IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.46586/tosc.v0.i0.0-0

Cryptanalysis of Reduced Round ChaCha – New Attack & Deeper Analysis

Sabyasachi Dey[1], Hirendra Kumar Garai[1], Subhamoy Maitra[2]

[1] Birla Institute of Technlogy and Science - Pilani, Hyderabad Campus, India,
sabya.ndp@gmail.com, p20190465@hyderabad.bits-pilani.ac.in

[2] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700108, India,
maitra.subhamoy@gmail.com

Abstract. In this paper we present several analyses on ChaCha, a software stream cipher. First, we consider a divide-and-conquer approach on the secret key bits by partitioning them. The partitions are based on multiple input-output differentials to obtain a significantly improved attack on 6-round ChaCha256 with a complexity of $2^{99.48}$. It is 2^{40} times faster than the currently best known attack. This is the first time an attack on a round reduced ChaCha with a complexity smaller than $2^{k/2}$, where the secret key is of k bits, has been successful.
Further, all the attack complexities related to ChaCha are theoretically estimated in general and there are several questions in this regard as pointed out by Dey, Garai, Sarkar and Sharma in Eurocrypt 2022. In this regard, we propose a toy version of ChaCha, with a 32-bit secret key, on which the attacks can be implemented completely to verify whether the theoretical estimates are justified. This idea is implemented for our proposed attack on 6 rounds. Finally, we show that it is possible to estimate the success probabilities of these kinds of PNB-based differential attacks more accurately. Our methodology explains how different cryptanalytic results can be evaluated with better accuracy rather than claiming that the success probability is significantly better than 50%.

Keywords: Stream cipher, ARX, ChaCha, Probabilistic Neutral Bits (PNBs), Differential attack

1 Introduction

ChaCha was designed in 2008 by Bernstein [Ber08a] as a variant of Salsa, which was one of the finalists of eSTREAM project (see [Ber08b] for details). ChaCha is an ARX cipher, i.e., the operations in the cipher involve Addition, Rotation and XOR, which are executed very fast in a Central Processing Unit (CPU). ChaCha is presently being used in many of the standards and quite safe with the number of rounds proposed. However, it is important to study the cipher with reduced rounds for cryptanalytic purposes. There are several important results in this direction for around a decade.

1.1 Summary of the existing attacks

Most of the attacks on the reduced round versions of this cipher are based on differential cryptanalysis.

- The fundamental attack was proposed in 2008 in [AFK+08]. The authors identified weaknesses of 6 and 7-round versions of ChaCha256. Moreover, they also provided cryptanalytic results on ciphers with similar structure such as ChaCha128, Salsa256,

Salsa128 and Rumba. In this work, the idea of Probabilistic Neutral Bits (PNB) was introduced for the first time. For ChaCha256, the attack complexities in this technique are 2^{139} for 6 rounds and 2^{248} for 7 rounds.

- Later in 2012, an idea called Column Chaining Distinguisher (CCD) based on the PNB idea was introduced in [SZFW12].
- In 2015, Maitra [Mai16] proposed an idea where the attacker would choose the IV's based on key guessing. This idea improved the attack complexity for 7-round ChaCha to $2^{238.9}$. Although this work did not present an attack on the 6-round version of the cipher.
- In a similar line, in 2016, the authors of [CM16] used the idea of linear extension of the differential attack. This approach led to the discovery of the distinguishers up to the 5-th round of ChaCha.
- Further results were identified in [DS17] in 2017, where they obtained a better set of probabilistic neutral bits. As a result the attack complexity for 6-round ChaCha was reduced to $2^{235.2}$.
- In Crypto 2020, the authors of [BLT20] discovered a single bit distinguisher in the 3.5 round of ChaCha. This helped to provide a significant improvement in cryptanalysis, and the attack complexity reduced to $2^{230.86}$. Further, with the help of this distinguisher the authors produced a partial key recovery attack on 6-round ChaCha, which recovered 36 bits with complexity 2^{77}.
- The same distinguisher was also observed in [CN20] independently in the same year. They presented an attack on the 7-round ChaCha with complexity $2^{231.9}$. Moreover, a distinguishing attack could be identified on six rounds with complexity 2^{75} and a key recovery attack with complexity 2^{102}. It should be noted that the key recovery attack with complexity 2^{102} mentioned in [CN20] is incorrect. The correct figure should be 2^{210} that we mention in Table 2 and the discussion in Section 3.1.
- In Eurocrypt 2021, the authors of [CN21] presented several new single bit distinguishers for 3.5 rounds. With this, they claimed to produce further improvement in the key recovery attack and a distinguisher for 7-round ChaCha256.
- However, in the next year, the authors of [DDSM22] showed that several of the claimed distinguishers of [CN21] are incorrect, which invalidates the improvement in their key recovery.
- Very recently, in 2022, the authors of [DGSS22] provided further improvements in the key recovery attacks of 7-round ChaCha256 by partitioning the key bits into memory key bits and non-memory key bits. This work produced an attack with complexity 2^{221}. For 6-round ChaCha128 they could produce an attack with complexity 2^{81} and also on 6.5 round with complexity 2^{123}.

1.2 Organisation & contribution

The outline of this paper is as follows. We present an improved attack which is better in terms of complexity, acquire the experimental findings using a toy version of ChaCha, and examine the success probabilities of our attack as well as the other existing attacks with further depth and accuracy.

- In section 2, we provide the background and related materials to the work, that include the structure of the cipher and the existing cryptanalytic ideas using probabilistic neutral bits (PNB).

Table 1: Notations

Symbol	Description
X	The state matrix of the cipher consisting of 16 words
X^r	State matrix after application of r ChaCha Round functions
X_i	i-th word of the state matrix X
$X_i[j]$	j-th bit of i-th word in matrix X
$x \boxplus y$	Addition of x and y modulo 2^{32}
$x \boxminus y$	Subtraction of x and y modulo 2^{32}
$x \oplus y$	Bitwise XOR of x and y
$x \lll n$	Rotation of x by n bits to the left
$x \ggg n$	Rotation of x by n bits to the right
$\Delta_i^r[j]$	XOR difference after r-th round of the j-th bit of the i-th word of X and X'
$(\mathcal{ID}, \mathcal{OD})$	Input Difference-Output Difference
k	Key length in bits

- In section 3, we begin with the analysis of the previous attacks on 6-round ChaCha256. In subsection 3.2 we present a novel cryptanalytic idea using multiple $(\mathcal{ID}, \mathcal{OD})$ pairs, and apply this technique on 6-round ChaCha256 to obtain a significantly improved attack complexity over the previously existing ones. This is the first time an attack complexity significantly less than $2^{k/2}$ is reported, where the secret key is of k bits. Note that the previous attacks, where the complexities were claimed to be less than $2^{k/2}$, were incorrect and we will explain that as and when required.

  While such an approach was not exploited against ChaCha so far, accumulating several biases to mount improved attacks is quite well known in literature. For example, the work of [ABP+13] exploited the idea of accumulating several biases (see [GMPS14] and the references therein) to mount an attack on RC4. In fact, the cryptanalysis of [ABP+13] is one of the prime reasons RC4 is being replaced by ChaCha in various standardized encryption systems. Also, similar ideas have been used in [SG18], [SGSL18].

- To concretely understand the complexity of the complete attack, in section 4, we present a toy version of ChaCha with 32-bit secret key. We implement the usual cryptanalytic attack approaches to achieve a better estimation of the complexities, false alarm error probabilities and the success probabilities. We also implement our new cryptanalytic technique and compare the complexity with the usual attack approaches.

- Lastly in section 5, we present a theoretical approach to identify more accurate ranges for the success probabilities for the PNB-based techniques. We use this theory to compare the practical results obtained from the toy cipher, and that matches convincingly. We exploit this calculation to refine the success probabilities of the attacks presented in [AFK+08] and its modified version with chosen IV approach by Maitra [Mai16].

2 Structure of ChaCha256 and Differential Attack Idea

Let us first explain the design of the cipher. The key stream generation machinery of ChaCha considers an input (the secret key) of size 256-bit (k), a constant of size 128-bit (c) along with the initialization vector (IV) v of size 128-bit (3 nonces and one counter)

which are divided in 16 blocks of size 32-bit each. They are organised in a 4×4 matrix form (X). Each 32-bit block is conventionally called a word. The first row is filled by the constants $c = (c_0, c_1, c_2, c_3)$, second and third row contains the key $k = (k_0, k_1, \ldots, k_7)$ and the last row has the initial vectors (IVs) $v = (t_0, v_0, v_1, v_2)$. The four constants are fixed as $c_0 = \texttt{0x61707865}, c_1 = \texttt{0x3320646e}, c_2 = \texttt{0x79622d32}, c_3 = \texttt{0x6b206574}$. That is, the initial state matrix format is as following:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & v_0 & v_1 & v_2 \end{pmatrix}.$$

This is denoted by X^0 (or sometimes by X) which goes through R many ChaCha `Round` functions [Ber08a]. The updated version of X^0 after r rounds is denoted by X^r. After the full R rounds of execution, the final state X^R is added word-wise (modulo 2^{32}) to the initial state X^0 forming the key stream Z, i.e., $Z = X^0 \boxplus X^R$.

Each of the ChaCha `Round` is formulated with the help of `quarterround` function which itself consists of four ARX functions. The four ARX operations of each `quarterround` function, which transforms a vector (a, b, c, d) to (a'', b'', c'', d'') via (a', b', c', d') is given by the following equations:

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

The odd numbered ChaCha `Round` is called the `column` round due to the fact that it updates the four column vectors (X_0, X_4, X_8, X_{12}), (X_1, X_5, X_9, X_{13}), $(X_2, X_6, X_{10}, X_{14})$, and $(X_3, X_7, X_{11}, X_{15})$ of the state matrix X. On the other hand the even numbered ChaCha `Round` is known as `diagonal` round, as the the diagonal vectors $(X_0, X_5, X_{10}, X_{15})$, $(X_1, X_6, X_{11}, X_{12})$, (X_2, X_7, X_8, X_{13}), and (X_3, X_4, X_9, X_{14}) of the matrix X are updated.

Generally one can reverse back to r-round state X^r from the $r+1$-round state X^{r+1} with the help of the reverse `quarterround` function, which transforms the vector (a'', b'', c'', d'') to (a, b, c, d) via (a', b', c', d') as follows:

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

Further details regarding these operations are available in [Ber08a].

Existing idea of Differential Attack. We denote the j-th bit of the i-th word of the state matrix X after r rounds by $X_i^r[j]$. In the differential attack against ChaCha, we apply an input difference $\Delta_i^0[j]$ to the j-th bit of the i-th word of initial state matrix X (which is by notation actually X^0) producing X'. To be precise, injecting the input difference is basically complementing that bit of the respective state matrix. That is, $\Delta_i^0[j] = X_i[j] \oplus X_i'[j]$. Now, the round function is applied on both X and X' for r rounds. In X^r and X'^r, the difference is observed at the q-th bit of p-th word, i.e, $\Delta_p^r[q] = X_p^r[q] \oplus X'^r_p[q]$. We compute the probability $\Pr(\Delta_p^r[q] = 0 | \Delta_i^0[j] = 1)$, which we write in the form $\frac{1}{2}(1 + \epsilon_d)$, where ϵ_d is called the forward bias. We aim to find an $(\mathcal{ID}, \mathcal{OD})$ pair $(\Delta_i^0[j], \Delta_p^r[q])$ for which we have a higher value of ϵ_d and consequently use it as a distinguisher.

In this approach of differential attack, the idea of probabilistic neutral bits (PNBs) [AFK+08] plays a vital role in the backward direction. From the output key streams $Z = X \boxplus X^R$ and $Z' = X' \boxplus X'^R$, to find a PNB, we proceed as follows. A key bit from X, X' is complemented and $\overline{X}, \overline{X'}$ are produced respectively. Then we compute $Z - \overline{X}, Z' - \overline{X'}$ and execute the reverse round function by $R - r$ many rounds on both of them and consequently achieve the matrices Y and Y' respectively. Let $\Gamma_p[q] = Y_p[q] \oplus Y'_p[q]$. Now if the probability of the event $\Gamma_p[q] = \Delta^r_p[q]$ demonstrates a bias which is higher than a predetermined threshold γ, we call the complemented key bit a probabilistic neutral bit (PNB). Otherwise we call it a significant key bit or non-PNB. In the pre-processing stage of the attack, the probabilistic neutral bits are identified by estimating the above mentioned probability through experiments.

Next in the actual attack, the attacker collects N samples of output key streams Z, Z', assigns random values to the PNBs of X, X', and first aims to guess the significant key bits correctly. Now let us compare the scenarios when the significant key bits are correctly guessed and when they are not. We denote by $\hat{X}, \hat{X}'$ the states where the significant key bits have correct values and the PNBs are random. We run the reverse round operations by $R - r$ rounds on $Z - \hat{X}, Z' - \hat{X}'$, and achieve $\hat{Y}, \hat{Y}'$. We observe the difference $\hat{\Gamma}_p[q] = \hat{Y}_p[q] \oplus \hat{Y}'_p[q]$. Let $\Pr(\hat{\Gamma}_p[q] = 0) = \frac{1}{2}(1 + \hat{\epsilon})$. On the other hand, $\tilde{X}, \tilde{X}'$ are two states where both significant bits as well as the PNBs are random. We compute $Z - \tilde{X}, Z' - \tilde{X}'$ and run the reverse round operation by $R - r$ rounds to achieve $\tilde{Y}, \tilde{Y}'$, check $\tilde{\Gamma}_p[q] = \tilde{Y}_p[q] \oplus \tilde{Y}'_p[q]$ and compute $\Pr(\tilde{\Gamma}_p[q] = 0) = \frac{1}{2}(1 + \tilde{\epsilon})$. Between $\hat{\epsilon}$ and $\tilde{\epsilon}$, $\hat{\epsilon}$ would have a noticeable value, since the significant key bits are correct, but $\tilde{\epsilon}$ would be approximately 0. After successfully identifying the significant bits, we may recover the PNBs by exhaustive search. The bias of the event $(\hat{\Gamma}_p[q] = \Delta_p[q])$ is usually called the backward bias and denoted by ϵ_a.

Now under some assumptions of independence (which are approximately valid and logical),

$$\Pr(\hat{\Gamma}_p[q] = 0) = \Pr(\hat{\Gamma}_p[q] = \Delta_p[q]) \cdot \Pr(\Delta_p[q] = 0) + \Pr(\hat{\Gamma}_p[q] \neq \Delta_p[q]) \cdot \Pr(\Delta_p[q] \neq 0)$$

$$= \frac{(1 + \epsilon_a)}{2} \cdot \frac{(1 + \epsilon_d)}{2} + \frac{(1 - \epsilon_a)}{2} \cdot \frac{(1 - \epsilon_d)}{2} \approx \frac{(1 + \epsilon_a \cdot \epsilon_d)}{2}$$

Therefore the bias $\hat{\epsilon}$ can be approximated by $\epsilon_a \epsilon_d$.

Complexity of the attack. We use hypothesis testing to distinguish the correct guess of significant keys from a wrong guess and find the complexity. Consider the following two hypotheses

- H_0: The guessed significant key bits are incorrect, i.e., the bias related to the key bits is 0.
- H_1: The guessed significant key bits are correct, i.e., their combined bias is $\hat{\epsilon}$.

From our N samples, we keep track of how many times the observed difference $\hat{\Gamma}_p[q] = 0$. Suppose the count is x. Hence, for a threshold T, a reasonable decision rule will be of the form:

$$\text{Reject } H_0 \text{ if } x > T$$
$$\text{Retain } H_0 \text{ if } x \leqslant T$$

In this regard we have the following two types of errors.

1. Type I error: Where the null hypothesis (H_0) is rejected in spite being true, i.e., for an incorrect guess of significant key bits, we achieve $x > T$ (False Alarm).

2. Type II error: Where the null hypothesis is retained in spite of being false (Non detection). Here, for the correct guess of the significant key bits, we achieve $x \leqslant T$.

The authors of [AFK+08] restricted the probability of non-detection error to be less than or equal to 1.3×10^{-3}. Based on this, using Neyman-Pearson lemma, the value of N is derived as:

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

Here α is such that the probability of false alarm is $2^{-\alpha}$. Let us denote the total key size of the cipher by k bits and the number of significant key bits in the key be m, then the complexity formula given by [AFK+08] was $2^m \cdot N + 2^{k-\alpha}$. However, in the work [DGSS22] it has been shown that the accurate complexity formula should be $2^m \cdot N + 2^{k-\alpha} + 2^{k-m}$.

3 Critical analysis of the previous works and a novel attack on 6-round ChaCha256

Based on the correction provided by [DGSS22] in the complexity formula, we identify that there is a limitation in the fundamental cryptanalytic approach in [AFK+08]. In this approach, the overall complexity of the attack can never go below $2^{k/2}$. Let $k - m$ be the number of PNBs in the key. As mentioned above, the complexity is then given by $2^m \cdot N + 2^{k-\alpha} + 2^{k-m}$. If $m < k/2$ then, $k - m > k/2$. Thus, $2^m \cdot N + 2^{k-m} + 2^{k-\alpha} > 2^{k/2}$. On the other hand, $m \geqslant k/2$ implies, $2^m \cdot N \geqslant 2^{k/2}$. So, $2^m \cdot N + 2^{k-m} + 2^{k-\alpha} \geqslant 2^{k/2}$. Therefore, for both the cases $m < k/2$ and $m \geqslant k/2$, the complexity is greater than or equal to $2^{k/2}$.

This is where we come up with a novel cryptanalytic idea of exploiting multiple $(\mathcal{ID}, \mathcal{OD})$ pairs to bring down the complexity below $2^{k/2}$, for k-bit key that we present in this section, particularly in subsection 3.2.

3.1 Correcting the complexity calculations of some previous works

In the literature there are several works in which the achieved complexity seems to be less than $2^{k/2}$. The reason for this is that the authors have used the complexity formula given in [AFK+08]. In fact, there are several previous works on the differential attacks on ChaCha whose complexities should be significantly different from their claim, if we use the complexity formula given in [DGSS22].

Explanations for the miscalculation in the complexity calculation. The key recovery is done in two stages. In the first stage, we recover the significant key bits, for which the required complexity is $2^m \cdot N + 2^{k-\alpha}$. In the second stage we recover the PNBs by exhaustive search, for which the complexity is 2^{k-m}. In the formula given by [AFK+08], this 2^{k-m} term was missing, i.e., the complexity for recovering the PNBs was not included in the formula, which has later been incorporated in [DGSS22]. In the complexity analysis of the following works, 2^{k-m} is significantly higher than $2^m \cdot N + 2^{k-\alpha}$. So the actual complexity is significantly higher than their claims. This we explain with little more details before proceeding further.

- **[AFK+08]:** In the work of [AFK+08] itself, the authors have attacked the 7-round and 6-round version of ChaCha256. For the 6-round version attack, the authors used 147 PNBs in their attack, i.e., the complexity for the second stage is $2^{k-m} = 2^{147}$. Unfortunately, because of considering the formula of [AFK+08], the authors claimed that the attack complexity as $2^m \cdot N + 2^{k-\alpha} = 2^{139}$, which is actually higher.

- **[SZFW12]:** In the last step of their attack on 6-round ChaCha256, they have to recover 139 PNBs, which requires a complexity of 2^{139}. However, the complexity claimed was 2^{136}.
- **[CM16]:** In FSE 2016, Rai Choudhuri and Maitra attacked 6-round ChaCha256 using multiple bit distinguishers and provided three different attack complexities, which are $2^{131.40}, 2^{129.53}$ and $2^{127.5}$. In these attacks, the number of PNBs are 159, 161 and 166 respectively. Therefore, the actual complexities should be $2^{159}, 2^{161}$ and 2^{166} respectively.
- **[CN20]:** Next in 2020, the authors of [CN20] claimed to further improve the attacks on 6-round ChaCha256 by using a better distinguisher, and proposed a cryptanalytic idea achieving complexities like 2^{102} and 2^{104}. Again, they have used 210 and 212 PNBs respectively in these attacks, because of which the actual complexities should be 2^{210} and 2^{212}.

We mention the claimed complexities and the actual complexities of all these attacks in Table 2, along with the complexity of our newly proposed technique in the next subsection.

3.2 Our cryptanalytic technique involving multiple $(\mathcal{ID}, \mathcal{OD})$ pairs

Now we propose a novel cryptanalytic technique using multiple $(\mathcal{ID}, \mathcal{OD})$ pairs with the help of which one can achieve a complexity less than $2^{k/2}$. Assume that we have q different $(\mathcal{ID}, \mathcal{OD})$ pairs each of which give a high bias (How the $(\mathcal{ID}, \mathcal{OD})$ pairs are obtained and how the number q is selected is explained in subsection 3.4). Here we exploit all these pairs in the attack. Let us denote these pairs as $(\mathcal{ID}_i, \mathcal{OD}_i)$, where $i \in \{1, 2, \ldots, q\}$.

Pre-processing Stage: Partitioning the key bits into $q + 1$ subsets

In this approach, we partition the set of all key bits into $(q + 1)$ subsets $S_1, S_2, \ldots, S_{q+1}$, such that for $i = 1, 2, \ldots, q$, S_i is the set of significant key bits corresponding to $(\mathcal{ID}_i, \mathcal{OD}_i)$. Further, S_{q+1} is the set of all remaining key bits.

Stage 1: We put the input difference at $\mathcal{ID}_1$, run r ChaCha Round functions and observe the difference at $\mathcal{OD}_1$. Let us call it $\Delta_{\mathcal{OD}_1}$. Then, we run the algorithm on both the matrices $R - r$ more rounds (i.e., R rounds in total), and generate Z, Z'. Now by changing a single key bit in X and X', we generate $\overline{X}$ and $\overline{X'}$. We compute $Z - \overline{X}$ and $Z' - \overline{X'}$, run the reverse round by $R - r$ rounds and check the differences at $\mathcal{OD}_1$. Let us call it $T_{\mathcal{OD}_1}$. We repeat this process for each key bit. If the bias in the event $(\Delta_{\mathcal{OD}_1} = T_{\mathcal{OD}_1})$ is less than a predetermined threshold γ_1, we consider the key bit to be in S_1, i.e., significant bits corresponding to $(\mathcal{ID}_1, \mathcal{OD}_1)$. We repeat this process for each key bit and thus construct S_1.

For $i = 2$ to q, we do the following:

Stage i: Similarly as above, by putting the input difference at $\mathcal{ID}_i$, we run both X and X' by r rounds and observe the difference at $\mathcal{OD}_i$, and call it $\Delta_{\mathcal{OD}_i}$. Then we generate Z, Z'. Consequently, for each of the key bits which are not in any of $S_1, S_2, \ldots, S_{i-1}$, we proceed as follows. Changing the key bit of the initial matrices, we achieve $\overline{X}$ and $\overline{X'}$, then compute $Z - \overline{X}$ and $Z' - \overline{X'}$ and run the reverse algorithm by $R - r$ rounds to check the difference $T_{\mathcal{OD}_i}$. The bits for which the bias of the event $(\Delta_{\mathcal{OD}_i} = T_{\mathcal{OD}_i})$ is less than a predetermined threshold γ_i, are included in S_i.

Stage $q+1$: After the construction of $S_1, S_2, \ldots, S_q$, the remaining key bits which are not in any of the sets S_1, S_2, $\ldots$, S_q, are assembled in the set S_{q+1}.

By the construction process it is clear that since during any stage S_i, key bits are chosen from those which are not in $S_1, S_2, \ldots, S_{i-1}$, so its intersection with $S_1 \cup S_2, \ldots \cup S_{i-1}$ is empty. Since this is true for any $i \in \{1, 2, \ldots, (q+1)\}$, $S_1, S_2, \ldots, S_{q+1}$ are all disjoint.

Online Phase

In the differential attack model we assume that the attacker has access over the IVs. The first stage of the cryptanalytic method consists of collection of the data.

Data Collection: For all $i = 1, 2, \ldots, q$, the following steps are executed. By assigning N_i different pairs of IVs (v, v') such that the difference is at the position $\mathcal{ID}_i$, the attacker runs the algorithm through R rounds and collects the outputs Z, Z'. Thus, the data complexity is $\sum_{i=1}^{q} N_i$.

Recovering the key bits: The key bits are recovered in $(q+1)$ stages. For each of $i = 1$ to q, in the i-th stage, we recover the key bits of S_i.

Stage 1: For each of the collected N_1 pairs of Z, Z' which are generated from X, X' with the input difference at $(\mathcal{ID}_1, \mathcal{OD}_1)$, the steps are as follows. The attacker guesses the key bits of S_1 and assigns random values in the remaining $k - |S_1|$ key bits. Thus, two states $\tilde{X}$ and $\tilde{X}'$ are produced. Now the attacker runs $Z - \tilde{X}$ and $Z - \tilde{X}'$ for $R - r$ rounds and checks the difference at $\mathcal{OD}_1$ position. Let us denote it by $\tilde{T}_{\mathcal{ID}_1}$. Out of N_1 pairs, if the number of times when $\tilde{T}_{\mathcal{ID}_1} = 0$ occurs is more than a predetermined threshold T_1, the guess is considered to be correct for the S_1 key bits, and the attack proceeds to stage 2. If it does not cross the threshold, the attacker takes a new guess of the S_1 key bits and repeats the procedure.

For $i = 2$ to q, the following procedure is performed:

Stage i: Till the beginning of the i-th stage, the attacker has already recovered the key bits of $S_1, S_2, \ldots, S_{i-1}$. Now for each of the N_i pairs of Z, Z', corresponding to $(\mathcal{ID}_i, \mathcal{OD}_i)$, the attacker puts the already recovered values for the key bits of $S_1 \cup S_2 \cup \cdots \cup S_{i-1}$, and guesses S_i, puts random values in the remaining $(k - |S_1| - |S_2| - \cdots - |S_i|)$ key bits. Then, $Z - \tilde{X}$ and $Z - \tilde{X}'$ are run backwards by $R - r$ rounds and their difference at $\mathcal{OD}_i$ position is observed, which we denote by $\tilde{T}_{\mathcal{ID}_i}$.

If out of these N_i pairs, the count that $\tilde{T}_{\mathcal{ID}_i} = 0$ occurs crosses a predetermined threshold T_i, the guess for S_i key bits is considered to be correct, and the algorithm proceeds to stage $i+1$. Otherwise, we proceed with a new guess and then the process is repeated.

Stage $q+1$: In this stage the remaining key bits S_{q+1} are obtained by the exhaustive search.

3.3 Complexity and error probability in our new approach

Here we present the complexity analysis in line of [AFK+08], with certain modifications. We aim to choose the data complexity in such a way that the probabilities of the two types of errors are within our desired limit.

Non-detection error in each stage: In each of the first q stages, the non-detection error can occur, i.e., the threshold may not cross even if the guess is correct. For simplicity, we aim to keep the non-detection error probability same in each stage. Let us denote this by $\Pr^*_{nd}$ and the overall non-detection probability by $\Pr_{nd}$. First, we find the relation between $\Pr^*_{nd}$ and $\Pr_{nd}$. In each individual stage, the probability that the correct key bits are detected is $(1 - \Pr^*_{nd})$. Therefore, the probability that correct key bits are detected in all the q stages is $(1 - \Pr^*_{nd})^q$. Thus,

$$\Pr_{nd} = 1 - (1 - \Pr^*_{nd})^q \approx \ q\Pr^*_{nd}. \tag{3}$$

False alarm error in each stage: Similarly, a false alarm can occur in each of the q stages. Let us denote the error probability of the i-th stage as $\Pr_{fa_i}$. If there is a false alarm in the i-th stage, we proceed to the $(i+1)$-th stage. However, in our attack approach, we will consider the probability of false alarm error so small that it would have negligible contribution to the overall complexity.

Complexity in the i-th stage: In the i-th stage, the complexity to find the correct significant key bits corresponding to the i-th $(\mathcal{ID}, \mathcal{OD})$ pair, can be expressed as $2^{m_i} \cdot N_i$. Particularly, in our attack we assign such values of α that the false alarm error does not have significant influence on the complexity. Therefore the total complexity can be written as

$$\sum_{i=1}^{q} 2^{m_i} \cdot N_i + 2^{m_{q+1}}. \tag{4}$$

Derivation of N_i: Here we derive the data complexity of each stage with the aim that the overall error probability of non-detection has the same upper bound as the previous works (1.3×10^{-3}). Let a random variable $\mathcal{X}$ follow a binomial distribution with N_i trials. If the null hypothesis is true, i.e., the guessed significant key bits are incorrect, then $p = \frac{1}{2}$. If the alternative hypothesis is true, we have $p = \frac{1}{2}(1 + \epsilon)$. Now, we have to distinguish between these two distributions. We approximate both of these by normal distributions. Then we have to decide a threshold T_i for which the two errors mentioned above are upper bounded by certain desired values, i.e., $\Pr^*_{nd}$ and $\Pr_{fa_i} = 2^{-\alpha_i}$. Let us denote the two random variables corresponding to H_0 and H_1 as $\mathcal{X}_0$ and $\mathcal{X}_1$.

When H_0 is true, we have $p = \frac{1}{2}$ and when the alternative H_1 is true we have $p = \frac{1}{2}(1 + \epsilon)$. Here the test statistic used is $\mathcal{Z}_j = (\mathcal{X}_j - \text{ mean})/\text{standard deviation}$ $(j = 1, 2)$, which follows a standard normal distribution. We consider the probability of false alarm error to be upper bounded by $2^{-\alpha_i}$, i.e.,

$$\Pr\left[\mathcal{Z}_0 > \frac{T_i - N_i/2}{\sqrt{N_i/4}}\right] \leqslant 2^{-\alpha_i}. \tag{5}$$

On the other hand, the probability of non-detection is $\Pr^*_{nd}$. Hence

$$\Pr^*_{nd} = \Pr\left[\mathcal{Z}_1 \leqslant \frac{T_i - N_i(1+\epsilon)/2}{\sqrt{N_i(1-\epsilon^2)}/2}\right] = \Phi\left[\frac{T_i - N_i(1+\epsilon)/2}{\sqrt{N_i(1-\epsilon^2)}/2}\right].$$

Therefore, $\dfrac{T_i - N_i(1+\epsilon)/2}{\sqrt{N_i(1-\epsilon^2)}/2} = \Phi^{-1}[\Pr^*_{nd}]$. Thus, T_i can be expressed as follows.

$$T_i = \frac{N_i(1+\epsilon)}{2} + \frac{\Phi^{-1}[\Pr^*_{nd}](\sqrt{N_i(1-\epsilon^2)})}{2} \tag{6}$$

Using this value of the expression from Equation 6 into Equation 5 we get,

$$\Pr\left[\mathcal{Z}_0 > \Phi^{-1}[\Pr^*_{nd}](\sqrt{1-\epsilon^2}) + \sqrt{N_i}\epsilon\right] \leqslant 2^{-\alpha_i}$$

Since $\int_y^\infty e^{-\frac{x^2}{2}} dy \leqslant e^{-\frac{y^2}{2}}$, we have,

$$\Pr\left[\mathcal{Z}_0 > \Phi^{-1}[\Pr^*_{nd}](\sqrt{1-\epsilon^2}) + \sqrt{N}\epsilon\right] \leqslant e^{-\frac{\left(\Phi^{-1}[\Pr^*_{nd}](\sqrt{1-\epsilon^2})+\sqrt{N}\epsilon\right)^2}{2}}$$

$$\text{or, } e^{-\frac{\left(\Phi^{-1}[\Pr^*_{nd}](\sqrt{1-\epsilon^2})+\sqrt{N}\epsilon\right)^2}{2}} = \frac{1}{2^{\alpha_i}}$$

Taking natural logarithm on both sides and using the equality $\frac{1}{q}\Pr_{nd} = \Pr^*_{nd}$ we have,

$$N_i \approx \left(\frac{\sqrt{(\alpha_i)\ln 4} - \Phi^{-1}[\frac{1}{q}\Pr_{nd}]\sqrt{1-\epsilon^2}}{\epsilon}\right)^2 \tag{7}$$

3.4 Key recovery of 6-round ChaCha256

We use this divide-and-conquer kind of approach to produce an attack against the 6-round version of ChaCha256. In this process we use three $(\mathcal{ID}, \mathcal{OD})$ pairs.

Choosing the $(\mathcal{ID}, \mathcal{OD})$ pairs

In section 6.1 of [BLT20], the authors reported four distinguishers for the 3.5-th round which they found experimentally. Among these four, we use three pairs ($q = 3$) in our attack, which are as follows: $(\mathcal{ID}_1, \mathcal{OD}_1)$: $(\Delta^0_{12}[6], \Delta^{3.5}_1[0])$, $(\mathcal{ID}_2, \mathcal{OD}_2)$: $(\Delta^0_{13}[6], \Delta^{3.5}_2[0])$, $(\mathcal{ID}_3, \mathcal{OD}_3)$: $(\Delta^{(0}_{14}[6], \Delta^{3.5}_3[0])$. However the authors have used suitable IVs such that the number of differences after the first round is minimum and thus obtained a bias of 0.00317 after 3.5 round. To achieve one suitable IV we need 2^5 random trials on average. To avoid these 2^5 extra trials we loose the minimum difference criterion after one round. As a result we achieve a bias (ϵ_d) of 0.0005 by experimenting over 2^{40} random key-IV pairs. Note that we want to keep the non-detection error probability same as in the previous works, i.e., $\Pr_{nd} = 1.3 \times 10^{-3}$. Thus, $\Phi^{-1}[(1 \times \Pr_{nd})/q] = \Phi^{-1}[(1 \times (1.3 \times 10^{-3}))/3]) \approx -3.4$. Now we discuss how many $(\mathcal{ID}, \mathcal{OD})$ pairs do we need to consider and in which order so that we can produce the best attack. To explain this, we study the complexity calculation given in formula Equation 4.

Explanation for using three $(\mathcal{ID}, \mathcal{OD})$ pairs: We consider a q to be suitable, if the complexity of the last stage (to recover the remaining key bits of S_{q+1} via exhaustive search) is almost negligible compared to the complexity of the first q stages (to recover $S_1, S_2, \ldots, S_q$). For example, in our case if we had taken $q = 2$, then the complexity to recover S_1 and S_2 would have been approximately 2^{99}, whereas the complexity of the last stage might be as high as 2^{142}, since we had to recover 142 remaining bits in the last stage. So, $q = 2$ is not a suitable choice. Thus we go for $q = 3$, where we see that the complexity at the last stage is 2^{92}, which is much less than 2^{99}. Further, we can proceed for $q = 4$, but that does not improve the overall complexity further, since the complexity of the first stage still remains 2^{99}.

Choosing the order of the $(\mathcal{ID}, \mathcal{OD})$ pairs: In this expression, among all the terms of the form $2^{m_i} \cdot N_i$ the first term $2^{m_1} \cdot N_1$ plays the vital part and the other terms are significantly smaller than this, i.e., the contribution of those terms in the overall complexity

is much less. The reason is, in the later stages the number of key bits we recover is less, and also, since the already recovered bits are correctly guessed, the bias increases and therefore N_i decreases. So we aim to make the term $2^{m_1} \cdot N_1$ as small as possible.

Thus, we have to focus on which $(\mathcal{ID}, \mathcal{OD})$ pair should be considered as the first pair $(\mathcal{ID}_1, \mathcal{OD}_1)$, since it has the primary role in deciding the complexity. Among all the pairs, we choose the $(\mathcal{ID}, \mathcal{OD})$ pair for which we can achieve the minimum value for $2^{m_1} \cdot N_1$. Particularly in our case, since each of the three pairs produces same backward and forward biases if considered as the first pair, so the order does not really matter much.

Particulars of the cryptanalysis

First Stage: Corresponding to $(\mathcal{ID}_1, \mathcal{OD}_1)$, we set the threshold 0.565 and obtain 58 significant bits with 2^{20} samples. The set S_1 of significant bits are:

{18, 17, 16, 13, 12, 11, 10, 9, 6, 5, 57, 56, 55, 46, 45, 44, 43, 38, 37, 36, 82, 81, 80, 76, 75, 70, 69, 102, 101, 100, 171, 170, 169, 166, 165, 164, 163, 162, 222, 221, 220, 210, 209, 208, 207, 199, 198, 197, 196, 195, 249, 248, 247, 235, 234, 233, 229, 228}

Second Stage: For $(\mathcal{ID}_2, \mathcal{OD}_2)$, setting a threshold of 0.756, we obtain 77 significant bits with same number of samples as in the first stage. We realize that there are 21 common elements in the significant bits from S_1. After removing those key bits we have the set S_2 to be made up of 56 significant bits which are as follows:

{3, 4, 39, 40, 41, 42, 47, 48, 49, 50, 67, 68, 74, 77, 78, 86, 87, 88, 89, 106, 107, 108, 111, 112, 113, 114, 131, 132, 133, 136, 137, 138, 139, 150, 151, 152, 153, 192, 193, 194, 200, 201, 202, 203, 226, 227, 230, 231, 238, 239, 240, 241, 242, 252, 253, 254}

Now, $S_1 \cup S_2$ has 114 elements, hence the number of remaining key bits is 142.

Third Stage: Using $(\mathcal{ID}_3, \mathcal{OD}_3)$, a threshold of 0.92 is set to obtain 101 significant bits. We filtered 51 common elements among them and the rest 50 elements constructing S_3 are:

{2, 8, 14, 15, 34, 35, 66, 71, 72, 73, 79, 98, 99, 104, 105, 109, 110, 117, 118, 119, 120, 121, 128, 129, 130, 134, 135, 140, 141, 142, 143, 144, 145, 146, 154, 155, 156, 157, 158, 161, 167, 168, 181, 182, 183, 184, 185, 224, 225, 232,}

Therefore the total number of elements in $S_1 \bigcup S_2 \bigcup S_3$ is 164, leaving the number of PNBs to be 92. This reduction of PNBs over the existing techniques and dividing the significant bits in three separate sets help us to achieve the complexity less than $2^{k/2}$, when the key size is k-bits.

Attack Complexity: For the PNBs of $(\mathcal{ID}_1, \mathcal{OD}_1)$, we observe the backward bias $\epsilon_{a_1} = 0.015$. Taking $\Pr_{fa_1} = 2^{-68}$, we achieve $N_1 = 2^{41.47}$. For the PNBs corresponding to $(\mathcal{ID}_2, \mathcal{OD}_2)$, we have $\epsilon_{a_2} = 0.183$, choosing $\Pr_{fa_2} = 2^{-68}$ we have the required samples $N_2 = 2^{34.26}$. Further, for the PNBs corresponding to $(\mathcal{ID}_3, \mathcal{OD}_3)$, we have $\epsilon_{a_3} = 0.715$, and choosing $\Pr_{fa_3} = 2^{-68}$ we estimate the required samples as $N_3 = 2^{30.32}$.

Using Equation 4 the total complexity becomes

$$2^{58} \cdot 2^{41.47} + 2^{56} \cdot 2^{34.26} + 2^{50} \cdot 2^{30.32} + 2^{92} = 2^{99.47} + 2^{90.26} + 2^{80.32} + 2^{92} \approx 2^{99.48}$$

Table 2: Corrected complexities of certain previous key-recovery attacks on 6-round ChaCha256 and our improved result.

Attack	# PNB	Complexity	
		Claimed	Actual
[AFK+08]	147	2^{139}	2^{147}
[SZFW12]	136	2^{136}	2^{139}
[CM16]	159	$2^{131.40}$	2^{159}
[CM16]	161	$2^{129.53}$	2^{161}
[CM16]	166	$2^{127.5}$	2^{166}
[CN20]	210	$2^{102.2}$	2^{210}
[CN20]	212	$2^{104.68}$	2^{212}
[Our Work]	92	$2^{99.48}$	$2^{99.48}$

4 Implementing the attacks on a toy version of ChaCha

It is evident that the complexities we discussed so far are at a level that cannot immediately be implemented to demonstrate the complete attack. On the other hand, as in many other cryptanalytic efforts, there are several statistical assumptions while we estimate the complexity and success probability of the attack. In this direction let us explain the importance of developing ToyChaCha, a toy version of ChaCha. The following points discuss several aspects of proposing ToyChaCha and implementing the attacks.

- The differential attacks that has been proposed so far are based on the probabilistic neutral bits, and generally we follow the complexity formula proposed in [AFK+08]. Unfortunately, this formula has been used almost as a black box in several works afterwards, and substantial verification has not been studied on the accuracy of this complexity estimation. This formula is based on several assumptions and approximations. So far we do not have any scientific validation of this entire approach. Implementation of these cryptanalytic approaches would be convincing towards the validity of the entire attack procedure and the complexity calculation.

- The complexity formula is based on an approximation of binomial distribution to normal distribution. Thus how closely the approximation helps us to get the actual complexity can be experimentally validated by an application on a toy cipher.

- Note that, the authors of [AFK+08] claimed that the attack has success probability for at least half of the all possible keys. However, there has not been detailed investigation on the exact proportion of keys for which the attack is applicable. Implementing the attack on the toy version helps us to get a more accurate measure of the success probability, at least for the toy version.

- In the formula, the probability of false alarm plays a vital role. In the work of [AFK+08], the authors did not accurately estimate this probability of false alarm, rather considered an upper bound for this. This influences the derived complexity to deviate from the actual value. In the ToyChaCha we can experimentally measure the actual probability of false alarm error and investigate quantitatively how it may revise the complexity estimation.

4.1 Structure of ToyChaCha

The design of the ToyChaCha has similar structure with respect to the original cipher except for the constant vectors and the `quarterround` function. Here each entry (word) of the matrix is of 8 bits. We consider 32-bit key and replicate it on the next row. The four constants chosen are $c_0 = \texttt{0x65}, c_1 = \texttt{0x6e}, c_2 = \texttt{0x32}, c_3 = \texttt{0x74}$. The equation of the `quarterround` function which transforms a vector (a, b, c, d) to (a'', b'', c'', d'') via (a', b', c', d') follows as below:

$$\begin{aligned} a' &= a \boxplus b; & d' &= ((d \oplus a') \lll 4); \\ c' &= c \boxplus d'; & b' &= ((b \oplus c') \lll 3); \\ a'' &= a' \boxplus b'; & d'' &= ((d' \oplus a'') \lll 2); \\ c'' &= c' \boxplus d''; & b'' &= ((b' \oplus c'') \lll 1); \end{aligned} \tag{8}$$

4.2 Implementation of key recovery attack on 3.5 round using ideas from [AFK+08] and [Mai16]

On this ToyChaCha, we implement the fundamental key recovery attack given in [AFK+08] and the further improvement given by Maitra [Mai16] using chosen IV approach. After that, we implement our cryptanalytic technique using multiple $(\mathcal{ID}, \mathcal{OD})$ as well. The details of the machine where we experimented these are as follows: Intel(R) Xeon(R) W-2265 CPU @ 3.50GHz with Ubuntu 20.04.4 LTS operating system.

Approach of [AFK+08]

Using a single bit distinguisher on 2 rounds, we produce an attack in 3.5 round ToyChaCha. In this process, we use the $(\mathcal{ID}, \mathcal{OD})$ as $(\Delta_{13}^{(0)}[0], \Delta_1^{(2)}[6])$. From 3.5-th round we come back to 2nd round. To achieve the PNBs, we use the threshold 0.42 and achieve 16 PNBs which are listed below:

{7, 6, 5, 4, 3, 2, 1, 0, 14, 19, 18, 31, 30, 26, 25, 24}

The forward bias observed here is $\epsilon_d = 0.9167$ while the backward bias is $\epsilon_a = 0.377$. Therefore, for approximate calculations, $\hat{\epsilon} = \epsilon_d \cdot \epsilon_a = 0.343$, number of significant key bits $m = 16$. We achieve the best complexity for $\alpha = 11$. For this, $N = 378$, $T = 227$ and the complexity is $2^{24.67}$.

Implementation: In the implementation experiment, we execute the code with 2^{15} different keys. The average time required to recover the key is 0.9658 seconds, and the complexity is $2^{23.60}$, which is close to (slightly less than) the theoretically achieved value $2^{24.67}$. Out of these keys, 32705 keys were successfully recovered. Thus, the success probability of the attack is 99.81%. To estimate the false alarm probability, for each of the 2^{15} keys, we count the number of times the false alarm occurred and divided it by the number of guesses, and then took the average. We calculated the false alarm probability as low as 0.000341, which is less than the theoretical claimed upper bound 2^{-11}. The source code of the attack program is available in GitHub [Gar22] and the summary of the experimental evidences are provided in Table 3.

Approach of Maitra [Mai16]

In the chosen IV approach of Maitra [Mai16], during the pre-processing stage, for all possible keys in the input difference column, the attacker lists the IVs that produce minimum number of difference between X and X' after the first round. We call them "suitable IV" according to [Mai16]. This choice of IV improves the forward bias ϵ_d. In this

Table 3: Comparison of theoretical claim and experimental results of the implemented attack on 3.5 round ToyChaCha

Parameter	Attack of [AFK+08]		Attack of [Mai16]	
	Theory	Experiment	Theory	Experiment
Data	378	378	185	185
Complexity for significant bits	$2^{24.56}$	$2^{23.56}$	$2^{24.53}$	$2^{23.47}$
False alarm Complexity	2^{21}	$2^{18.18}$	2^{21}	$2^{17.59}$
Complexity for PNBs	2^{16}	$2^{15.01}$	2^{15}	$2^{13.99}$
Total Complexity	$2^{24.67}$	$2^{23.60}$	$2^{24.65}$	$2^{23.50}$
Success probability	$\geqslant 0.50$	0.9981	$\geqslant 0.50$	0.9971
$\Pr_{fa}$	$\leqslant 0.00049$	0.00034	$\leqslant 0.00049$	0.00015

approach the author did not include any key bit from the input difference column into the PNB set. In the actual attack, while guessing the key, the attacker uses the corresponding IVs from the prepared list. We implement this chosen IV approach in the ToyChaCha. We choose the input difference position $\Delta_{13}^{(0)}[0]$ and observe the output difference at $\Delta_1^{(2)}[6]$ after 2 rounds. The minimum number of differences between X and X' after the first round is 10. By assigning threshold 0.45, we achieve the following PNB set :

{ 7, 6, 5, 4, 3, 2, 1, 0, 19, 18, 31, 30, 26, 25, 24 }

Here, in the pre-processing stage, we prepare the list of key-IV pairs. In the input difference column, since 8 key bits are involved, there are 256 possible values. Out of them we observe that for 8 keys we do not get any IV which gives 10 differences after the first round. We call them *strong keys* according to the terminology used in [BLT20]. For each of the remaining 248 weak key values of k_1, we find out one IV value v_1 and prepare a list. So the list IV contains 248 key-IV pairs. In the program, after guessing a value of the significant key bits, we find the corresponding IV from the prepared list IV, and then implement the attack. In this approach, we observe $\epsilon_d = 0.98$ and $\epsilon_a = 0.49$. The PNB size is 15. So, based on the complexity formula, we get $N = 185, T = 119$ and the complexity is $2^{24.65}$ for $\alpha = 11$.

Implementation: In the implementation program, we run it over 2^{15} different keys. The average time required to recover the key is 0.866 seconds, and the complexity is $2^{23.50}$, which is slightly less than the theoretically achieved value of $2^{24.65}$. Out of 2^{15} keys, 32673 could be successfully recovered. So the success probability of the attack is 99.71%. To estimate the false alarm probability, for each of the 2^{15} keys, we count the number of times the false alarm occurred, divide it by the number of guesses, and then calculate the average. We estimate a false alarm probability of 0.00015, which is less than the theoretical claimed upper bound $2^{-11} = 0.000488$. The source code of the attack program is given in the GitHub link [Gar22] and the summary in Table 3.

Implementing Multiple $(\mathcal{ID}, \mathcal{OD})$ attack: Comparison with single $(\mathcal{ID}, \mathcal{OD})$

Finally, we present application of our technique on the 3-round ToyChaCha and confirm that our approach produces a more efficient cryptanalysis. Here, the distinguishers in

the second round are considered. We use the input difference at $\Delta_{13}^{(0)}[0]$ and the output difference in $\Delta_1^{(2)}[6]$, which produces a bias 0.91. In the backward part, we have to come back by one round only. There are 24 PNBs, each of which provides a backward bias $\epsilon_a = 1$.

As we discussed that a high number of PNBs can actually increase the complexity, in this approach one has to exploit 8 significant bits and 24 PNBs. According to the complexity formula given in [AFK+08], the complexity is $2^{14.56}$ (which is actually not correct) for $\alpha = 40$ and the data required is $94.7 \approx 95$. However, according to the modified and corrected complexity calculation formula in [DGSS22], the complexity is of the order of 2^{24}, as the 24 PNBs need to be exhaustively searched at the end. We implement this attack and achieved the complexity $2^{23.01}$. The details can be found in Table 4.

Thus, now we explain the attack using three $(\mathcal{ID}, \mathcal{OD})$ pairs $(\mathcal{ID}_1, \mathcal{OD}_1)$: $(\Delta_{13}^{(0)}[0], \Delta_1^{(2)}[6])$, $(\mathcal{ID}_2, \mathcal{OD}_2)$: $(\Delta_{14}^{(0)}[0], \Delta_2^{(2)}[6])$, $(\mathcal{ID}_3, \mathcal{OD}_3)$: $(\Delta_{15}^{(0)}[0], \Delta_3^{(2)}[6])$. For each pair, the forward bias $\epsilon_d = 0.91$. Also, for each of the above, the significant key bits are basically all the key bits which lies in the same column as the output difference bit, and the remaining are PNBs. In each stage, the observed backward bias is $\epsilon_a = 1$. Since we use 3 pairs, $Pr_{nd}^* = (1.3 \times 10^{-3})/3 \approx 0.43 \times 10^{-3}$. Thus, the formula for N_i for each of $i = 1, 2, 3$ is:

$$N \approx \left(\frac{\sqrt{\alpha \log 4} + 3.4\sqrt{1 - \epsilon_a^2 \epsilon_d{}^2}}{\epsilon_a \epsilon_d} \right)^2 .$$

For $(\mathcal{ID}_1, \mathcal{OD}_1)$, we get 8 significant key bits $\{15, 14, 13, 12, 11, 10, 9, 8\}$. For $(\mathcal{ID}_2, \mathcal{OD}_2)$, we obtain another set of 8 significant key bits *viz.*, $\{23, 22, 21, 20, 19, 18, 17, 16\}$, and since there is no common element among themselves, hence total number of significant bits become 16. With $(\mathcal{ID}_3, \mathcal{OD}_3)$ we get 8 more distinct significant key bits $\{31, 30, 29, 28, 27, 26, 25, 24\}$. Keeping the false alarm error $\alpha_i = 40$ for each i, we obtain $N_i = 2^{6.56}$. Therefore the overall attack complexity is $2^8 \cdot 2^{6.56} + 2^8 \cdot 2^{6.56} + 2^8 \cdot 2^{6.56} + 2^8 = 2^{16.15}$.

Implementation: We execute the program for 2^{15} different keys. The average time required to recover the key is 10.066 ms and the complexity is $2^{13.67}$. The details of the complexity for each stage is given in Table 4. We also compare this attack with the single $(\mathcal{ID}, \mathcal{OD})$ based effort in the same table.

Discussion: As we can see in Table 4, it is validated that for single $(\mathcal{ID}, \mathcal{OD})$, the complexity formula proposed in [AFK+08] does not provide correct complexity when the number of PNBs is large. On the other hand, the correctness of the modified complexity formula by [DGSS22] is validated too. Secondly, the multiple $(\mathcal{ID}, \mathcal{OD})$ attack approach proposed in our work and its complexity formula are properly verified through the experimental result. It is clear that our method reduces the attack complexity significantly than that of the existing single $(\mathcal{ID}, \mathcal{OD})$ strategy.

5 Success probability estimation for the attacks

Here we propose a theoretical approach to achieve a better estimation of success probability corresponding to the PNB-based differential cryptanalysis than what has been claimed in [AFK+08]. It was claimed in [AFK+08] that the success probability is at least 50%. So far, there has not been any disciplined investigation in this regard to estimate the success probability in a more accurate manner. Later, in Maitra's [Mai16] approach using the chosen IVs, it was claimed that the right IVs are available for 70% of the keys only.

Table 4: Comparison of theory and experiments for 3-round attack using multiple $(\mathcal{ID}, \mathcal{OD})$ and single $(\mathcal{ID}, \mathcal{OD})$

Complexity	Single $(\mathcal{ID}, \mathcal{OD})$			Multiple $(\mathcal{ID}, \mathcal{OD})$	
	Theory [AFK+08]	Theory [DGSS22]	Experiment	Theory	Experiment
Data	95	95	95	94	94
Recover S_1	$2^{14.56}$	$2^{14.56}$	$2^{13.51}$	$2^{14.56}$	$2^{13.51}$
Recover S_2	-	-	-	$2^{14.56}$	$2^{13.51}$
Recover S_3	-	-	-	$2^{14.56}$	$2^{13.5}$
False alarm	2^{-8}	2^{-8}	0	0	0
Recover PNB	0	2^{24}	$2^{23.01}$	2^{8}	$2^{6.95}$
Total	$2^{14.56}$	2^{24}	$2^{23.01}$	$2^{16.15}$	$2^{15.1}$

So, he computed the median bias over those 70% keys only and then used the same approach as in [AFK+08] to compute the complexity. Thus, the success probability of the chosen IV approach can be claimed to be at least 35%. There has been no analysis how effective the attack is for the rest 65% keys. Therefore, this is an important area of further investigation to obtain a more accurate range for the success probability. Interestingly, our implementation on the toy cipher shows that both Aumasson's attack and Maitra's attack have success probability more than 99% on the toy version. This identifies that the attacks are far more effective than what was initially assumed. Thus, we aim to obtain a better measure of success probability.

Theorem 1. *For each $i \in \{0, 1, \ldots, n\}$, let $\mathcal{X}_i$ denote the normal random variable with mean $\frac{N}{2}(1+\epsilon_i)$ and standard deviation $\sqrt{\frac{N}{4}(1-\epsilon_i^2)}$ $(0 \le \epsilon_i \le \epsilon_{max})$. Let $\mathcal{Y}$ be a random variable such that $\mathcal{Y} = X_i$ for each i with probability $\frac{1}{n}$. Let $\rho_0, \rho_1, \ldots, \rho_k$ be such that $0 = \rho_0 < \rho_1 < \cdots < \rho_{k-1} < \rho_k = \epsilon_{max}$ and, for each of $j = 0$ to $k-1$, $\mathbb{X}'_j$ be the set of all $\mathcal{X}_i$ such that $\rho_j < \epsilon_i < \rho_{j+1}$ for $1 \le j \le k-1$. Consider E_j be the event that $\mathcal{Y}$ chooses a $\mathcal{X}_i$ from $\mathbb{X}_j$. Then,*

$$\sum_{j=0}^{k-1} \Phi\left(\frac{T - \frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}\right) \cdot \Pr(E_j) < \Pr(\mathcal{Y} < T) < \sum_{j=0}^{k-1} \Phi\left(\frac{T - \frac{N}{2}(1+\rho_j)}{\sqrt{\frac{N}{4}(1-\rho_j^2)}}\right) \cdot \Pr(E_j). \tag{9}$$

Proof. $\Pr(\mathcal{Y} < T) = \sum_{j=0}^{k-1} \Pr((\mathcal{Y} < T) \cap E_j) = \sum_{j=0}^{k-1} \Pr((\mathcal{Y} < T) | E_j) \cdot \Pr(E_j)$.

First we will find a lower and an upper bound for each $\Pr((\mathcal{Y} < T)|E_j)$. Let Φ be the Cumulative Distribution Function of standard normal distribution. Then we know that,

$$\Pr\left(X_i \le T\right) = \Pr\left(\frac{X_i - \frac{N}{2}\left(1+\epsilon_i\right)}{\sqrt{\frac{N}{4}\left(1-\epsilon_i^2\right)}} \le \frac{T - \frac{N}{2}\left(1+\epsilon_i\right)}{\sqrt{\frac{N}{4}\left(1-\epsilon_i^2\right)}}\right) = \Phi\left(\frac{T - \frac{N}{2}\left(1+\epsilon_i\right)}{\sqrt{\frac{N}{4}\left(1-\epsilon_i^2\right)}}\right).$$

For any $X_i \in \mathbb{X}_j$,

$$\frac{1}{\sqrt{1-\rho_j^2}} < \frac{1}{\sqrt{1-\epsilon_i^2}} < \frac{1}{\sqrt{1-\rho_{j+1}^2}}, (\text{since } \rho_j < \epsilon_i < \rho_{j+1}). \tag{10}$$

Now, if $T - \frac{N}{2}(1+\epsilon_i) < 0$, then

$$\frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\epsilon_i^2)}} > \frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}} > \frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}$$

On the other hand, if $T - \frac{N}{2}(1+\epsilon_i) \geq 0$,

$$\frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\epsilon_i^2)}} \geq 0 > \frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}$$

Therefore for any ϵ_i such that $\epsilon_i < \rho_{j+1}$, $\frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\epsilon_i^2)}} > \frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}$. Since Φ is an increasing function, for any $X_i \in \mathbb{X}_j$ such that $j \in \{0, 1, \ldots, k-2\}$,

$$\Pr(X_i \leq T) = \Phi\left(\frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\epsilon_i^2)}}\right) > \Phi\left(\frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}\right)$$

$$\implies \Pr(\mathcal{Y} < T | E_j) > \Phi\left(\frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}\right).$$

Therefore, $\Pr(\mathcal{Y} < T) = \sum_{j=0}^{k-1} \Pr(\mathcal{Y} < T | E_j) \cdot \Pr(E_j) > \sum_{j=0}^{k-1} \Phi\left(\frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}\right) \cdot \Pr(E_j)$.

In a similar manner for any ϵ_i such that $\epsilon_i > \rho_j$,

$$\frac{T-\frac{N}{2}(1+\epsilon_i)}{\sqrt{\frac{N}{4}(1-\epsilon_i^2))}} < \frac{T-\frac{N}{2}(1+\rho_j)}{\sqrt{\frac{N}{4}(1-\rho_j^2)}}$$

Therefore $\Pr(\mathcal{Y} < T) = \sum_{j=0}^{k-1} \Pr(\mathcal{Y} < T | E_j) \cdot \Pr(E_j) < \sum_{j=0}^{k-1} \Phi\left(\frac{T-\frac{N}{2}(1+\rho_j)}{\sqrt{\frac{N}{4}(1-\rho_j^2)}}\right) \cdot \Pr(E_j)$. □

Exploiting Theorem 1 to obtain the range for success probability: We use the above theorem to measure the success probability corresponding to the attacks. For any key, the observed bias at the $\mathcal{OD}$ bit $\Delta_p^{(r)}[q]$, $\Gamma_p[q] = 0$ is ϵ. We collect the output key stream for N different IVs. So in this theorem, each $\mathcal{X}_i$ can be considered to be the count of $\Gamma_p[q] = 0$ out of the N samples corresponding to a key, say k_i. Therefore the distribution is approximated by normal with mean $\frac{N}{2}(1+\epsilon_i)$ and standard deviation $\sqrt{\frac{N}{4}(1-\epsilon_i^2)}$. Now, $\Pr(\mathcal{X}_i < T)$ is the probability that even for the correct guess of significant key bits of k_i, it is not detected. Further, $\Pr(Y < T)$ represents that for a randomly chosen key, the key is not detected in the attack even after the guess for significant bits are correct. Therefore, $1 - \Pr(Y < T)$ represents the success probability of the attack. So, $1 - \sum_{j=0}^{k-1} \Phi\left(\frac{T-\frac{N}{2}(1+\rho_j)}{\sqrt{\frac{N}{4}(1-\rho_j^2)}}\right) \cdot \Pr(E_j)$ is a lower bound and $1 - \sum_{j=0}^{k-1} \Phi\left(\frac{T-\frac{N}{2}(1+\rho_{j+1})}{\sqrt{\frac{N}{4}(1-\rho_{j+1}^2)}}\right) \cdot \Pr(E_j)$ is an upper bound for the success

probability. In the attack, the bias ϵ is approximated by $\epsilon_d \cdot \epsilon_a$. Since $\epsilon_d \leq 1$, the maximum value of ϵ, i.e., ϵ_{max} is ϵ_a. Instead of dealing with ϵ, we deal with ϵ_d, since its value is high. We choose $0 = \rho'_0 < \rho'_1 \cdots < \rho'_k = 1$ and define $\rho_i = \rho'_i \cdot \epsilon_a$. Then, ρ_i's will follow the property mentioned in the theorem. Moreover, for some $\epsilon_i = \epsilon_{d_i}.\epsilon_a$, $\rho_j < \epsilon_i < \rho_{j+1}$ implies $\rho'_j < \epsilon_{d_i} < \rho'_{j+1}$. Therefore, the lower and upper bounds are respectively

$$1 - \sum_{j=0}^{k-1} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_j \cdot \epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_j \cdot \epsilon_a)^2)}}\right) \cdot \Pr(E_j), \quad 1 - \sum_{j=0}^{k-1} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_{j+1} \cdot \epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_{j+1} \cdot \epsilon_a)^2)}}\right) \cdot \Pr(E_j).$$

5.1 Calculation on ToyChaCha and experimental verification

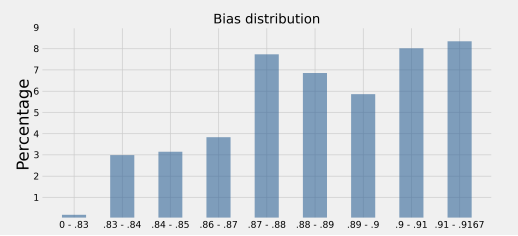

Figure 1: Bias distribution in ToyChaCha for $(\mathcal{ID}, \mathcal{OD})$ pair $(\Delta_{13}^{(0)}[0] - \Delta_1^{(2)}[6])$ after 2 rounds in [AFK+08].

We apply Theorem 1 to estimate a range for the success probability of the attack based on the approach of [AFK+08] as given in subsection 4.2. Further, we verify this comparing with the experimental result. For convenience, let us call each ρ_j a marker.

Result 1. *Success probability of the attack proposed using Aumasson's approach [AFK+08] on ToyChaCha is in the range* $[0.996246, 0.998663]$.

Proof. We consider $k + 1 = 7$ markers. We choose $\rho'_0 = 0.0, \rho'_1 = 0.79, \rho'_2 = 0.83, \rho'_3 = 0.87, \rho'_4 = 0.91, \rho'_5 = 0.95, \rho'_6 = 1.0$. We find the probabilities of E_i's experimentally, which is as follows: $\Pr(E_0) = 0.0, \Pr(E_1) = 0.00157, \Pr(E_2) = 0.174, \Pr(E_3) = 0.292, \Pr(E_4) = 0.281, \Pr(E_5) = 0.252$. Thus, we obtain the following.

$$\text{Lower bound: } 1 - \sum_{j=0}^{5} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_j.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_j.\epsilon_a)^2)}}\right) \cdot \Pr(E_j) = 1 - 0.003753660 \approx 0.996246.$$

$$\text{Upper bound: } 1 - \sum_{j=0}^{5} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_{j+1}.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_{j+1}.\epsilon_a)^2)}}\right) \cdot \Pr(E_j). = 1 - 0.00133734 \approx 0.998663.$$

□

Referring to Table 3, one can verify that the success probability is 0.9983, which validates our theory.

5.2 Success probability of attack [AFK+08] against ChaCha256

In the attack produced by [AFK+08] against ChaCha256 for 7 rounds, the input difference was given at the position $\Delta_{13}^{(0)}[13]$ and the output difference was observed at $\Delta_{11}^{(3)}[0]$ after 3 rounds. The median forward bias is 0.026 and the backward one is 0.00059.

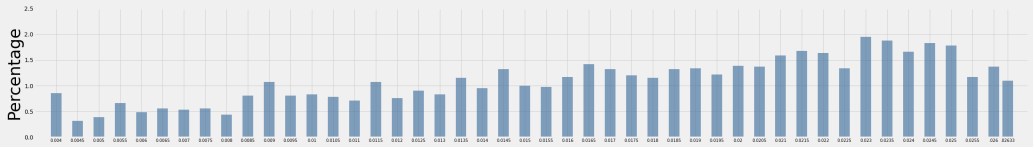

Figure 2: Bias Distribution for approach from [AFK+08] for $\mathcal{ID}$ $\Delta_{13}^{(0)}[13]$, $\mathcal{OD}$ $\Delta_{11}^{(3)}[0]$.

Table 5: ρ'_j values and corresponding $\Pr(E_j)$ for the bias distribution using Aumasson's approach for $\mathcal{ID}$ $\Delta_{13}^{(0)}[13]$ and $\mathcal{OD}$ $\Delta_{11}^{(3)}[0]$ after 3 rounds

j	0	1	2	3	4	5	6	7	8	9	10	11
ρ'_j	0.0	0.008	0.010	0.012	0.014	0.016	0.018	0.020	0.022	0.024	0.026	0.028
$\Pr(E_j)$	0.056	0.030	0.034	0.037	0.040	0.045	0.054	0.063	0.068	0.066	0.063	0.442

Now, we find the forward bias ϵ_d for 2^{15} randomly chosen key, and observe that the bias values are distributed in a wide range. For some keys, the bias is sometimes even as low as 0.001. Refer to Figure 2, that provides a spectral representation of the forward bias observed in ChaCha256, for different keys whose values are in the range $[0, 0.026]$. We divided the entire range into 46 parts, each of length 0.005. For each sub-range, the bar represents the percentage of keys which produces a forward bias in that range.

We apply Theorem 1 to find a lower and upper bound of the success probability of the differential attack proposed in [AFK+08] against ChaCha256. Here, the 7-round ChaCha was cryptanalysed using a distinguisher in the third round. The input difference was given in the position $\Delta_{13}^{(0)}[13]$ and the output difference was observed at $\Delta_{11}^{(3)}[0]$. The median of the forward bias ϵ_d was 0.026 and the backward bias was 0.023. The data complexity was $N = 2^{27}$. For accuracy, we use $N = 133330148 \approx 2^{26.99}$. Using the formula for the threshold T, we obtain $T \approx 66687619$.

Result 2. *In the differential attack using random IVs with $\mathcal{ID}$ position $\Delta_{13}^{(0)}[13]$ and $\mathcal{OD}$ position $\Delta_{11}^{(3)}[0]$ after 3 rounds, with data complexity $N = 133330148$ and threshold $T = 66687619$, the success probability is in the range $[0.799311, 0.842324]$.*

Proof. We use total 12 markers $\rho'_0, \rho'_1, \ldots, \rho'_{11}$. The values and the corresponding probabilities of E_j are given in Table 5.

$$\text{Lower Bound: } 1 - \sum_{j=0}^{11} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_j.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_j.\epsilon_a)^2)}}\right) \cdot \Pr(E_j) = 1 - 0.200689 = 0.799311$$

$$\text{Upper Bound: } 1 - \sum_{j=0}^{11} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_{j+1}.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_{j+1}.\epsilon_a)^2)}}\right) \cdot \Pr(E_j) = 1 - 0.157676 = 0.842324$$

□

5.3 Chosen IV

In the chosen IV approach, Maitra [Mai16] used the same $(\mathcal{ID}, \mathcal{OD})$ pair. Because of the chosen IVs, the median of forward biases increased to 0.14 and the backward bias is 0.015862. The data complexity is $N = 15430828$ and threshold $T = 7726261$.

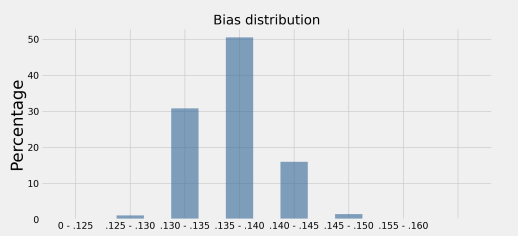

Figure 3: Bias Distribution for Chosen IV approach for $\mathcal{ID}$ $\Delta_{13}^{(0)}[13]$ and $\mathcal{OD}$ $\Delta_{11}^{(3)}[0]$

We obtain the forward bias ϵ_d for 2^{15} randomly chosen key. In Figure 3, a spectral representation of the forward bias is observed for ChaCha256, for different keys. The bias values are primarily distributed in the range $[0.125 - 0.150]$. We divided the entire range into sub-ranges of length 0.005 each.

Table 6: ρ'_j values and corresponding $\Pr(E_j)$ for the bias distribution using Chosen IV approach for $\mathcal{ID}$ $\Delta_{13}^{(0)}[13]$ and $\mathcal{OD}$ $\Delta_{11}^{(3)}[0]$ after 3 rounds

j	0	1	2	3	4	5
ρ'_j	0	0.125	0.130	0.135	0.140	0.145
$\Pr(E_j)$	0.000008	0.03115	0.308	0.505	0.16	0.015

Result 3. *In the differential attack using chosen IV approach with $\mathcal{ID}$ position $\Delta_{13}^{(0)}[13]$ and $\mathcal{OD}$ position $\Delta_{11}^{(3)}[0]$ after 3 rounds, for $N = 15430828$ and threshold $T = 7726261$, the success probability is in the range $[0.997243, 0.998972]$.*

Proof. To obtain the success probability, we use $k+1 = 7$ markers, $0, 0.125, 0.13, 0.135, 0.14, 0.145, 1$. The probabilities $\Pr(E_j)$ for each range is given in Table 6.

$$\text{Lower Bound: } 1 - \sum_{j=0}^{5} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_j.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_j.\epsilon_a)^2)}}\right) \cdot \Pr(E_j) = 1 - 0.002757 = 0.997243$$

$$\text{Upper Bound: } 1 - \sum_{j=0}^{5} \Phi\left(\frac{T - \frac{N}{2}(1 + \rho'_{j+1}.\epsilon_a)}{\sqrt{\frac{N}{4}(1 - (\rho'_{j+1}.\epsilon_a)^2)}}\right) \cdot \Pr(E_j) = 1 - 0.001028 = 0.998972$$

Table 7: More accurate success probabilities of the attacks on 7-round ChaCha256.

IV type	$\mathcal{ID}$	$\mathcal{OD}$	N	T	Success prob.
Random IV [AFK+08]	$\Delta_{13}^{(0)}[13]$	$\Delta_{11}^{(3)}[0]$	133330148	66687619	[0.799311, 0.842324]
Chosen IV [Mai16]	$\Delta_{13}^{(0)}[13]$	$\Delta_{11}^{(3)}[0]$	15430828	7726261	[0.997243, 0.998972]

□

6 Conclusion

This work first shows the limitation of the existing attack approaches using a single $\mathcal{ID}, \mathcal{OD}$ pair against ChaCha, when the number of PNBs is high. Apart from improving the attack

with multiple pairs significantly, our idea opens a new direction of further work exploiting a divide-and-conquer approach with several sets. If distinguishers in higher rounds can be discovered in future, this strategy can significantly reduce the attack complexity for 7 or higher rounds. A toy model of ChaCha is proposed as well for more detailed investigations to compare the efficacy of the existing and the new attacks. This helps to build a clearer understanding of the cryptanalytic techniques as the complete attack can be implemented with a reasonable complexity. Finally, we exploit statistical techniques to estimate the success probabilities of different cryptanalytic approaches against ChaCha and validate our idea on the toy version.

7 Acknowledgement

We are very grateful to the reviewers for their valuable comments and suggestions that greatly improved the presentation and quality of this paper. The second author acknowledges the financial support provided by the Council of Scientific & Industrial Research (CSIR) through Junior Research Fellowship (file no. 09/1026(0031)/2020-EMR-I) to carry out the research work.

References

[ABP+13] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt. On the Security of RC4 in TLS. *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA*, pages 305–320, 2013. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan.

[AFK+08] J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. *Fast Software Encryption, 15th International Workshop, Lausanne, Switzerland, Revised Selected Papers*, 5086:470–488, 2008. https://doi.org/10.1007/978-3-540-71039-4_30.

[Ber08a] D. J. Bernstein. ChaCha, a variant of Salsa20. *Workshop record of SASC*, 8:3–5, 2008. https://cr.yp.to/chacha/chacha-20080128.pdf.

[Ber08b] D. J. Bernstein. Which phase-3 estream ciphers provide the best software speeds? 2008. http://www.ecrypt.eu.org/stream.

[BLT20] C. Beierle, G. Leander, and Y. Todo. Improved Differential-Linear Attacks with Applications to ARX Ciphers. *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, Santa Barbara, CA, USA, Proceedings, Part III*, 12172:329–358, 2020. https://doi.org/10.1007/978-3-030-56877-1_12.

[CM16] A. R. Choudhuri and S. Maitra. Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. *IACR Trans. Symmetric Cryptol.*, 2016(2):261–287, 2016. https://doi.org/10.13154/tosc.v2016.i2.261-287.

[CN20] M. Coutinho and T. C. S. Neto. New Multi-bit Differentials to Improve Attacks Against ChaCha. *IACR Cryptol. ePrint Arch.*, page 350, 2020. https://eprint.iacr.org/2020/350.

[CN21] M. Coutinho and T. C. S. Neto. Improved Linear Approximations to ARX Ciphers and Attacks Against ChaCha. *Advances in Cryptology - EUROCRYPT*

2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, Proceedings, Part I, 12696:711–740, 2021. https://doi.org/10.1007/978-3-030-77870-5_25.

[DDSM22] S. Dey, C. Dey, S. Sarkar, and W. Meier. Revisiting Cryptanalysis on ChaCha From Crypto 2020 and Eurocrypt 2021. *IEEE Trans. Inf. Theory*, 68(9):6114–6133, 2022. https://doi.org/10.1109/TIT.2022.3171865.

[DGSS22] S. Dey, H. K. Garai, S. Sarkar, and N. K. Sharma. Revamped Differential-Linear Cryptanalysis on Reduced Round ChaCha. *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, Proceedings, Part III*, 13277:86–114, 2022. https://doi.org/10.1007/978-3-031-07082-2_4.

[DS17] S. Dey and S. Sarkar. Improved analysis for reduced round Salsa and Chacha. *Discret. Appl. Math.*, 227:58–69, 2017. https://doi.org/10.1016/j.dam.2017.04.034.

[Gar22] H. K. Garai. ToyChaCha. *GitHub repository*, 2022. https://github.com/reiner1757/ToyChaCha.git.

[GMPS14] S. Sen Gupta, S. Maitra, G. Paul, and S. Sarkar. (non-)Random Sequences from (Non-)Random Permutations - Analysis of RC4 Stream Cipher. *J. Cryptol.*, 27(1):67–108, 2014. https://doi.org/10.1007/s00145-012-9138-1.

[Mai16] S. Maitra. Chosen IV cryptanalysis on reduced round ChaCha and Salsa. *Discret. Appl. Math.*, 208:88–97, 2016. https://doi.org/10.1016/j.dam.2016.02.020.

[SG18] L. Song and J. Guo. Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP. *IACR Trans. Symmetric Cryptol.*, 2018(3):182–214, 2018. https://doi.org/10.13154/tosc.v2018.i3.182-214.

[SGSL18] L. Song, J. Guo, D. Shi, and S. Ling. New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions. *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, Proceedings, Part II*, 11273:65–95, 2018. https://doi.org/10.1007/978-3-030-03329-3_3.

[SZFW12] Z. Shi, B. Zhang, D. Feng, and W. Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. *Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, Revised Selected Papers*, 7839:337–351, 2012. https://doi.org/10.1007/978-3-642-37682-5_24.