IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 January 2024, accepted 16 February 2024, date of publication 27 February 2024, date of current version 4 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3367797

RESEARCH ARTICLE

Enhancing the Key Recovery Attack on Round Reduced Salsa

CHANDAN DEY[1], SABYASACHI DEY[2], RAHUL GIRME[1], AND SANTANU SARKAR[1]
[1]Department of Mathematics, Indian Institute of Technology Madras, Chennai 600036, India
[2]Department of Mathematics, Birla Institute of Technology and Science, Hyderabad Campus, Hyderabad 500078, India

Corresponding author: Santanu Sarkar (santanu@iitm.ac.in)

ABSTRACT Salsa is the most well-known stream cipher and a finalist of the eSTREAM project. The concept of probabilistic neutral bits (PNBs) first presented by Aumasson et al., is the most important step in the cryptanalysis of Salsa. In this paper, we provide a strategy to find a better set of PNBs and we improve the existing attacks. Our attack complexity is $2^{210.38}$, which is an improvement of the latest work at ASIACRYPT 2022. We also revisit the work of Ghafoori et al. (ISPEC 2022). In their study, they used a PNB-based differential attack to present a key recovery attack on Salsa20/8 with a time complexity of $2^{144.75}$. They claimed their approach was the most effective single-bit differential attack to date. Our paper challenges this claim, providing experimental results and reasoned arguments to support our case.

INDEX TERMS Differential cryptanalysis, PNBs, stream cipher, Salsa.

I. INTRODUCTION

Symmetric key ciphers serve as crucial tools for safeguarding security and privacy. Within this realm, two prominent categories emerge based on the nonlinear operations they employ: S-box-based ciphers, which rely on substitution boxes, and ARX ciphers, constructed solely through modular additions, bit rotations, and bitwise XOR operations. The only way to increase trust in symmetric key cipher once it is constructed is to keep trying to assess its security. Many attacks exist to evaluate the security of symmetric key ciphers. The most two important ones are differential cryptanalysis [5] and linear cryptanalysis [19]. Generally, the attacks fall into two categories; the distinguisher and the key recovery part. In this paper, we focus on key recovery of the cipher using differential cryptanalysis.

Daniel J. Bernstein proposed the design of Salsa stream cipher in April 2005 [2], which provides a security of 256-bit against key-recovery attacks. Salsa stream cipher has the option of supporting keys with a 128-bit security level. Salsa is an ARX-based stream cipher design. Salsa stream cipher, which has 20 rounds, was submitted by the designer to the eSTREAM project [21] under the ECRYPT Stream Cipher Project. Salsa20/12, a Salsa stream cipher variant with 12 rounds, was one of the selected candidates for the eSTREAM software portfolio when it was finalized in September 2008. Later, Bernstein made some modifications in the design of Salsa so that after each round gets a good diffusion and these changes create a stream cipher called ChaCha [3]. Since Salsa is a famous stream cipher, in order to update the security margin, security analysis is essential. In this work, we discuss a key recovery attack on reduced round Salsa.

Related Work: Crowley introduced the cryptanalysis on Salsa in 2005 [9]. For the cryptanalysis of these type of ciphers, Aumasson et al. developed the idea of Probabilistic Neutral Bits (PNBs) at FSE 2008 [1]. After that, a lot of authors offered small amendments to Aumasson et al.'s attack. As an instance, in [22] the idea of Column Chaining Distinguisher (CCD) was proposed to achieve some incremental improvement over Salsa and ChaCha. In [17] Maitra et al. attacked 8 rounds of Salsa with a key search complexity of $2^{247.2}$. To achieve specific improvements over the prior findings, Maitra [16] put forward IV selection. Later, Dey and Sarkar [10] improved the attack by choosing appropriate values for the PNBs. Choudhari and Maitra [6] significantly improved the attack against Salsa and ChaCha by using multibit differential distinguishers. Subsequently,

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

Coutinho et al. [7], [8], presented the first differential-linear distinguishers that can reach rounds 7 and 8 of the Salsa cipher. Additionally, they enhanced the efficacy of PNB-based key-recovery attacks against 7 and 8 rounds of Salsa. In [12] Ding provided the related-cipher attack with time complexity $2^{193.58}$ using two separate IVs in Salsa20/12 and Salsa20/8 with a secret key. Also, in [13] Ghafoori et al. proposed a PNB-based differential attack on 8 rounds of Salsa with time complexity $2^{144.75}$.

Our Contribution: In this work, we improve the key recovery attack on 8-round Salsa than ASIACRYPT 2022 paper [7]. For this, we apply a new strategy to find the PNBs set. Using this strategy, we increase the PNBs set and enhance the attack complexity for 8-round Salsa. Also, we show that the forward 5-round differential biases presented in the paper [13] are incorrect. As a result, their proposed attack on 8-round Salsa at ISPEC 2022 [13] becomes invalid. In the following Table 1, we provide our findings with existing results.

Paper Outline: This paper is structured as follows: in section II we define our notation and illustrate the structure of Salsa stream cipher. In section III, we discuss about basic attack framework using the concept of probabilistic neutral bits (PNBs). We review the work of ASIACRYPT 2022 [7] and provide a strategy to find a better set of PNBs that improve the attack complexity over the previous findings in section IV. In the following section V, we review the work of ISPEC 2022 [13], we show some flaws in their work and their key recovery attack for 8 rounds is incorrect. After that, we conclude the paper in the section VI.

II. NOTATIONS AND STRUCTURE OF SALSA

In the following Table 2, we list the notations used in the paper.

A. SALSA

Salsa operates on 64 bytes, comprising 16 words of 32 bits each. The state is represented as a 4×4 matrix, where each element is a 32-bit word. The state of Salsa initialized with 256-bit key $k_0, k_1, \ldots, k_7$ where each k_i is a 32-bit integer, two 32-bit nonces v_0, v_1, two 32-bit counters t_0, t_1(we may also use the terms IV to refer to the nonce and counter words), and four constant $c_0 =$ `0x61707865`, $c_1 =$ `0x3320646e`, $c_2 =$ `0x79622d32` and $c_3 =$ `0x6b206574` for Salsa, the following state of the matrix is:

$$S = \begin{pmatrix} S[0] & S[1] & S[2] & S[3] \\ S[4] & S[5] & S[6] & S[7] \\ S[8] & S[9] & S[10] & S[11] \\ S[12] & S[13] & S[14] & S[15] \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

The state matrix is updated in each round by a Quarter-Round Function (`QRF`). This `QRF` is a nonlinear function that operates on 4-tuple $(S^{m-1}[a], S^{m-1}[b], S^{m-1}[c], S^{m-1}[d])$ to give an output of 4-tuple $(S^m[a], S^m[b], S^m[c], S^m[d])$ where each of $S^m[a], S^m[b], S^m[c], S^m[d]$ is a 32-bit word. The `QRF` is as follows:

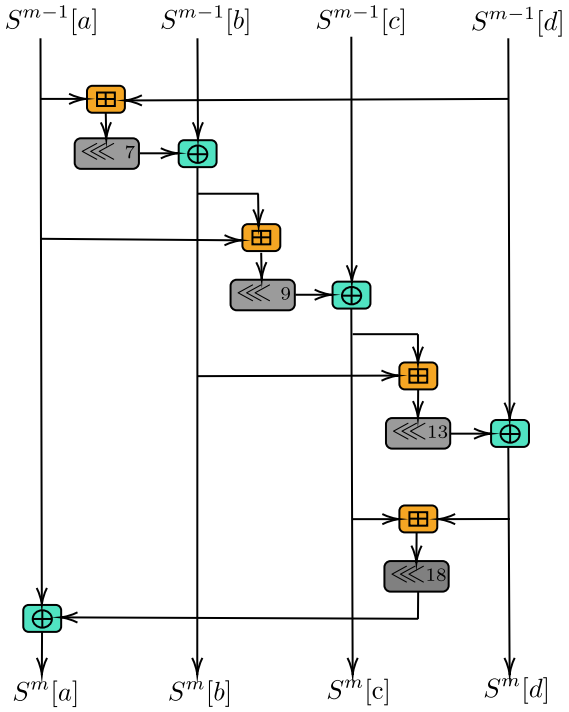

FIGURE 1. Quarter round function of salsa.

$$S^m[b] = S^{m-1}[b] \oplus ((S^{m-1}[d] \boxplus S^{m-1}[a]) \lll 7),$$
$$S^m[c] = S^{m-1}[c] \oplus ((S^{m-1}[a] \boxplus S^m[b] \lll 9),$$
$$S^m[d] = S^{m-1}[d] \oplus ((S^m[c] \boxplus S^m[b]) \lll 13),$$
$$S^m[a] = S^{m-1}[a] \oplus ((S^m[d] \boxplus S^m[c]) \lll 18).$$

Here $\boxplus$ denotes addition modulo 2^{32}, $\boxminus$ denotes subtraction modulo 2^{32}, $\oplus$ is the usual XOR operation, and $\lll$ is left cyclic rotation. Figure 1 represents the `QRF` of Salsa. The state matrix of Salsa is updated by using odd round and even round alternatively. Odd rounds and even rounds are different. In an odd number of rounds that the original Salsa specification referred to as column-rounds, applying the nonlinear operation `QRF` to columns $(S[0], S[4], S[8], S[12])$, $(S[5], S[9], S[13], S[1])$, $(S[10], S[14], S[2], S[6])$, $(S[15], S[3], S[7], S[11])$. In an even number of rounds that are referred to as the row rounds, applying the nonlinear operation `QRF` to rows $(S[0], S[1], S[2], S[3])$, $(S[5], S[6], S[7], S[4])$, $(S[10], S[11], S[8], S[9])$, $(S[15], S[12], S[13], S[14])$. S^R is the updated state after the R round applied to the initial state S^0. A keystream of 512 bits is obtained as $Z = S \boxplus S^R$. Here, one should note that each round is reversible; we can compute S^{i-1} from S^i. For more details on the same, the reader may have a look on [4].

III. ATTACK FRAMEWORK USING PROBABILISTIC NEUTRAL BITS

Most of the important attacks in the literature on security analysis of Salsa and ChaCha are based on the differential

TABLE 1. Comparison of attack complexities.

Cipher	Rounds	Attack Type	Time	Data	Works
Salsa256	8	Key recovery	2^{256}	0	Brute-force attack
			2^{251}	2^{31}	[1]
			2^{250}	2^{27}	[22]
			$2^{245.5}$	$2^{22.46}$	[16]
			$2^{244.9}$	2^{96}	[6]
			$2^{241.62}$	$2^{31.5}$	[14]
			$2^{217.14}$	$2^{113.14}$	[8]
			$2^{210.32}$	$2^{117.29}$	Our work
Cipher	Rounds	Attack Type	Time	Data	Our Observation
Salsa256	8	Key recovery	$2^{144.75}$	$2^{55.74}$	[13] Incorrect

TABLE 2. Notations.

Notation	Description
S	State matrix of 4×4 with 16 words of 32 bit each
S^0	Initial state matrix of Salsa
S^R	State matrix of Salsa after R round
$S^R[i]$	i^{th} word of state matrix of S^R
$S^R[i][j]$	j^{th} bit of i^{th} word of state matrix S^R
$S[i] \boxplus S[j]$	Addition of word $S[i]$ and $S[j]$ modulo 2^{32}
$S[i] \boxminus S[j]$	Subtraction of word $S[i]$ and $S[j]$ modulo 2^{32}
$S[i] \oplus S[j]$	Bit wise XOR operation of word $S[i]$ and $S[j]$
$S[i] \lll n$	Left rotation of word $S[i]$ by n bits
$\Delta S[i]$	XOR of word $S[i]$ and $S^{'}[i]$ defined as $\Delta S[i] = S[i] \oplus S^{'}[i]$
$\Delta S[i][j]$	XOR of bit $S[i][j]$ and $S^{'}[i][j]$ defined as $\Delta S[i][j] = S[i][j] \oplus S^{'}[i][j]$
ϵ_f, ϵ_b	Forward and backward bias respectively
ID, OD	Input and Output difference respectively

attack with probabilistic neutral bits (PNBs) technique that concept was introduced by Aumasson et al. [1]. Here we discuss the probabilistic neutral bits (PNBs) and the attack model using the probabilistic neutral bits (PNBs) technique. We consider the secret key size 256 bits for this attack discussion. In the following, we start the discussion with differential biases and then the PNBs.

Forward Differential Biases: Initially, we initialize the two-state matrices S^0 and $S^{'0}$, which consist the same keywords $(k_0, k_1, \ldots, k_7)$ and constants (c_0, c_1, c_2, c_3). The $S^{'0}$ matrix, however, is made up of a single bit change in the nonce v or counter t. For a given difference $\Delta S^0[i][j] = 1$ to the initial matrix S^0 which is called input difference, we obtain the corresponding initial state matrix $S^{'0}$ as $v' = v \oplus \Delta v$ or $t' = t \oplus \Delta t$, where v and t indicate the difference of one bit at the counter or nonce. Next, we use the starting state matrices S^0 and $S^{'0}$ as inputs to perform the Salsa round function and obtain the difference in a single bit of output. $\Delta S^r[p][q] = S^r[p][q] \oplus S^{'r}[p][q]$ is the output difference from the r-round internal state matrices S^r and $S^{'r}$. For all possible choices of nonces and counters and fixed key, the single-bit forward differential probability is defined as

$$\mathbf{Pr}(\Delta S^r[p][q] = 1 \mid \Delta S^0[i][j] = 1) = \frac{1}{2}(1 + \epsilon_f),$$

where ϵ_f stands for the forward differential bias. If the key bits are random, we calculate the value of ϵ_f^* as the median of ϵ_f [1]. Here throughout our work, we calculate the differential bias by taking random key bits and we denote this as ϵ_f for avoiding the notation's confusion.

Probabilistic Neutral Bits: Aumasson first introduced the PNBs idea in 2008. This idea is useful to reduce the complexity of searching 256 bits of the secret key. Among the 256 bits key, there are some key bits, say n bits, that have negligible influence on the differential that we call invaluable key bits and other $256 - n$ bits have a significant influence on the differential that we call valuable key bits. Using these two sets of key bits one can recover the secret key.

To differentiate these two sets Aumasson at el. [1] calculates the neutrality measure of each key bit and the neutrality measure is defined as follows:

Definition 1 (Neutrality Measure): The neutrality measure of the key bit position k concerning the output difference is defined as ϕ_k, where $\frac{1}{2}(1+\phi_k)$ is the probability that complementing the key bit k does not change the output difference.

By performing the following procedure, we compute the neutrality measure and divide the secret key into two parts invaluable key bits and valuable key bits:

1. Compute the keystream blocks $Z = S^0 \boxplus S^R$ and $Z' = S'^0 \boxplus S'^R$, where (S^R, S'^R) be a R-round internal state matrix pair corresponding to the input pair (S^0, S'^0) with $\Delta S^0[i][j] = 1$.
2. Change the key bit position k of the initial input pair (S^0, S'^0) to obtain the new pair $(\bar{S}^0, \bar{S}'^0)$.
3. Apply the inverse round function of Salsa on $Z \boxminus \bar{S}^0$ and $Z' \boxminus \bar{S}'^0$ for $R-r$ rounds to obtain (Y, Y').
4. Find $\Phi[p][q]$, where $\Phi[p][q] = Y[p][q] \oplus Y'[p][q]$ for the fixed output difference bit and $Y[p][q]$ and $Y'[p][q]$ denote the q^{th} bit of p^{th} word of Y and Y' respectively.
5. Compute the neutrality measure of the key bit as
$$\mathsf{Pr}(\Delta S^r[p][q] = \Phi[p][q] \mid \Delta S^0[i][j] = 1) = \frac{1}{2}(1+\phi_k),$$
for different initial state matrices with the same input difference.
6. Set a threshold ϕ and place all key bits with $\phi_k \geq \phi$ into the set of n-bit invaluable key bits and those with $\phi_k < \phi$ into the set of m-bit valuable key bits.

The main idea behind key recovery is to search the two sets (PNBs and the non-PNBs[1]) separately. Suppose the size of the subset containing PNBs is n, which implies that the number of non-PNB bits is $m = 256 - n$.

Attack after finding PNBs and complexity computation: Our goal is to determine the values of the non-PNBs without the knowledge of the accurate values of the PNBs. It is to be noted that changing the PNBs affects the output with a low probability. The attacker runs both states in the backward direction while guessing the valuable key bits with random values for the PNBs. Using the PNBs described in [1], the actual attack method is outlined as follows:

1. For a random fixed key, gather N pairs of key stream blocks (Z, Z'),
   where each of which is produced by states using a random nonce and counter (with appropriate ID).
2. For every choice of the m valuable key bits, perform the following procedure:
   (a) Compute the bias ϵ_f using the N pairs of initial states.
   (b) Set the random values for the invaluable key bits and obtain the states $\hat{S}$ and $\hat{S}'$.
   (c) Reverse the states $Z \boxminus \hat{S}$ and $Z' \boxminus \hat{S}'$ for $R-r$ rounds and obtain the states $\hat{Y}$ and $\hat{Y}'$ respectively.
   (d) Compute $\hat{\Phi}[p][q] = \hat{Y}[p][q] \oplus \hat{Y}'[p][q]$ and the probability using the N pair of initial states as
   $$\mathsf{Pr}(\hat{\Phi}[p][q] = \Delta S^r[p][q] | \Delta S^0[i][j] = 1) = \frac{1}{2}(1+\epsilon_b).$$
   (e) Also, we compute the
   $$\mathsf{Pr}(\hat{\Phi}[p][q] = 1 \mid \Delta S^0[i][j] = 1) = \frac{1}{2}(1+\epsilon)$$
   where $\epsilon = \epsilon_f \cdot \epsilon_b$ provided the two events are independent. If $\mathsf{Pr}(\hat{\Phi}[p][q] = 1 \mid \Delta S^0[i][j] = 1)$ offer a significant bias ϵ, we conclude that our guess of valuable key bits is correct. Once the valuable key bits have been identified, we may recover the invaluable key bits by exhaustive search.

When the PNBs are given random values in this attack, the bias in the backward direction is known as the backward bias and is denoted by ϵ_b. For m valuable key bits, there are 2^m possible sequences among which only one sequence is correct. Considering the null hypothesis H_0 that the selected sequence is incorrect, the $2^m - 1$ sequences satisfy the null hypothesis, and only one satisfies the alternative hypothesis H_1 that the selected sequence is correct. In this attack scenario, two possible errors can occur:

- **Error of non-detection**: The selected sequence is correct, but cannot be detected. The probability of this error occurring is P_{nd}.
- **False alarm error**: The selected sequence is incorrect albeit offers a significant bias. The probability of this event is P_{fa}.

Aumasson et al. in [1] used a result given by Neyman-Pearson decision theory for achieving the bound on these probabilities. Concerning this result, the number of samples is

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-(\epsilon_f \epsilon_b)^2}}{\epsilon_f \epsilon_b} \right)^2,$$

and time complexity of the attack is given by

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}.$$

Here, the probability of non-detection $P_{nd} = 1.3 \times 10^{-3}$ and probability of false alarm is $P_{fa} = 2^{-\alpha}$.

IV. REVIEWING THE WORK OF ASIACRYPT 2022

At ASIACRYPT 2022 Coutinho et al. [7], provided the key recovery attack on 8 rounds of Salsa with time complexity $2^{217.14}$ and data complexity $2^{213.14}$. They provide a new technique called Bidirectional Linear Expansions (BLE) and find the single-bit differential correlation for the bit $S^5[4][7]$, here we summarise their work briefly.

[1]PNB means invaluable key bit and non-PNB means valuable key bit. These are interchangeably used.

A. BIDIRECTIONAL LINEAR EXPANSIONS

In [7] and [8], Coutinho et al. introduced the BLE approach to enhance the attacks against Salsa. Using this technique, they investigated the expansion of a single bit in both forward and backward directions and improved the key recovery attack on both 7 and 8-round Salsa. The backward linear approximation is used to search for correlations in the previous round using the techniques of BLE. For instance, if we can compute all possible single-bit differentials for m rounds of Salsa, we have a backward linear approximation.

$$S^{m+1}[i][j] = S^m[i_1][j_1] \oplus S^m[i_2][j_2] \oplus \cdots S^m[i_k][j_k]$$

Then, we can aggregate the bias for each single bit from the previous round to achieve a differential bias for the next rounds. Mathematically, it can be expressed as: if $\mathsf{Pr}(\Delta S^m[i_p][j_p] \mid \mathrm{ID}) = \frac{1}{2}(1+\epsilon_p)$ and $\mathsf{Pr}(\Delta S^{m+1}[i][j] \mid \mathrm{ID}) = \frac{1}{2}(1+\epsilon_f)$ then using the Piling-Up Lemma [19], the targeted bias is $\epsilon_f = \prod_{p=1}^{k} \epsilon_p$.

Using this strategy, one can compute the differential bias for 5 rounds of Salsa. This is discussed in the following.

B. SINGLE BIT DIFFERENTIAL BIAS FOR 5 ROUNDS OF SALSA

In [7] and [8], Coutinho et al., provided the technique to find the single-bit differential correlation for the bit $S^5[b][7]$ using BLE. Using Linear Approximations for Salsa, the bit $S^5[b][7]$ in the fifth round can be written as

$$S^5[b][7] = S^4[a][0] \oplus S^4[b][7] \oplus S^4[d][0]$$

with probability 1, where $(a, b, d) \in \{(0, 4, 12), (5, 13, 1), (10, 2, 6), (15, 7, 11)\}$. Using this relation, first, they find the bias of three bits in the fourth round, and then combining these biases they get the bias for a bit $S^5[b][7]$ in the fifth round.

They start with input difference $\Delta S^0[7][31] = 1$ i.e.,

$$\Delta S^0 = \begin{pmatrix} 0\,0\,0 & 0 \\ 0\,0\,0 & \texttt{0x80000000} \\ 0\,0\,0 & 0 \\ 0\,0\,0 & 0 \end{pmatrix}.$$

By propagating the differential using the algorithm from [15] starting with the input difference ΔS^0 and picking the one that minimized the hamming weight after one round and get the differential state matrix as follows:

$$\Delta S^1 = \Psi = \begin{pmatrix} 0\,0\,0 & \texttt{0x00000000} \\ 0\,0\,0 & \texttt{0x80000000} \\ 0\,0\,0 & \texttt{0x00001000} \\ 0\,0\,0 & \texttt{0x40020000} \end{pmatrix}$$

Here the given input difference ΔS^0 generates output difference at four places after one round with probability $\frac{1}{2}$. Figure 2 illustrates the differential part of their proposed attack.

To estimate the transition probability from ΔS^1 to $\Delta S^5[b][7]$, first they find the differential bias for the following bits $S^4[0][0]$, $S^4[4][7]$ and $S^4[12][0]$ in the fourth round using 2^{45} random samples, one can experimentally also check the following bias. In the following Table 3, we provide the biases.

TABLE 3. Computational result.

ID	OD	Differential Bias
$\Delta S^1 = \Psi$	$\Delta S^4[0][0]$	-0.00000159
$\Delta S^1 = \Psi$	$\Delta S^4[4][7]$	-0.00085
$\Delta S^1 = \Psi$	$\Delta S^4[12][0]$	0.000167

The differential bias from round 1 to round 5 of Salsa is given by

$$\mathsf{Pr}(\Delta S^5[4][7] = 0 | \Delta S^1 = \Psi) = \frac{1}{2}(1+\epsilon_f)$$

where $\epsilon_f = 2^{-42.01}$.

Applying this 5-round forward differential and PNBs idea they mount an 8-round key recovery attack on Salsa. In the following, we discuss their attack complexity.

Attack complexity: In their attack, they use a 5-round forward differential and 3-round in the backward direction to find the PNBs set. They obtained a 152 size PNBs set which is as follows:

{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, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 174, 175, 176, 177, 178, 179, 180, 181, 186, 187, 192, 193, 194, 195, 199, 200, 204, 205, 206, 207, 208, 213, 218, 224, 225, 231, 232, 237, 238, 239, 240, 245, 249, 250, 255}.

They obtained backward bias $\epsilon_b = 0.000305$ for $\phi = 0.3$. Here the attack has to repeat two times on average since the transition probability from ΔS^0 to $\Delta S^1 = \Psi$ is $\frac{1}{2}$. Thus the attack has data complexity $2^{113.14}$ and time complexity $2^{217.14}$ for $\alpha = 14$.

C. STRATEGY TO FIND A BETTER SET OF PNBs

Here we provide the strategy to find the PNBs:

1. We assign a threshold bias ϕ_1 to find the PNBs and select those key bits as PNBs that give a higher neutrality measure than the threshold. This procedure is very similar to the traditional method. But it's not the complete set of PNBs. Suppose for this ϕ_1 we have n_1 number of PNBs set $\mathcal{A}$.
2. After that we choose a second threshold ϕ_2 that is less than the threshold ϕ_1. In this step, since ϕ_2 is very small, we may get k number of sets say $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ such that each set contains n_2 $(> n_1)$ number of candidates for PNBs. Here each set $\mathcal{A}_i$ contains n_1 number of candidates that are found in the first step. It may happen

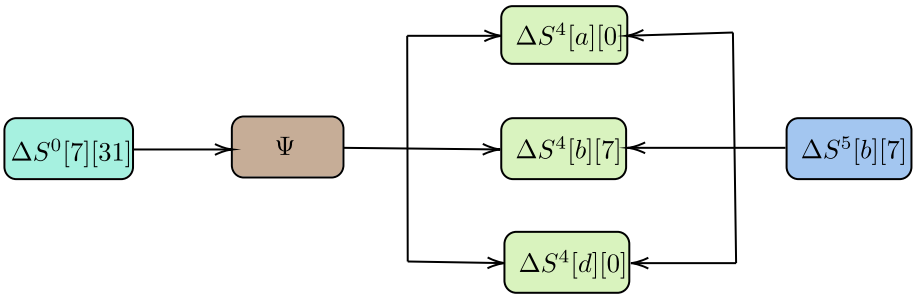

FIGURE 2. Bidirectional linear expansion of ID and OD.

that from each set remaining $n_2 - n_1$ candidates are different, so we have $k \cdot (n_2 - n_1)$ many candidates.

3. Now we arrange the $k \cdot (n_2 - n_1)$ candidates in decreasing order with respect to the neutrality measures.
4. In this step, we choose one by one candidate from $k \cdot (n_2 - n_1)$ many candidates according to the order. Then we check the backward bias, if it gives better complexity than the existing one then this key bit we considered as PNB. By filtering in this way we can construct the set $\mathcal{B}$. Therefore, we find the complete set of PNBs as $\mathcal{P} = \mathcal{B} \cup \mathcal{A}$.

For a better understanding, one can take a look at Figure 3.

Discussion: In [11], Dey et al. provided a three-step strategy to find a better set of PNBs. Here we discuss the comparison between our strategy and theirs. In their strategy, they first shortlist the possible number of PNBs by assigning the threshold as ϕ_{prelim}. For this threshold, suppose the number of PNBs is n_{prelim} and $\mathcal{A}$ denotes the set of PNBs. After that, they set the threshold to ϕ_{direct}, which is higher than ϕ_{prelim}. For this threshold, supposse the number of PNBs is n_{direct} and $\mathcal{B}$ denotes the set of PNBs. So, clearly, $\mathcal{B} \subset \mathcal{A}$. Now, they add more PNBs from $\mathcal{A} - \mathcal{B}$ by following their strategy.

In our case, we do not shortlist the set of PNBs at the beginning. We first choose the threshold as ϕ_1 and select the key bit with a neutrality measure higher than ϕ_1. Then, consider those key bits as PNBs, and the set of PNBs is denoted by $\mathcal{A}$. After that, we add a few more PNBs by assigning the second threshold ϕ_2. However, the second threshold is very small. Consequently, we may obtain k sets $\mathcal{A}_i's$, where $\mathcal{A} \subset \mathcal{A}_i$. Thus, we add the remaining candidates by following our strategy outlined in IV-C.

Our Computational Result and Attack Complexity: In our attack evaluation we consider the same ID $= \Delta S^0[7][31]$ and OD $= \Delta S^5[4][7]$ as provided in [7]. Here we set the first threshold $\phi_1 = 0.36$ and we found 129 PNBs as follows:

$\mathcal{A} = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 36, 37, 38, 39, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 103, 104, 105, 106, 107, 108, 115, 116, 117, 118, 119, 120, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 159, 160, 161, 162, 163, 164, 165, 166, 167, 174, 175, 176, 177, 178, 179, 180, 192, 193, 194, 199, 204, 205, 206, 207, 224, 225, 237, 238, 239, 241, 249, 255\}$.

After that according to the strategy we set another threshold $\phi_2 = 0.24$. Then we calculate the five-set $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5$ (k=5), in each of the five sets we have 129 PNBs common, and in each set, we have 31 candidates extra. Now we make a new set $\mathcal{U}$ by collecting candidates from each $\mathcal{A}_i$[2] excluding 129 common PNBs. Here the set $\mathcal{U}$ contains 88 distinct candidates.

Then we choose one by one candidate from $\mathcal{U}$ and check the backward bias. If the bias is better then we add this candidate to the PNB set. Therefore by this filtering, we found 34 candidates that give better backward bias. So the cardinality of the final set of PNBs becomes 163.

Here we aim to improve the attack on 8-round Salsa. We apply a 5-round forward differential for the attack. So, we must travel back 3 rounds in the backward direction to mount the attack. We obtain backward bias $\epsilon_b = 0.00016$ in this instance. We have the following list of 163 PNB bits

$\mathcal{P} = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 36, 37, 38, 39, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 103, 104, 105, 106, 107, 108, 115, 116, 117, 118, 119, 120, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 159, 160, 161, 162, 163, 164, 165, 166, 167, 174, 175, 176, 177, 178, 179, 180, 192, 193, 194, 199, 204, 205, 206, 207, 224, 225, 237, 238, 239, 241, 249, 255, 128, 129, 153, 154, 40, 168, 47, 181, 182, 186, 187, 188, 195, 200, 201, 75, 76, 208, 209, 213, 214, 218, 97, 226, 100, 231, 232, 250, 109, 110, 240, 245, 121, 122\}$.

One can verify the backward bias using our source code.[3] So, we get data complexity of $2^{116.29}$ and time complexity $2^{209.32}$ for $\alpha = 52.33$, and we have to repeat this attack two times on average because transition probability from ΔS^0 to $\Delta S^1 = \Psi$ is $\frac{1}{2}$. So, the final key recovery attack for 8-round Salsa has data complexity $2^{117.29}$ and time complexity $2^{210.32}$.

[2]In Appendix we have provided the sets $\mathcal{A}_i$ for $i = 1, 2, 3, 4, 5$.
[3]https://github.com/Rahul150192/Salsa/tree/main

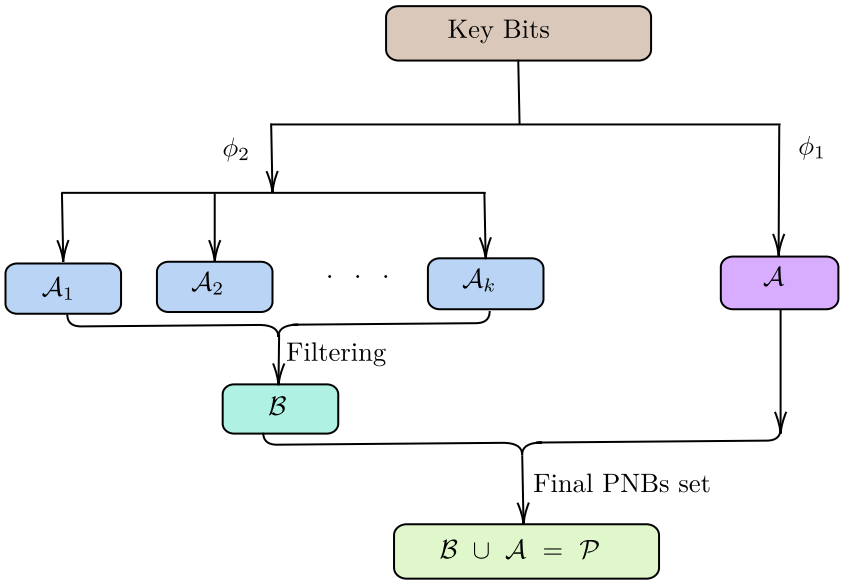

FIGURE 3. Diagram of PNBs finding strategy.

Remark: In our attack, we restrict the Hamming weight of the differential state matrix to 4 after one round in search of better 5 rounds forward distinguisher. Also, to find the right pairs for satisfying the restriction in the first round one have to repeat the procedure twice on an average. In this attack scenario, we get the data complexity 2^{117} for 8 rounds key recovery attack. In order to execute the attack, we need to generate this amount of data without hampering the restriction of hamming weight mentioned above. In the first round, the other columns have no influence on the input difference columns, we have 96 free IV bits available, which have no influence in the input difference column.

Apart from that, experimentally, we have verified that there are additional 23 bits of $S[7]$ (v_1) can be regarded as almost neutral. Specifically, given that the Hamming weight of difference after one round is 4, altering those bits of (v_1) will still produce 4 differences. The positions of these bits are {0, 1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26}. Using one billion random keys, we observed that this neutrality holds with a probability of 0.92. Consequently, we essentially have a total of $96 + 23 = 119$ free bits, implying that we can construct 2^{119} different IVs which do not hamper the restriction of minimum hamming weight with probability 0.92. This makes our data complexity of 2^{117} reasonable.

V. REVIEWING THE WORK OF ISPEC 2022

The existing methodology for differential cryptanalysis of Salsa stream cipher involves examining the ID − OD pair with the best differential bias from all potential pairs suitable for the attack. Initially, the ID is determined, and subsequently, the corresponding OD with the best forward differential bias is selected. In essence, previous research has concentrated on evaluating the differential bias at specific ID − OD pairs and then attempted to identify the subset of PNBs suitable for attacking a particular round of Salsa.

At ISPEC 2022 [13], Ghafoori et al. significantly improved the key recovery attack using the PNB-based differential attack. For their work, they followed the attack idea on ChaCha proposed at ACISP 2022 [20]. In the attack, they first find the output difference (OD) position where the PNB set is larger with better backward bias. They concentrate on thoroughly analyzing the neutrality measures of the 256 key bits concerning all potential OD bits. They used Algorithm 1 [13] to calculate the neutrality measures for 256 key bits concerning 512 OD bits. In an attempt to determine the best ID position, all 128 available ID positions were systematically explored. Surprisingly, no substantial influence of the ID on the neutrality measure of the key bits was discernible. Consequently, the decision was made to opt for a random ID selection in this phase of the experiment.

We review their work and show that there are some flaws in their work.

Claim of [13] on forward differential bias: Analysing each output difference position, they obtained the OD position $\Delta S^5[0][18]$ with best neutrality measure. Then they used a 2^{25} IDs sample for each of the 2^{10} key trials to find the ID position with the best forward differential bias (ϵ_f). They obtained ID positions $\Delta S^0[6][31]$, $\Delta S^0[6][15]$, and $\Delta S^0[7][9]$ with forward differential biases of 0.000829, 0.000793, and 0.000767, respectively, for the OD position $\Delta S^5[0][18]$. Then, they selected $\Delta S^0[6][31]$ as the ID position for the attack, as the obtained differential bias is higher than the others.

Our observation: To validate a bias experimentally, the number of random samples plays an important role. From the paper [18], we know the following theorem.

TABLE 4. Comparison of the claimed bias in [20] with our observation.

ID	Claim in [20]	Our observation	Sample
$\Delta S^0[6][31]$	0.000829	0.000001577	2^{40}
$\Delta S^0[6][15]$	0.000793	0.00000002045	2^{40}
$\Delta S^0[7][9]$	0.000767	-0.0000008128	2^{40}

Theorem 1 ([18]): Let X, Y be distributions, and suppose that the event e happens in X with probability p and in Y with probability $p(1+q)$. Then for small p and q, $\mathcal{O}(\frac{1}{pq^2})$ samples suffice to distinguish X from Y with a constant probability of success.

It can be easily verified that if one chooses $\frac{64}{pq^2}$ random samples for the experiment, the distinguishing success probability will be 99.99%. Here, in this case, p is $\frac{1}{2}$ and q is the differential bias for the targeted event. At ISPEC 2022 [13] paper, their obtained differential biases are $q_1 = 0.000829$, $q_2 = 0.000793$, and $q_3 = 0.000767$ as mentioned above. The required number of random samples to verify these biases with 99.99% success probability are $\frac{64}{pq_1^2} \approx 2^{27.47}$, $\frac{64}{pq_2^2} \approx 2^{27.60}$ and $\frac{64}{pq_3^2} \approx 2^{27.7}$ respectively. Therefore, given their claim is correct, 2^{28} many random samples are sufficient to verify their obtained differential biases with more than 99.99% success probability.

We observe that their obtained 5-round forward differential biases are incorrect. We use 2^{40} random samples to verify their differential biases. According to the formula $\frac{64}{pq^2}$, any bias higher than $2^{-16.5} \approx 0.000011$ is supposed to be detected using these many samples.

For the input difference positions $\Delta S^0[6][31]$, $\Delta S^0[6]$, [15] and $\Delta S^0[7][9]$, we get the forward differential biases 0.000001577, 0.00000002045 and -0.0000008128 respectively for the output difference position $\Delta S^5[0][18]$ using 2^{40} random samples. This is not only significantly smaller compared to their claim, but also smaller than $2^{-16.5}$. In Table 4, we provide the comparison of their claim and the bias obtained by us.

Therefore, we can claim than none of their claimed distinguisher provides a bias higher than $2^{-16.5} \approx 0.000011$. We have provided a github[4]link for the program of one of the ID positions. The reader can verify the biases by suitably changing the input difference position.

However, note that, the biases obtained by us are not accurate as the number of random samples is not sufficient to verify such small values. Our experimental result only verifies that the biases are not higher than 0.000011.

In the key recovery attack, they used the forward bias 0.000829 to mount the attack. Since this bias is not accurate, so their 8-round key recovery attack becomes invalid.

[4]https://github.com/Rahul150192/Salsa/tree/main

VI. CONCLUSION

In this study, we have reviewed two recent attacks [7], [8], [13] on Salsa as well as improved the attack on reduced round Salsa. We have presented a strategy for constructing a better set of probabilistic neutral bits. We have demonstrated that with complexity $2^{210.38}$, it is possible to recover the secret key of 8-round Salsa. Also, we have shown that the 5-round forward differential biases presented at ISPEC 2022 [13] are inaccurate, and because of that their 8-round key recovery attack for Salsa becomes meaningless.

APPENDIX
PNBs SETS

Here we list the five PNBs sets of size 160 as follows:

$\mathcal{A}_1 = \{4, 5, 6, 7, 8, 9, 10, 12, 11, 89, 82, 13, 83, 84, 14, 85, 90, 86, 88, 93, 15, 91, 94, 87, 95, 92, 16, 17, 50, 18, 51, 19, 139, 52, 140, 141, 53, 20, 142, 64, 21, 54, 65, 143, 66, 22, 55, 144, 67, 23, 56, 160, 145, 68, 161, 24, 57, 174, 146, 175, 162, 255, 69, 25, 58, 103, 115, 147, 116, 104, 163, 176, 70, 26, 59, 148, 117, 105, 71, 27, 60, 149, 177, 164, 36, 118, 106, 28, 37, 61, 159, 204, 72, 150, 119, 62, 63, 29, 178, 205, 237, 30, 31, 38, 73, 107, 165, 192, 151, 193, 179, 206, 238, 39, 108, 166, 239, 207, 120, 180, 224, 40, 194, 152, 240, 121, 74, 186, 225, 167, 199, 231, 187, 208, 181, 249, 233, 232, 241, 123, 155, 134, 112, 138, 129, 200, 99, 122, 188, 196, 248, 223, 0,183, 209, 212, 228, 229, 157, 253\}$,

$\mathcal{A}_2 = \{4, 5, 6, 7, 8, 9, 90, 10, 13, 11, 14, 91, 82, 12, 83, 84, 85, 15, 93, 86, 92, 87, 88, 94, 89, 95, 16, 17, 50, 18, 51, 139, 140, 19, 52, 20, 141, 53, 64, 142, 21, 65, 143, 54, 66, 22, 55, 144, 67, 23, 56, 145, 160, 68, 24, 161, 57, 174, 175, 146, 162, 255, 69,103, 25, 58, 115, 147, 176, 116, 104, 163, 26, 70, 59, 148, 117, 105, 177, 164, 60, 71, 27, 36, 118, 149, 28, 61, 106, 159, 72,204, 37, 150, 29, 62, 63, 119, 178, 165, 107, 30, 31, 38, 73, 205, 237, 206, 238, 179, 151, 192, 120, 166, 39, 152, 193, 224, 108, 74, 231, 207, 239, 180, 249, 186, 187, 225, 167, 208, 194, 240, 109, 199, 209, 226, 200, 181, 250, 251, 241, 154, 112, 188, 218, 190, 133, 130, 98, 252, 217, 170, 157, 173, 97, 49, 227, 96, 99, 197\}$,

$\mathcal{A}_3 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 90, 92, 82, 84, 93, 83, 85, 94, 15, 95, 91, 89, 16, 86, 87, 88, 14, 17, 50, 18, 19, 51, 139, 140, 141, 52, 20, 53, 64, 142, 143, 21, 65, 54, 66, 22, 55, 144, 67, 160, 23, 56, 145, 161, 24, 68, 57, 174, 146, 162, 175, 25, 69, 255, 103, 115, 58, 147, 176, 116, 104, 163, 26, 59, 70, 117, 148, 105, 71, 27, 60, 164, 177, 149, 118, 36, 159, 61, 28, 106, 72, 204, 37, 150, 178, 119, 29, 62, 63, 30, 205, 237, 31, 38, 107, 165, 73, 206, 238, 192, 151, 120, 108, 179, 193, 39, 121, 239, 207, 224, 180, 166, 74, 109, 152, 225, 194, 208, 240, 249, 213, 181, 167, 199, 128, 241, 226, 195, 40, 201, 183, 227, 245, 235, 125, 244, 97, 212, 248, 198, 190, 170, 203, 168, 126, 134, 217, 114, 169, 49\}$,

$\mathcal{A}_4 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 82, 83, 13, 14, 90, 88, 91, 84, 85, 86, 15, 87, 89, 92, 93, 95, 94, 16, 17, 50, 51, 18, 139, 140, 19, 52, 141, 20, 142, 53, 64, 21, 65, 54, 143, 66, 22, 55, 144, 67, 160, 23, 56, 145, 161, 24, 68, 174, 57, 146, 175, 162, 69, 25, 58, 255, 103, 115, 147, 116, 176, 104, 163, 70, 26, 59, 148, 117, 105, 177, 27, 71, 60, 149, 164, 36, 106, 118, 159,$

28, 61, 204, 72, 37, 150, 62, 29, 119, 63, 178, 205, 237, 30, 31, 38, 107, 165, 151, 73, 120, 192, 179, 238, 206, 39, 108, 166, 207, 239, 193, 224, 152, 180, 225, 121, 208, 194, 249, 240, 74, 199, 231, 167, 186, 241, 40, 109, 122, 100, 188, 110, 200, 153, 236, 201, 253, 243, 112, 32, 196, 138, 214, 79, 101, 247, 157, 43, 217, 113, 216, 202},

$\mathcal{A}_5$ = {4, 5, 6, 11, 7, 8, 9, 10, 12, 89, 82, 90, 91, 83, 84, 14, 88, 87, 85, 13, 15, 86, 93, 92, 94, 95, 16, 17, 50, 18, 19, 51, 139, 140, 52, 141, 20, 53, 64, 142, 21, 65, 54, 143, 66, 22, 55, 144, 67, 56, 23, 160, 145, 161, 174, 68, 24, 57, 146, 175, 162, 69, 25, 255, 58, 103, 115, 147, 116, 176, 104, 163, 70, 26, 59, 148, 117, 105, 71, 27, 60, 149, 36, 164, 177, 118, 159, 106, 204, 28, 61, 37, 72, 150, 29, 62, 63, 178, 205, 237, 165, 119, 30, 31, 38, 73, 192, 107, 151, 120, 238, 206, 108, 179, 39, 193, 166, 152, 224, 207, 239, 109, 74,180, 199, 231, 186, 225, 249, 250, 187, 167, 251, 40, 218, 194, 75, 241, 153, 154, 110, 209, 214, 129, 121, 102, 247, 101, 127, 202, 198, 211, 112, 171, 99, 156, 182, 80, 79, 197}.

REFERENCES

[1] J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger, "New features of Latin dances: Analysis of Salsa, ChaCha, and Rumba," in *Proc. Int. Workshop Fast Softw. Encryption*, vol. 5086. Lausanne, Switzerland: Springer, 2008, pp. 470–488.

[2] D. J. Bernstein. (2005). *Salsa20 Specification*. [Online]. Available: http://www.ecrypt.eu.org/stream/salsa20pf.html

[3] D. J. Bernstein, "ChaCha, a variant of Salsa20," in *Proc. Workshop Record SASC*, vol. 8, no. 1, 2008, pp. 3–5.

[4] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New Stream Cipher Designs: The ESTREAM Finalists*. Berlin, Germany: Springer, 2008, pp. 84–97.

[5] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," in *Proc. CRYPTO*, vol. 537. Santa Barbara, CA, USA: Springer, 1990, pp. 2–21.

[6] A. R. Choudhuri and S. Maitra, "Significantly improved multi-bit differentials for reduced round salsa and Chacha," *IACR Trans. Symmetric Cryptol.*, vol. 2016, no. 2, pp. 261–287, Feb. 2017.

[7] M. Coutinho, I. Passos, J. C. G. Vásquez, F. L. L. de Mendonça, R. T. de Sousa, and F. Borges, "Latin dances reloaded: Improved cryptanalysis against Salsa and ChaCha, and the proposal of Forró," in *Proc. ASIACRYPT*, vol. 13791. Taipei, Taiwan: Springer, 2022, pp. 256–286.

[8] M. Coutinho, I. Passos, J. C. G. Vásquez, S. Sarkar, F. L. L. de Mendonça, R. T. de Sousa, and F. Borges, "Latin dances reloaded: Improved cryptanalysis against salsa and ChaCha, and the proposal of Forró," *J. Cryptol.*, vol. 36, no. 3, p. 18, Jul. 2023.

[9] P. Crowley. (2005). *Truncated Differential Cryptanalysis of Five Rounds of Salsa20*. Cryptol. ePrint Arch. [Online]. Available: https://eprint.iacr.org/2005/375

[10] S. Dey and S. Sarkar, "Improved analysis for reduced round salsa and Chacha," *Discrete Appl. Math.*, vol. 227, pp. 58–69, Aug. 2017.

[11] S. Dey, H. K. Garai, S. Sarkar, and N. K. Sharma, "Revamped differential-linear cryptanalysis on reduced round Chacha," in *Proc. EUROCRYPT*, vol. 13277. Trondheim, Norway: Springer, 2022, pp. 86–114.

[12] L. Ding, "Improved related-cipher attack on Salsa20 stream cipher," *IEEE Access*, vol. 7, pp. 30197–30202, 2019.

[13] N. Ghafoori and A. Miyaji, "Differential cryptanalysis of Salsa20 based on comprehensive analysis of PNBs," in *Proc. ISPEC*, vol. 13620. Taipei, Taiwan: Springer, 2022, pp. 520–536.

[14] N. Ghafoori, A. Miyaji, R. Ito, and S. Miyashita, "PNB based differential cryptanalysis of Salsa20 and Chacha," *IEICE Trans. Inf. Syst.*, vol. 106, no. 9, pp. 1407–1422, 2023.

[15] H. Lipmaa and S. Moriai, "Efficient algorithms for computing differential properties of addition," in *Proc. FSE*. Yokohama, Japan: Springer, 2001, pp. 336–350.

[16] S. Maitra, "Chosen IV cryptanalysis on reduced round Chacha and salsa," *Discrete Appl. Math.*, vol. 208, pp. 88–97, Jul. 2016.

[17] S. Maitra, G. Paul, and W. Meier. (2015). *Salsa20 Cryptanalysis: New Moves and Revisiting Old Styles*. Cryptol. ePrint Arch. [Online]. Available: https://eprint.iacr.org/2015/217

[18] I. Mantin and A. Shamir, "A practical attack on broadcast RC4," in *Proc. FSE*, vol. 2355. Yokohama, Japan: Springer, 2001, pp. 152–164.

[19] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Proc. EUROCRYPT*, vol. 765. Lofthus, Norway: Springer, 1993, pp. 386–397.

[20] S. Miyashita, R. Ito, and A. Miyaji, "PNB-focused differential cryptanalysis of ChaCha stream cipher," in *Proc. ACISP*, vol. 13494. Wollongong, NSW, Australia: Springer, 2022, pp. 46–66.

[21] M. Robshaw and O. Billet, *New Stream Cipher Designs: The ESTREAM Finalists*, vol. 4986, Springer, 2008.

[22] Z. Shi, B. Zhang, D. Feng, and W. Wu, "Improved key recovery attacks on reduced-round Salsa20 and ChaCha," in *Proc. ICISC*, vol. 7839. Seoul, South Korea: Springer, 2012, pp. 337–351.

CHANDAN DEY received the M.Sc. degree in mathematics from Visva-Bharati, Santiniketan, West Bengal, and the Ph.D. degree in mathematics from the Indian Institute of Technology Madras, Chennai, India, in 2023. His research interests include the design and analysis of symmetric key ciphers.

SABYASACHI DEY received the Ph.D. degree in mathematics from the Indian Institute of Technology Madras, Chennai, India, in 2018. He is currently an Assistant Professor with the Birla Institute of Technology and Science (BITS), Pilani, India. His main research interest includes symmetric key cryptology.

RAHUL GIRME received the M.Sc. degree in mathematics from Pune University, in 2015. He is currently pursuing the Ph.D. degree in mathematics with the Indian Institute of Technology Madras, Chennai, India. His research interests include the design and analysis of stream ciphers and block ciphers.

SANTANU SARKAR received the Ph.D. degree in mathematics from the Indian Statistical Institute, Kolkata, India, in 2011. He was a Guest Researcher with the National Institute of Standards and Technology (NIST). He is currently a Professor with the Indian Institute of Technology, Madras, India. His main research interests include cryptology and number theory.