bias. On average, out of N pair of matrices, $\left[p_1 \times \frac{N(1+\epsilon)}{2} + (1 - p_1) \times \frac{N}{2}\right] = \frac{N}{2}(1 + p_1 \cdot \epsilon)$ will satisfy the desired output difference. This results in a change in the formulation of data complexity value mentioned in previous works [CPV+23].

We have written the program on the toy version (introduced in [DGM23]) in order to verify the validity of the formula. Out of 2^{20} experiments, we detected the correct key with 100% success. The experiment took less than 5 min to complete. Hence, we have experimentally verified the formula of N (Equation 4) for the toy version, and it is valid. The source code for this experiment is given in the GitHub link [Sha24].

3.3 Estimation of Attack Complexity

Here, we estimate the complexity values of the complete attack. The explanation here is not new, but we must refer to it clearly for two reasons.

- To explain the achievements of our approach in terms of complexity (as in Section 4).

- To critically revisit certain complexity calculations of existing research papers in Section 5.

First, one tries to discover the values of the significant key bits. In the initial states X and X', we fix guessed values in those. The new initial state matrices obtained are $\tilde{X}$ and $\tilde{X}'$. The state matrices $\tilde{M}$ and $\tilde{M}'$ are then obtained by applying the reverse round function to $Z - \tilde{X}$ and $Z' - \tilde{X}'$.

A bias ϵ_a is observed for the event $(\Delta\tilde{M}_p[q] = \Delta X_p^{(r)}[q])$. If the value of ϵ_a is high, then the estimated values of significant key bits are accurate. The bias ϵ_a is known as backward bias. The steps to carry out the attack are as follows:

1. For each guessed key, N pairs of keystream blocks are gathered.
2. Using the N pairs, we calculate the bias of the differential output for each significant key bit.
3. The PNBs are then identified by performing an exhaustive search for all remaining keys.

There is only one correct guess for the significant key out of 2^m possible guesses. We define a null and an alternative hypothesis as follows:

H_0 : The selected guess is incorrect. H_1 : The selected variable is correct.

It is clear that $2^m - 1$ guesses satisfy H_0, but only one guess satisfies H_1. The possible errors in this testing of the hypothesis are given as:

1. Error of Non-detection: The chosen variable in this case is correct but not detected. The probability of this event is $\text{Pr}_{e_{ND}}$.
2. Error of False Alarm: In this case, a variable that causes significant bias is selected incorrectly. The occurrence has a probability of $\text{Pr}_{e_{FA}}$.

Let $\mathcal{X}_0$ be the normal distribution if the null hypothesis H_0 is true. The mean and standard deviation for the $\mathcal{X}_0$ distribution are: $\mu_0 = \frac{N}{2}$ and $\sigma_0 = \frac{\sqrt{N}}{2}$.

Similarly, $\mathcal{X}_1$ is the normal distribution if the alternative hypothesis H_1 is true. The mean and standard deviation for the distribution $\mathcal{X}_1$ are denoted by μ_1 and σ_1 respectively and are given by:

$$\mu_1 = \frac{N}{2}(1 + \epsilon_a\epsilon_d) \quad \text{and} \quad \sigma_1 = \sqrt{\sigma_{\mathcal{X}}} = \sqrt{\frac{N}{4}(1 + \epsilon_a\epsilon_d) \cdot (1 - \epsilon_a\epsilon_d)}.$$

Using the Neyman-Pearson lemma, for $\text{Pr}_{e_{FA}} = 2^{-\alpha}$ and $\text{Pr}_{e_{ND}} = 1.3 \times 10^{-3}$, required N samples to achieve a bound on these probabilities is

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

In [CPV+23], the formula of N given in Equation 3 is used to compute the data complexity value. In Subsection 3.2, we explain the concept of probabilistically independent $\mathcal{IV}$ bits and the probability p_1 and their impact on the number of key-$\mathcal{IV}$ pair satisfying the condition of becoming a right pair. Consequently, we observe that the ϵ_d in mean and standard deviation of $\mathcal{X}_1$ distribution should be replaced by $p_1 \cdot \epsilon_d$. Thus, the formula of N given in Equation 3 becomes

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1 - \epsilon_a^2 \cdot (p_1 \cdot \epsilon_d)^2}}{\epsilon_a \cdot (p_1 \cdot \epsilon_d)}\right)^2. \tag{4}$$

The time complexity of the attack is given by the equation

$$C = 2^m \times \left(N + 2^{(256-m)} Pr_{e_{FA}}\right) + 2^{256-m} = 2^m \cdot N + 2^{256-\alpha} + 2^{256-m}. \tag{5}$$

For a right pair-based attack mentioned in Subsection 2.2, the final data complexity value will become $p^{-1} \times N$. Similarly, the final time complexity formula will be:

$$p^{-1} \times C = p^{-1} \times \left(2^m \cdot N + 2^{256-\alpha} + 2^{256-m}\right). \tag{6}$$

3.4 Divide-and-Conquer Approach & Revised Complexity Calculation

Recently, [Dey24] proposed a new technique to reduce the number of operations to observe the differential-linear correlation obtained using the PNB-based approach. We use the same technique in our context, as explained.

If the output difference is observed at a linear combination of multiple bits, $OD_1, OD_2, \cdots, OD_k$, then at first, the PNB set is constructed by the usual approach for the linear combination of $\mathcal{OD}$ bits. Then, for each of the OD_i's, a separate PNB set is constructed by adding extra PNBs corresponding to OD_i only. Therefore, for each OD_i, we have a set PNB_{OD_i}. Now, during the actual attack, for each of the OD_i, we do as follows:

- From Z, Z', we compute $Z - \bar{X}, Z' - \bar{X}'$, where $\bar{X}, \bar{X}'$ are constructed by assigning random values at PNB_{OD_i}.
- Then, after applying the reverse round function, we find out the difference at OD_i bits.
- For each of N pairs of Z, Z', we perform this process and store the difference in the form of N-tuple.
- Then, after guessing the significant key, we find its projection g_{OD_i} on the significant bits corresponding to OD_i (called S_i). For example, suppose (1,2,3,4) is a vector in a 4-dimensional space XYZW, then the projection of this vector on the XYZ-plane is (1,2,3).
- For each $OD_i (1 \leq i \leq k)$, we obtain the correlation value denoted by ϵ_i.

We collect the corresponding N-tuple. We find the XOR of the N-tuple corresponding to each OD_i. Then, by finding the Hamming weight of the XOR, we determine whether the guess is correct. This technique results in modification in the formulation of N. Hence, the modification of our proposed formula of N is

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1 - (p_1 \cdot \epsilon)^2}}{(p_1 \cdot \epsilon)} \right)^2. \tag{7}$$

Here ϵ denotes the product of the correlation values ϵ_d, ϵ_a and ϵ_i's i.e., $\epsilon = \epsilon_d \times \epsilon_a \cdot \prod_{i=1}^{k} \epsilon_i$.

The modified time complexity value is given as:

$$C = \sum_{i=1}^{k} 2^{m_i} \cdot N + 2^m \cdot N \cdot \frac{k-1}{2^{10} \cdot (R-r)} + 2^{256-\alpha} + 2^{256-m}. \tag{8}$$

Here, m_i denotes the cardinality of the set S_i (set of significant bits), and k is the number of output difference positions OD_i's of the OD position. The value $(R - r)$ denotes the number of reverse rounds in the computation. To compute the final data and time complexity values, N and C will be multiplied by p^{-1}.

4 Applying Our Techniques to Cryptanalyse Salsa20

In this section, we discuss how the ideas of Subsection 3.1 and Subsection 3.2 can produce improved cryptanalysis against 128 and 256-bit versions of Salsa20. In each of the attacks we use the same $\mathcal{ID} - \mathcal{OD}$ pair $X_7^{(0)}[31] \rightarrow X_4^{(5)}[7]$. Using the 2-round criterion proposed in Subsection 3.1, we achieve the forward bias $\epsilon_d = 2^{-33.75}$. By applying linear approximation over the OD bit $X_4^{(5)}[7]$, we obtain the OD position $\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25]$ in the 6-th round with correlation value $\epsilon_l = 2^{-1}$. Hence the correlation value for the $\mathcal{ID} - \mathcal{OD}$ pair $X_7^{(0)}[31] \rightarrow (\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25])$ is $\epsilon_d * \epsilon_l^2 = 2^{-35.75}$.

To provide the key-recovery attack on 256-bit key version of Salsa20/8 and Salsa20/8.5 and 128-bit key version of Salsa20/7.5, this 6-round differential-linear distinguisher ($\mathcal{ID} - \mathcal{OD}$ pair) is used by implementing the divide and conquer approach and obtaining the data and time complexity using Equation 7 and Equation 8 respectively. For the attack against the 128-bit key version of Salsa20/7, we use the Equation 4 to compute the data complexity value.

Table 5: No. of $\mathcal{IV}$ bits of Salsa20 in Different Ranges for the 'measure of independence'.

Range	= 1	0.99 - 1.0	0.98 - 0.99	0.97 - 0.98	0.95 - 0.97	0.90 - 0.95	Below 0.90
No. of $\mathcal{IV}$ bits	68	13	9	2	2	6	28

In Table 5, we present the number of probabilistically independent $\mathcal{IV}$s with their measure of independence in different ranges, which we achieve through Algorithm 1 mentioned in Subsection 3.2. For this right pair, we find out 98 probabilistically independent $\mathcal{IV}$ bits. The table shows that there are 68 bits for which the measure of independence is 1, i.e., changing the values of any of these bits of an $\mathcal{IV}$ will form a right pair with the same key, which forms a right pair with the initial $\mathcal{IV}$. Moreover, there are 13 bits with measures higher than 0.99. This result shows that our approach generates much more $\mathcal{IV}$'s than the previously existing approach.

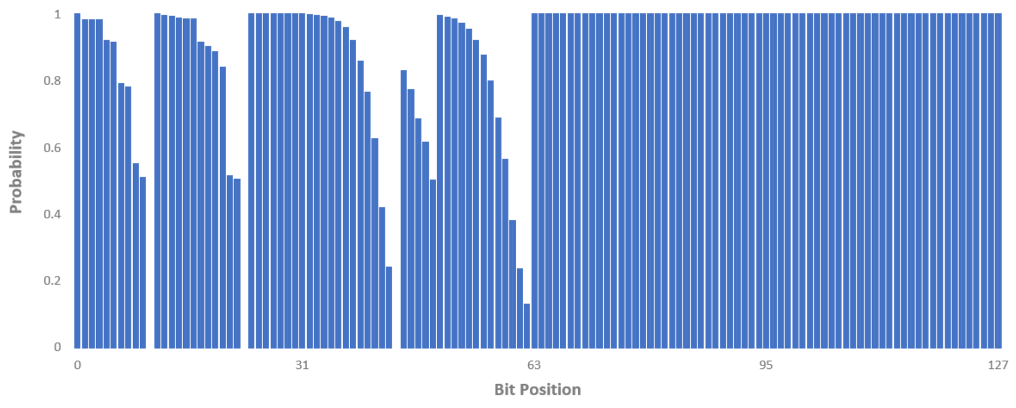


Figure 4: Graph of the Distribution of "measure of independence" for 128 $\mathcal{IV}$ Bit Positions.

In Figure 4, for each of the 128 $\mathcal{IV}$ bits, we show the measure of independence by the height of the corresponding bar. The numbers $0-31, 32-63, 64-95$, and $96-127$ represent the bits corresponding to the words X_6, X_7, X_8 and X_9 respectively.

4.1 Attack on 256-Bit Key Version of Salsa20/8

To provide an attack on Salsa20/8, we use the 6-round ($\mathcal{ID}-\mathcal{OD}$ pair $X_7^{(0)}[31] \rightarrow (\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25])$) and obtain 160 PNBs by keeping the bias limit γ to 0.3. The backward bias ϵ_a for these 160 PNBs is 0.0039. The source code for the backward bias is in the GitHub link [Sha24]. The PNB set of 160 bits is mentioned below:

4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 36, 37, 38, 39, 40, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 100, 103, 104, 105, 106, 107, 108, 109, 110, 115, 116, 117, 118, 119, 120, 121, 122, 128, 129, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 174, 175, 176, 177, 178, 179, 180, 181, 182, 186, 187, 188, 192, 193, 194, 195, 199, 200, 204, 205, 206, 207, 208, 209, 213, 218, 224, 225, 226, 231, 232, 233, 237, 238, 239, 240, 245, 249, 250, 251, 255.

To find the probabilistically independent $\mathcal{IV}$ bits, we use the idea mentioned in Subsection 3.2. Assigning a threshold of 0.91, we achieve 98 bits. Therefore, N should be less than 2^{98}. For these

98 bits, we achieve $p_1 = 0.85$. Subsection 3.1 provides a detailed explanation of the criterion for finding a right pair, which reduces the forward bias value. Further, the ideas of Subsection 3.1 and Subsection 3.2 are used similarly to improve the time complexity for the 256-bit key version of Salsa20/8.5 and 128-bit key version of Salsa20/7 and Salsa20/7.5.

We use the procedure of assigning values to PNBs as explained in [DGSS23, Section V]. We have mentioned in Subsection 3.1 that the probability of achieving a right pair is $\frac{1}{8}$, and hence the data and time complexity will be multiplied by a factor of 2^3. In recent work, [Dey24] introduced a new technique, the Divide-and-Conquer Approach, in which the PNBs are obtained individually for all OD_i positions of the OD position.

Applying the technique in the context of Salsa20, we obtained PNBs (PNB Sets for Salsa20/8) for all the 4 OD_i positions of the OD position $(\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25])$. The number of PNBs for OD bits $(\Delta X_4^{(6)}[7], \Delta X_7^{(6)}[26], \Delta X_6^{(6)}[26]$ and $\Delta X_6^{(6)}[25]$ are 40, 53, 56 and 61 respectively. Hence, the memory required for the key-recovery attack is 2^{56}. The correlation values ϵ_i's for the 4 OD_i positions are 0.893, 0.945, 0.918, and 0.833, respectively.

Therefore the correlation value for the attack on Salsa20/8 is $\epsilon = 2^{-35.75} \times 0.0039 \cdot 0.893 \cdot 0.945 \cdot 0.918 \cdot 0.833 = 2^{-44.38}$.

For $\epsilon = 2^{-44.38}$, $p_1 = 0.85$ and $\alpha = 79$, we obtain $N = 2^{96.73}$ using Equation 7, keeping $k = 4$, $R - r = 8 - 6 = 2$ and on substituting the value of N in Equation 8 we get $C = 2^{183.01}$. The final data and time complexity is $p^{-1} \times 2^{96.73} = 2^{99.73}$ and $p^{-1} \times 2^{183.01} = 2^{186.01}$, respectively, keeping $p = \frac{1}{8}$.

4.2 Attack on 256-Bit Key Version of Salsa20/8.5

By Salsa20/8.5, we mean all column operations in round number 9 considering the first two operations (half) of the quarter round function of Salsa20 as mentioned in Equation 2. We use the same set of probabilistically independent $\mathcal{IV}$s as in the attack on Salsa20/8.5, hence $p_1 = 0.85$.

For the $\mathcal{ID} - \mathcal{OD}$ pair $X_7^{(0)}[31] \to (\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25])$, we have obtained 100 PNBs. The PNB set is mentioned below. The backward bias obtained for these 100 PNBs uses the procedure of assigning values to PNBs and is given $\epsilon_a = 0.0034$. The source code for obtaining the backward bias is given in the GitHub link [Sha24].

4, 5, 6, 17, 18, 19, 20, 21, 22, 23, 24, 25, 31, 36, 37, 38, 39, 40, 41, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 93, 94, 95, 111, 115, 116, 117, 118, 128, 129, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 159, 160, 161, 162, 163, 164, 165, 166, 167, 174, 175, 176, 177, 178, 179, 192, 193, 194, 199, 204, 205, 206, 207, 218, 233, 237, 238, 239, 245, 255.

Using the Divide-and-Conquer Approach mentioned in Subsection 3.4, we also obtained PNB for all 4 OD_i positions (PNB Sets for Salsa20/8.5) of the OD position. Hence, the memory required for the key-recovery attack is 2^{102}. The correlation values ϵ_i's for the 4 OD_i positions are 0.818, 0.869, 0.811, and 0.816, respectively. We computed the value $\epsilon = 2^{-35.75} \times 0.0034 \cdot 0.818 \cdot 0.869 \cdot 0.811 \cdot 0.816 = 2^{-45.03}$ as explained in Subsection 3.4.

For $\epsilon = 2^{-45.03}$, $\alpha = 17$ and $p_1 = 0.85$, using Equation 7 we obtain $N = 2^{96.47}$. Substituting $k = 4$, $R - r = 8.5 - 6 = 2.5$ in Equation 8, the time complexity is $2^{242.84}$. The final data and time complexity is $p^{-1} \times 2^{96.47} = 2^{99.47}$ and $p^{-1} \times 2^{242.84} = 2^{245.84}$, respectively for $p = 2^{-3}$.

4.3 Attack on 128-Bit Key Version of Salsa20/7

The Divide-and-Conquer Approach mentioned in Subsection 3.4, in the context of a 256-key bit version of Salsa20, can be applicable in the case of a 128-key bit version. Here, in the case of Salsa20/7 with 128-key bit, we only see a slight improvement over the previous attack [DLS24]. Thus, we use the attack model that uses two Input-Output pairs mentioned in Algorithm 2 in [DGSS22] and conclude that corresponding to the Salsa20 cipher for each and every reduced round version, there are different ways to get the optimized value of the time complexity.

In this procedure, two $\mathcal{ID}-\mathcal{OD}$ pairs are used to improve the time complexity. The two $\mathcal{ID}-\mathcal{OD}$ pairs used in this attack are

$\mathcal{ID}_1-\mathcal{OD}_1 : X_7^{(0)}[31]-X_4^{(5)}[7]$, $\mathcal{ID}_2-\mathcal{OD}_2 : X_7^{(0)}[0]-(X_9^{(5)}[0]\oplus X_1^{(5)}[13]\oplus X_{13}^{(5)}[0])$.

1. The first $\mathcal{ID}-\mathcal{OD}$ pair for this attack procedure is $X_7^{(0)}[31]\rightarrow X_4^{(5)}[7]$ with forward bias $\epsilon_d = 2^{-33.75}$. Applying the PNB algorithm, we have obtained 117 PNBs by keeping the bias limit γ to 0.10. The PNB set is mentioned below.

   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 103, 104, 105, 106, 109, 110, 111, 112, 113, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127.

   Hence, the remaining 11 bits are significant bits.

   Significant Bits: 29, 30, 43, 44, 45, 79, 80, 102, 107, 108, 114. **Count = 11**.

   The backward bias ϵ_a for these 117 PNBs is 0.045. For $\alpha = 39$ and $p_1 = 0.85$, we obtain $N_1 = 2^{83.67}$ using Equation 4. The source code for obtaining the backward bias is given in the GitHub link [Sha24].

2. The second $\mathcal{ID}-\mathcal{OD}$ pair for this attack procedure is $X_7^{(0)}[0]\rightarrow(X_9^{(5)}[0]\oplus X_1^{(5)}[13]\oplus X_{13}^{(5)}[0])$ with forward bias $\epsilon_d = 0.116754$. Restricting the bias limit to 0.12, we obtained 56 PNBs as given below.

   0, 1, 13, 14, 15, 16, 19, 20, 27, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 44, 57, 58, 62, 63, 64, 76, 77, 78, 84, 85, 86, 87, 88, 89, 90, 91, 92, 96, 97, 101, 102, 107, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127.

   The backward bias ϵ_a for these 56 PNBs is 0.0016. The source code for obtaining the backward bias for these 56 PNBs is given in the GitHub link [Sha24].

   For $\alpha = 29$ and $p_1 = 0.85$, we obtain $N_2 = 2^{31.68}$ using Equation 4. In this case, we have 72 bits as significant bits. The set of significant bits is mentioned below.

   Significant Bits: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 17, 18, 21, 22, 23, 24, 25, 26, 28, 29, 30, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 59, 60, 61, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 79, 80, 81, 82, 83, 93, 94, 95, 98, 99, 100, 102, 103, 104, 105, 106, 108, 109, 110, 111, 112. **Count = 72**.

In both sets, we have 10 elements in common. Hence, using Algorithm 2 from [DGSS22], we have $m_1 = 11$ and $m_2 = 62$.

From [DGSS22, Section 9.2.2], the formula for data complexity is $p^{-1}\times(N_1+N_2)$, and the time complexity for this attack procedure is $p^{-1}\times(2^{m_1}\cdot N_1+2^{m_2}\cdot N_2)+2^{128-(m_1+m_2)}$, where p is the probability of getting a right pair. The value of p is 2^{-3}.

The final data complexity is $p^{-1} \times (2^{83.67} + 2^{31.68}) = 2^3 \times (2^{83.67} + 2^{31.68}) = 2^{86.67}$ and time complexity is $p^{-1} \times (2^{11} \cdot 2^{83.67} + 2^{62} \cdot 2^{31.68}) + 2^{128-(11+62)} = 2^3 \times (2^{94.67} + 2^{93.68}) + 2^{55} = 2^{98.25}$.

4.4 Attack on 128-Bit Key Version of Salsa20/7.5

The $\mathcal{ID} - \mathcal{OD}$ pair for this attack procedure is $X_7^{(0)}[31] \rightarrow (\Delta X_4^{(6)}[7] \oplus \Delta X_7^{(6)}[26] \oplus \Delta X_6^{(6)}[26] \oplus \Delta X_6^{(6)}[25])$. The forward bias is $\epsilon_d * \epsilon_l^2 = 2^{-35.75}$. Applying the PNB algorithm, we have obtained the 107 PNBs listed below.

0, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 103, 104, 105, 106, 107, 109, 110, 111, 112, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127.

The backward bias ϵ_a for these 107 PNBs is 0.014. The backward bias source code is given in the GitHub link [Sha24]. The divide-and-conquer Approach mentioned in Subsection 3.4 can also be incorporated for the 128-bit Key version. Hence, the time complexity formula for the 128-bit Key version is given by

$$\sum_{i=1}^{k} 2^{m_i} \cdot N + 2^m \cdot N \cdot \frac{k-1}{2^{10} \cdot (R-r)} + 2^{128-\alpha} + 2^{128-m}. \tag{9}$$

The number of PNB bits is obtained for all 4 OD_i positions (See PNB Set for Salsa20/7.5) of the OD position. Hence, the value of m_i corresponding to each OD_i position is considered, and the correlation value is given by $\epsilon = 2^{-35.75} \times 0.014 \cdot 0.877 \cdot 1 \cdot 1 \cdot 1 = 2^{-42.09}$.

Substituting, $\epsilon = 2^{-42.09}$, $\alpha = 31$ and $p_1 = 0.85$, we obtain $N = 2^{91.16}$ using Equation 7 and on substituting value of N, $k = 4$ and $R - r = 7.5 - 6 = 1.5$ in Equation 9 we get $C = 2^{108.47}$. The final data and time complexity is $p^{-1} \times 2^{91.16} = 2^{94.16}$ and $p^{-1} \times 2^{108.47} = 2^{111.47}$, respectively, where $p = 2^{-3}$.

5 Critically Revisiting a Few Existing Attacks in Terms of Complexities

In Asiacrypt 2022 [CPGV+22] and Journal of Cryptology [CPV+23], Coutinho et al. proposed an attack on Salsa20 in which they provided a significant improvement in complexity by approximately 2^{23} against the 256-bit key version of Salsa20/8 over the previous attack [DLS24]. The authors have introduced a new technique called Bidirectional Linear Expansions (BLE). Using the Bidirectional Linear Expansions technique, they have a single-bit differential of 5 rounds. $\mathcal{ID} - \mathcal{OD}$ pair for this attack procedure is $X_7^{(0)}[31] \rightarrow X_4^{(5)}[7]$ with forward bias $\epsilon_d \approx 2^{-42.01}$.

Primarily, there are two main issues that we address as the limitations in their contributions. The first pertains to the claimed complexity of the attack on Salsa20/8, which we find incorrect based on the data and time complexity calculation formula provided in the paper. The second issue concerns the data complexity of several attacks, which exceeds the upper limit for a valid attack according to their technique. In the following two subsections, we analyze these limitations in detail.

5.1 Analyzing the Time Complexity Calculation of [CPV$^+$23]

In the attack of [CPV$^+$23], the authors produced an attack on the 256-bit key version of Salsa20/8 with time complexity $2^{217.14}$, which was a major improvement over the previously existing attack. This attack used 152 PNBs, producing a backward bias $\epsilon_a = 0.000305$. The authors used Equation 5 in their attack (as mentioned in [CPV$^+$23, Subsection 5.2.4]) to compute the time complexity and claimed to achieve the time complexity for $\alpha = 14$. However, we can see that for $\alpha = 14$, the term $2^{256-\alpha}$ alone is 2^{242}, which makes the time complexity significantly higher than their claim. Further, in the following result, we show that for the given ϵ_d and the same set of PNBs as in [CPV$^+$23], whichever value of α is used, the time complexity can never go below 2^{219}.

Result 1. *With forward bias $\epsilon_d \approx 2^{-42.01}$, backward bias $\epsilon_a = 0.000305$ for 152 PNBs and $p = \frac{1}{2}$, for any α, the time complexity of the key-recovery attack cannot be less than 2^{219}.*

Proof. For 152 PNBs, $m = 104$. Suppose, if possible, the time complexity is less than 2^{219}. Then, from Equation 6 we have $p^{-1} \times \left(2^m N + 2^{256-\alpha}\right) < 2^{219}$. Substituting $p^{-1} = 2$, we obtain two inequalities $2^{256-\alpha} < 2^{218}$ and $2^m N < 2^{218}$. From the first inequality, we have $\alpha > 38$. From the second inequality, putting $m = 104$, we have $N < 2^{114}$, which implies $\alpha < \left(\frac{\left(2^{57} \times \epsilon_a \epsilon_d\right) - 3\sqrt{1-\epsilon_a^2 {\epsilon_d}^2}}{\sqrt{\log 4}}\right)^2 = 34.54$. Thus, we arrive at a contradiction, which implies that no such value of α exists for which the time complexity is less than 2^{219}. ■

5.2 Invalidity of the Attacks Due to High Data Complexity

The attacks proposed in [CPV$^+$23] are based on right key-$\mathcal{IV}$ pairs, which have been discussed in Subsection 2.2. In this attack procedure, an $\mathcal{IV}$ is fixed in the $\mathcal{ID}$ column, and the remaining $\mathcal{IV}$'s are varied to generate the keystream. Therefore, in this approach, for a fixed key, at most 2^{96} possible keystreams can be generated because the attacker cannot change the $\mathcal{IV}$ of the input difference column. This has been mentioned in [CPV$^+$23, Section 5.2.1] as quoted below:

> "The QRF of Salsa20 is independently applied to each column in the first round. Therefore, when the output difference of one QRF is restricted, the input of the other three QR functions is trivially independent of the output difference. It implies that we have 96 independent bits and can easily amplify the probability of the differential-linear distinguisher."

Therefore, according to their approach, for a valid/feasible attack, the condition $N \leq 2^{96}$, i.e., data complexity $p^{-1} \times N \leq p^{-1} \times 2^{96}$ must hold. Any key-recovery attack where the required data is higher than this limit is infeasible. In [CPV$^+$23], we find that all the key-recovery attacks against Salsa20 have crossed this limit of data complexity. Similarly, the authors of [CPV$^+$23] also proposed several distinguishing attacks on Salsa20/7 and Salsa20/8. By the same argument as above, the data should be less than 2^{97}, which is not followed by the proposed attacks. Our claim has been confirmed by one of the authors of [CPV$^+$23]. Table 6, lists all the key-recovery and distinguishing attacks against Salsa20.

Correction in the Bias

In the key-recovery attack on Salsa20/8 [CPV$^+$23] with forward bias $\epsilon_d \approx 2^{-42.01}$, we observe that the backward bias value ϵ_a for 152 PNBs is 0.00305 instead of 0.000305. So, we calculate the data and time complexity for the given ϵ_d and the same set of PNBs. Substituting the values in Equation 3, we observe that whatever the value of α, the value of N will never be less than or equal to 2^{96}. Hence, the data complexity value can never go below 2^{97} for the given values of ϵ_d and ϵ_a for the given set of PNBs.

Re-Evaluating the Attack Complexity of [CPV$^+$23]

By suitable modification of the PNB set, we can make the attack feasible. Given that this condition is satisfied, we find out that for the attack methodology of [CPV$^+$23], the best attack

Table 6: Attacks on Salsa20 for 256-Bit Key That are Invalid as the Data Complexity Exceeds the Maximum Limit.

Round	Reference	Attack Type	Data	Maximum Limit
7	Journal Of Cryptology [CPV+23]	Key-Recovery	$2^{104.97}$	2^{97}
	Asiacrypt [CPGV+22]	Distinguishing	2^{109}	
	Journal Of Cryptology[CPV+23]		$2^{108.96}$	
8	Asiacrypt [CPGV+22]	Key-Recovery	2^{114}	2^{97}
	Journal Of Cryptology [CPV+23]		$2^{213.14}$	
	Asiacrypt[CPGV+22]	Distinguishing	2^{216}	
	Journal Of Cryptology [CPV+23]		$2^{215.62}$	

can be obtained for 117 PNBs, achieved by the threshold $\gamma = 0.64$. For these 117 PNBs, we obtain the backward bias $\epsilon_a = 0.139 \approx 2^{-2.84}$. Using Equation 3 and Equation 5, for $\alpha = 24$, the values of N and C for a key-recovery attack on Salsa20/8 are computed to be $2^{95.97}$ and $2^{235.15}$, respectively. Hence, the final data and time complexity for a key-recovery attack on Salsa20/8 are given as $p^{-1} \times 2^{95.97} = 2^{96.97}$ and $p^{-1} \times 2^{235.15} = 2^{236.15}$, respectively. It indicates this is the least possible complexity for a feasible attack using the approach of [CPV+23]. Increasing the number of PNBs to 118, we obtain the value of backward bias $\epsilon_a = 0.121 \approx 2^{-3.04}$; however, to restrict the value of N below 2^{96}, the value of α should be less than 16, which would make the time complexity equal to 2^{240}, but it was claimed to be $2^{217.14}$ in [CPV+23].

As we mentioned in Table 4, the bias of $(\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0)$ for the 1-round condition is $0.0000060 = 2^{-17.35}$. Therefore, the overall bias $2^{-36.61}$ after applying Piling-up Lemma on $\Delta X_4^{(4)}[7] \oplus \Delta X_{12}^{(4)}[0] = 0$ and $\Delta X_0^{(4)}[0] = 0$. Using this experimental bias computed for the 1-round condition, we observe that the attack of [CPV+23] remains invalid ($N > 2^{96}$) if the rest of the parameters remain the same, as mentioned in [CPV+23, Section 4.2]. To provide a valid attack on Salsa20/8 i.e., for ($N <= 2^{96}$) with bias $\epsilon_d = 2^{-36.61}$, the obtained complexity is $2^{238.2}$, keeping $\alpha = 18.8$ and $N = 2^{95.9}$.

Also, as we mentioned, the bias ϵ_a was incorrectly measured in the attack on Salsa20/8 mentioned by [CPGV+22, CPV+23]. Fixing these mistakes, we performed the recalculation with the correct value of $\epsilon_a = 0.00305$ and obtained $N = 2^{105.77}$, time $= 2^{209.84}$. Compared to this, our improvement is of 2^{23}.

In Section 5, we have shown that the attack against Salsa20/8 mentioned in [CPV+23] is invalid due to incorrect time complexity computation and the data complexity exceeding the upper limit. Hence, in our work, we have provided a huge improvement of more than 2^{54} over [DLS24] for the attack against the 256-bit key version of Salsa20/8.

6 Conclusion

In this paper, we have introduced several interesting directions in the cryptanalysis of Salsa20. Initially, the concept of identifying a right pair was restricted to the first round alone. However, we have successfully extended this approach to the second round. This extension serves as a foundation for pursuing distinguishers across additional rounds, as the added conditions enhance the biases, paving the way for more advanced cryptanalytic efforts. Secondly, the idea of probabilistically independent bits has also been used for the first time against Salsa20, and this idea can be exploited in the future if any stronger criterion for the right pair can be explored. These ideas are not only restricted to Salsa20 but can also be used for other ciphers with similar design principles. With these techniques, we have successfully cryptanalyzed Salsa20 for more than 8 rounds for the first time. The sharpness of our technique is also underlined by the ability to cryptanalyze Salsa20/8 with significantly improved time and data complexities compared to existing methods. Further improvement in the PNB technique can help us provide an attack against Salsa20/9, though we have checked that this may not be achieved immediately with the existing techniques. Finally, we would like to point out that complexity calculations should be executed more disciplined in literature, as we obtain several calculations that are not properly executed in certain state-of-the-art publications.

Acknowledgments: The authors like to thank the anonymous reviewers for their detailed comments that improved the editorial as well as technical presentation of the paper. Subhamoy Maitra acknowledges the funding support provided by the "Information Security Education and Awareness (ISEA) Project phase - III, Cluster - Cryptography, initiatives of MeitY, Grant No. F.No. L-14017/1/2022-HRD".

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, *FSE 2008*, volume 5086 of *LNCS*, pages 470–488. Springer, Heidelberg, February 2008. doi:10.1007/978-3-540-71039-4_30.

[Ber08] Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs: The eSTREAM Finalists*, volume 4986, pages 84–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. URL: https://doi.org/10.1007/978-3-540-68351-3_8.

[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.

[CM16] Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round salsa and ChaCha. Cryptology ePrint Archive, Report 2016/1034, 2016. https://eprint.iacr.org/2016/1034.

[CPGV$^+$22] Murilo Coutinho, Iago Passos, Juan C. Grados Vásquez, Fábio L. L. de Mendonça, Rafael Timteo de Sousa, and Fábio Borges. Latin Dances Reloaded: Improved Cryptanalysis Against Salsa and ChaCha, and the Proposal of Forró. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 256–286, Cham, 2022. Springer Nature Switzerland. URL: https://doi.org/10.1007/978-3-031-22963-3_9.

[CPV$^+$23] Murilo Coutinho, Iago Passos, Juan Vásquez, Santanu Sarkar, Fábio Lucio Mendonça, Rafael Sousa, and Fábio Borges. Latin Dances Reloaded: Improved Cryptanalysis Against Salsa and ChaCha, and the Proposal of Forró. *Journal of Cryptology*, 36, 2023. URL: https://doi.org/10.1007/s00145-023-09455-5.

[Cro05] Paul Crowley. Truncated differential cryptanalysis of five rounds of Salsa20. Cryptology ePrint Archive, Paper 2005/375, 2005. URL: https://eprint.iacr.org/2005/375.

[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.

[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. Cryptology ePrint Archive, Paper 2023/134, 2023. URL: https://eprint.iacr.org/2023/134.

[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.

[DLS24] Sabyasachi Dey, Gregor Leander, and Nitin Kumar Sharma. Improved key recovery attacks on reduced-round Salsa20. *Designs, Codes and Cryptography*, November 2024. URL: https://doi.org/10.1007/s10623-024-01522-7.

[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.

[DS18] Kakumani K. C. Deepthi and Kunwar Singh. Cryptanalysis of Salsa and ChaCha: Revisited. In Jiankun Hu, Ibrahim Khalil, Zahir Tari, and Sheng Wen, editors, *Mobile Networks and Management*, pages 324–338, Cham, 2018. Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-90775-8_26.

[eST] ECRYPT. eSTREAM. the ECRYPT Stream Cipher Project. URL: https://www.ecrypt.eu.org/stream/.

[FMB+06] Simon Fischer, Willi Meier, Côme Berbain, Jean-François Biasse, and M. J. B. Robshaw. Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. In Rana Barua and Tanja Lange, editors, *Progress in Cryptology - INDOCRYPT 2006*, pages 2–16, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/11941378_2.

[LH94] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 17–25. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48658-5_3.

[Mai15] Subhamoy Maitra. Chosen IV cryptanalysis on reduced round ChaCha and Salsa. Cryptology ePrint Archive, Report 2015/698, 2015. https://eprint.iacr.org/2015/698.

[MPM15] Subhamoy Maitra, Goutam Paul, and Willi Meier. Salsa20 Cryptanalysis: New Moves and Revisiting Old Styles. Cryptology ePrint Archive, Paper 2015/217, 2015. URL: https://eprint.iacr.org/2015/217.

[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.

[Sal] SalsaApp. SalsaApplications. URL: https://ianix.com/pub/salsa20-deployment.html.

[Sha24] Nitin Kumar Sharma. Cryptanlysis of Salsa20. GitHub Repository, 2024. URL: https://github.com/SharmaNitinKumar/FSE.

[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, *ICISC 12*, volume 7839 of *LNCS*, pages 337–351. Springer, Heidelberg, November 2013. doi:10.1007/978-3-642-37682-5_24.

[TSK+07] Yukiyasu Tsunoo, Teruo Saito, Hiroyasu Kubo, Tomoyasu Suzaki, and Hiroki Nakashima. Differential cryptanalysis of Salsa20/8. *Workshop Record of SASC*, page volume 28, 2007. URL: http://www.ecrypt.eu.org/stream/papersdir/2007/010.pdf.

[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.

Appendix: PNB SETS

6.1 PNB Sets for 256-bit Key Version

6.1.1 PNB Sets for Salsa20/8

For each OD_i

$\mathcal{OD}_1$ ($X_4^{(6)}[7]$) {0, 1, 2, 3, 33, 34, 35, 46, 47, 48, 49, 78, 79, 80, 81, 96, 97, 124, 125, 126, 127, 138, 155, 156, 157, 158, 189, 190, 191, 219, 220, 221, 222, 223, 234, 235, 236, 252, 253, 254.}
Count = 40, Bias = 0.893.

$\mathcal{OD}_2$ ($X_7^{(6)}[26]$) {42, 43, 44, 45, 46, 47, 48, 49, 81, 101, 102, 112, 113, 114, 169, 170, 171, 172, 173, 182, 183, 184, 185, 189, 190, 191, 196, 197, 198, 201, 202, 203, 209, 210, 211, 212, 214, 215, 223, 226, 227, 228, 229, 230, 234, 235, 236, 241, 242, 243, 244, 246, 247.}
Count = 53, Bias = 0.945.

$\mathcal{OD}_3$ ($X_6^{(6)}[26]$) {33, 34, 35, 97, 98, 99, 101, 102, 130, 131, 132, 133, 134, 135, 136, 137, 138, 155, 156, 157, 158, 169, 170, 171, 172, 173, 182, 183, 184, 196, 197, 198, 201, 202, 203, 209, 210, 211, 212, 214, 215, 216, 226, 227, 228, 229, 241, 242, 243, 244, 246, 247, 248, 252, 253, 254.}
Count = 56, Bias = 0.918.

$\mathcal{OD}_4$ ($X_6^{(6)}[25]$) {3, 33, 34, 35, 49, 81, 96, 97, 98, 99, 101, 102, 114, 130, 131, 132, 133, 134, 135, 136, 137, 138, 155, 156, 157, 158, 169, 170, 171, 172, 173, 182, 183, 191, 196, 197, 198, 201, 202, 203, 209, 210, 211, 212, 214, 215, 223, 226, 227, 228, 236, 241, 242, 243, 244, 246, 247, 248, 252, 253, 254.}
Count = 61, Bias = 0.833.

6.1.2 PNB Sets for Salsa20/8.5

For each OD_i

$\mathcal{OD}_1$ ($X_4^{(6)}[7]$) {0,1, 2, 26, 27, 28, 29, 30, 32, 33, 34, 35, 46, 47, 48, 49, 81, 82, 83, 84, 119, 120, 121, 122, 123, 124, 125, 126, 127, 138, 155, 156, 157, 158, 186, 187, 188, 189, 190, 191, 219, 220, 221, 222, 223, 231, 232, 234, 249, 250, 251, 252, 253, 254.}
Count = 54, Bias = 0.818.

$\mathcal{OD}_2$ ($X_7^{(6)}[26]$) {42, 43, 44, 45, 46, 47, 48, 49, 91, 92, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 112, 113, 168, 169, 170, 171, 172, 173, 189, 190, 191, 195, 196, 197, 198, 200, 201, 202, 203, 208, 209, 210, 211, 212, 213, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 234, 235, 236, 240, 241, 242.}
Count = 62, Bias = 0.869.

$\mathcal{OD}_3$ ($X_6^{(6)}[26]$) {7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 26, 27, 28, 29, 30, 32, 33, 34, 35, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 119, 120, 155, 156, 157, 158, 172, 173, 195, 196, 197, 198, 200, 201, 202, 203, 208, 209, 210, 211, 212, 213, 214, 215, 240, 241, 242, 243, 244, 246, 247, 248, 249, 250, 251, 252, 253, 254.}
Count = 68, Bias = 0.811.

$\mathcal{OD}_4$ ($X_6^{(6)}[25]$) {3, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 26, 27, 28, 32, 33, 34, 35, 49, 92, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 114, 119, 138, 155, 156, 157, 158, 171, 172, 173, 191, 195, 196, 197, 198, 200, 201, 202, 203, 208, 209, 210, 211, 212, 213, 223, 236, 240, 241, 242, 243, 244, 246, 247, 248, 249, 250, 251, 252.}
Count = 70, Bias = 0.816.

6.2 PNB Set for 128-bit Key Version

PNB Set for Salsa20/7.5

For each OD_i

$\mathcal{OD}_1$ ($X_4^{(6)}[7]$) {28, 29, 30, 80, 108.} **Count = 5, Bias = 0.877**.

$\mathcal{OD}_2$ ($X_7^{(6)}[26]$) {1, 2, 3, 4, 42, 43, 44, 45, 100, 101, 102, 108, 113, 114, 120.}

Count = 15, Bias = 1.

$\mathcal{OD}_3$ ($X_6^{(6)}[26]$) {4, 8, 9, 45, 80, 120.} **Count = 6, Bias = 1**.

$\mathcal{OD}_4$ ($X_6^{(6)}[25]$) {3, 4, 8, 9, 44, 45, 80, 114, 120.} **Count = 9, Bias = 1**.