Differential Cryptanalysis of Salsa and ChaCha – An Evaluation with a Hybrid Model

Arka Rai Choudhuri and Subhamoy Maitra

Indian Statistical Institute, Kolkata, India
arkarai.choudhuri@gmail.com, subho@isical.ac.in

Abstract. While Salsa and ChaCha are well known software oriented stream ciphers, since the work of Aumasson et al in FSE 2008 there aren't many significant results against them. The basic model of their attack was to introduce differences in the IV bits, obtain biases after a few forward rounds, as well as to look at the Probabilistic Neutral Bits (PNBs) while reverting back. In this paper we first consider the biases in the forward rounds, and estimate an upper bound on the number of rounds till such biases can be observed. For this, we propose a hybrid model (under certain assumptions), where initially the nonlinear rounds as proposed by the designer are considered, and then we employ their linearized counterpart. The effect of reverting the rounds with the idea of PNBs is also considered. Based on the assumptions and analysis, we conclude that 12 rounds of Salsa and ChaCha should be considered sufficient for 256-bit keys under the current best known attack models.

Keywords: ARX Cipher, Stream Cipher, ChaCha, Salsa, Non-Randomness, Probabilistic Neutral Bit (PNB).

1 Introduction

Salsa and ChaCha are stream ciphers designed by Dan Bernstein, with their main design criteria are as follows. There is some nonlinear 1-1 round function $F : \{0,1\}^{512} \rightarrow \{0,1\}^{512}$. Given a 512-bit input x (understood as 16 words of 32-bit each), the cipher calculates $F^R(x)$, i.e., R rounds of the function F. It then adds x to $F^R(x)$ in the corresponding words to produce the final output. In simple notations, $x + F^R(x)$ is the 512-bit key-stream.

As for the structure of the input, x consists of a 256-bit secret key[1] (8 words), 128-bit IV (4 words) and a 128-bit constant (4 words). Unlike the traditional designs of stream ciphers, Salsa and ChaCha do not have different key-scheduling and pseudo-random generator phases. Instead, these are motivated from the concept of Pseudo Random Functions (PRFs) in block cipher paradigm.

Salsa20 [4] was designed in 2005 as a candidate for eStream [9], and its variant Salsa20/12 was finally accepted in the software portfolio. The ChaCha [5] stream cipher was proposed in early 2008 to provide improved diffusion in comparison

[1] For 128-bit key the key-bits are repeated twice.

to Salsa. This cipher recently attracted attention due to its deployment in several applications by Google [19].

Contribution. The security claims of commercial stream ciphers are mostly conjectures, primarily based on existing attacks. In order to build confidence about the designs, simple but effective tools may be considered for analysis. The possible standardization of ChaCha makes it all the more relevant for a more comprehensive study of the cipher. Because of only a conjectured security, design of the ciphers are generally defensive, and hence speed is compromised to avoid any potential pitfall. This is emphasized in [3] where Bernstein remarks, "I'm comfortable with the 20 rounds of Salsa20 as being far beyond what I'm able to break. Perhaps it will turn out that, after more extensive attempts at cryptanalysis, the community is comfortable with a smaller number of rounds; I can imagine using a smaller number of rounds for the sake of speed".

We introduce the simple idea of a **hybrid model** for the evaluation of the differential cryptanalysis of Salsa and ChaCha where the initial rounds are run with the original non-linear function, and subsequent rounds are with the linearized counterpart. This idea stems from the ease of analysing a linear structure rather than the non-linearities arising from an ARX construction. We show, for any function f consisting of only modular additions and XORs (as is the case in ARX constructions), we can upper bound the absolute value of its biases with the absolute value of the bias of its linear approximation. Next, we perform extensive calculations for bias propagation in the forward direction for both Salsa and ChaCha. Using these, we derive bias bounds at the end of the non-linear rounds in order to have a secure PRF in the forward direction for a desired number of rounds.

Lastly, combining both the forward and backward biases, under certain assumptions, we claim that only 12 rounds for both Salsa and ChaCha are sufficient to provide security against certain kinds of differential cryptanalysis for 256-bit keys.

Related Work. While to the best of our knowledge there aren't methods similar to our proposed hybrid model for stream cipher cryptanalysis, there are several works that have studied the cryptanalysis of Salsa and ChaCha [8, 10, 21, 1, 11, 20, 18, 15, 14, 6]. The basic ideas in these works, which we take to be our reference attack model, consider the following strategy: (1) Apply certain input differences at the initial state to study significant biases at some output. (2) If it is possible to proceed a few rounds forward as above, one may try to revert back a few rounds from a final state to obtain further non-randomness.

Significant development has been achieved in the area of ARX toolkits[12]. While there have been applications of these toolkits towards some ARX based ciphers to build complex differential characteristics [13], no significant success has been reported yet in its application to Salsa and ChaCha.

2 Description of Salsa and Some Notations

In both Salsa and ChaCha, the cipher state is of 16 words, where each word is of 32 bits, and these words can be represented as a 4×4 matrix. Let us first describe Salsa. We have the following state matrix

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix},$$

The matrix on the right shows the initial configuration of the state that takes four predefined constants[2] (totalling to 128 bits), 256-bit key $k_0, \ldots, k_7$, 64-bit nonce v_0, v_1 and 64-bit counter t_0, t_1. For the 128-bit version of Salsa, the key words are repeated twice and the constant values differ slightly. In this paper, we will focus on the 256-bit key version. Further, we will refer to the nonce and counter words together as IV words.

For Salsa, a quarterround on (a, b, c, d) to update its values is defined as follows:

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

Each round consists of two stages, the first applies quarterround to all the four columns in the following order: quarterround(x_0, x_4, x_8, x_{12}), quarterround(x_5, x_9, x_{13}, x_1), quarterround$(x_{10}, x_{14}, x_2, x_6)$, and quarterround$(x_{15}, x_3, x_7, x_{11})$, and then the second stage consisting of a transpose(X) as:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \rightarrow X^T = \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}.$$

By $X^{(R)}$, we mean that R such rounds (each of four quarterrounds and a transpose) have been applied to the initial state $X^{(0)}$. A keystream block of 16 words or 512 bits is obtained as $Z = X + X^{(R)}$, where the addition is of the corresponding 32-bit words modulo 2^{32}. While for Salsa20, $R = 20$, the accepted cipher in eStream [9] software portfolio is Salsa20/12, where $R = 12$.

Each Salsa20 round is reversible as the state-transition operations are reversible, i.e., if $X^{(r+1)} = \mathsf{round}(X^{(r)})$, then $X^{(r)} = \mathsf{reverseround}(X^{(r+1)})$, where reverseround is the inverse of round and consists of first transposing the state and then applying the inverse of quarterround to each column as follows:

$$\left.\begin{aligned} a &= a \oplus ((d + c) \lll 18), \\ d &= d \oplus ((c + b) \lll 13), \\ c &= c \oplus ((b + a) \lll 9), \\ b &= b \oplus ((a + d) \lll 7). \end{aligned}\right\} \tag{2}$$

[2] $c_0 =$ 0x61707865, $c_1 =$ 0x3320646e, $c_2 =$ 0x79622d32, $c_3 =$ 0x6b206574

Consider that one obtains a state $X^{(1)}$ after one round of Salsa (or ChaCha). To know whether it is a valid state after one round, one needs to come back by one reverse round and then verify whether the constants in the first row are indeed the specified ones.

For the description of ChaCha we refer the reader to Appendix A.

Notations. Here x_i is the i^{th} word of the matrix X. Further, by $x_{i,j}$, we mean the j^{th} bit of x_i, where the 0^{th} bit is the least significant bit.

Given two states $X^{(r)}, X'^{(r)}$, we denote the differential of individual words by $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$. Extending to bits, by $\Delta_{i,j}^{(r)} = x_{i,j}^{(r)} \oplus x_{i,j}'^{(r)}$, we mean the difference between two states at the j^{th} bit of the i^{th} word after r rounds. For example, '$\Delta_{13,5}^{(0)} = 1$' means that we have two initial states $X^{(0)}, X'^{(0)}$ that differ at the 5^{th} bit of the 13^{th} word. Unless otherwise specified, the differentials at all bit positions are defined to be 0.

From the perspective of cryptanalysis, we are interested in introducing a difference at the initial state (call it Input Differential or $\mathcal{ID}$) and then attempt to obtain certain biases corresponding to combinations of some output bits (call it Output Differential or $\mathcal{OD}$). In this direction, one can compute

$$\Pr(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \varepsilon_d),$$

where the probability is estimated for a fixed key and all possible choices of nonces and counter words, other than the constraints imposed due to the input differences. Here, the bias is denoted by ε_d, and we will concern ourselves only with the absolute value of the bias, $|\varepsilon_d|$.

In fact, one can consider a more general scenario as

$$\Pr((\bigoplus_u \Delta_{p_u,q_u}^{(r)}) = 1 | \Delta_{i_0,j_0}^{(0)} = 1, \Delta_{i_1,j_1}^{(0)} = 1, \ldots]) = \frac{1}{2}(1 + \varepsilon_d),$$

where one may observe the biases at certain linear combination of output differences given the input differences at one or more than one position. We will be using **differential-linear** biases.

3 The Hybrid Model to Estimate the Biases in the Forward Direction

In this section, we present some ideas and propose a model to provide an upper bound to the biases in the forward direction given any differential. Our main assumptions, which we will try to substantiate later, are as follows:

- Given any differential at the IV bits, after a few rounds, $|\varepsilon_d| \leq 1 - \delta$, for some pre-defined constant $\delta > 0$.

– Consider that the cipher is run for m rounds as designed. We then run the subsequent rounds of the cipher in two different ways: (1) As originally designed; (2) We consider linear versions of the round functions.
  We prove that the absolute value of the bias in the second case will be greater than the first.
– We assume the evolution of the state will be quite complicated after the initial few rounds, and hence can consider the bits to be independent, to allow for the application of the piling-up lemma.

It is conjectured that the random functions $(v, c) \mapsto \mathsf{Salsa20/R}(v, c)$ and $(v, c) \mapsto \mathsf{ChaChaR}(v, c)$ are indistinguishable from uniform [2]. In our attack model, we would like to distinguish the keystream generated by the ciphers, by varying the inputs to these random function to obtain a PRG, from a completely random string of bits.

It has been well documented for Salsa and ChaCha, for low values of R, the corresponding PRGs are not secure. These are a consequence of high biases obtained by experiments[1, 14, 15, 8, 10, 11, 20, 21]. But there exists no security proof as to how many rounds are required to ensure the security of the underlying PRF. Hence, designers take the safer option of specifying a very high number of rounds to avoid any potential pitfalls. While it might ensure safety, and make cryptanalysis of the cipher potentially harder, it is often more than necessary and affects the performance of the cipher in terms of speed. This was addressed by Bernstein in [3], by proposing variants of Salsa20 to improve on the cipher speed. Taking a similar view, we are of the opinion that ChaCha20 has excessive number of rounds.

In an effort to address this, we introduce the Hybrid Model for cryptanalysis for ARX based ciphers. It is essentially a two-part split of the running of the cipher. Initially for m rounds, the cipher is run with the original function as intended. Subsequent rounds of the cipher are run on the linear approximation of the round function, where the addition operation is replaced by XOR. Before we explain the rationale behind this, we state the piling-up lemma, which shall be used extensively.

Lemma 1 (Piling-up Lemma[17]). *Suppose that $x_1, x_2, \cdots x_k$ are independent random variables taking values from the set $\{0, 1\}$. Let $p_1, p_2, \cdots p_k$ be real numbers such that $\forall i,\ 0 \leq p_i \leq 1$, and $\mathsf{Pr}(x_i = 0) = p_i$. The bias[3] of x_i is defined to be $\varepsilon_i = 2 \cdot p_i - 1$. Let $\varepsilon_{1,2\cdots k}$ denote the bias of the random variable $x_1 \oplus x_2 \oplus \cdots \oplus x_k$. Then, $\varepsilon_{1,2\cdots k} = \prod_{j=1}^{k} \epsilon_i$.*

As can be seen, $-1 \leq \varepsilon_i \leq 1$. But since we will concern ourselves primarily with the magnitude, $|\varepsilon_i| \leq 1$ Since XOR is a bitwise operation, this result can be easily extended to variables $x_i \in \{0, 1\}^n$ with the bias of each bit in resultant the XOR calculated as above.

[3] In the section we will be considering only forward biases and will drop the subscript d, to denote the forward bias as ε.

The need for the initial non-linear rounds comes from the piling-up lemma. Assume, from the initial round, all the round functions are replaced by its linear approximation. Any introduced differential has an absolute value bias of 1 (or probability of 1), and the linear structure implies we can use the piling-up lemma to see how this bias propagates. A simple calculation shows that the absolute value of the resultant biases will always be 1. This leads to the idea that the absolute value of all the biases should be made strictly less than 1 before we can apply the linear approximation. We will return to this idea shortly.

3.1 Linear Approximation

The major challenge in the cryptanalysis of the ARX based ciphers is due to the non linearity introduced by the modular addition. Linear structures, where the addition is replaced by XOR are easier to analyse using the piling-up lemma. Unfortunately we cannot directly replace the modular additions directly by XOR in the ciphers without losing valuable information.

A function f, where all the modular additions have been replaced by XOR is said to be the linear approximation of f, say f^L. But before we can apply the piling-up lemma in this context, the independence requirement must be satisfied, and to this end, we make the following assumption.

Assumption 1 *After sufficient number of rounds, the bits of the state of the cipher would not have significant dependencies among themselves, and can be assumed to be independent.*

Theoretically this is not true, since bits in a given round are derived from the previous rounds. But for any good cipher, we would expect no noticeable dependencies after a few rounds. This estimate seems to serve well, as is evident from experimental results.

Using the linear approximation f^L, we now derive an upper bound for the biases of f.

Lemma 2. *For any function f consisting only of XOR and modular additions, if we consider its linear approximation f^L, the biases of any bit i of the output are related by*

$$|\varepsilon_i^L| \geq |\varepsilon_i| \tag{3}$$

Here ε_i^L is the bias for i^{th} bit of f^L, and ε_i is the corresponding bias for f. The inputs to the function are required to be independent.

Proof. Let the function f have n inputs $x_1, x_2, \cdots, x_n \in \{0,1\}^l$. In the case of the ciphers we consider, $l = 32$.

In an effort to move to the linear structure in the original function f, we use the concept of a carry vector. Using a carry vector, any modular addition (between say a and b) can be represented as

$$z = a + b = a \oplus b \oplus c \tag{4}$$

where c is the carry vector. If we express it in terms of bits,

$$z[i] = a[i] \oplus b[i] \oplus c[i] \tag{5}$$

where $c[0] = 0$, and

$$c[i] = a[i-1]b[i-1] \oplus (a[i-1] \oplus b[i-1])c[i-1] \text{ for } i = 1, \cdots 31.$$

It is important to revisit the independence assumption for this particular case before we can apply the piling-up lemma. Note that the i^{th} bit of c, $c[i]$, depends not on $a[i]$ and $b[i]$, but on the bits of a and b preceding i (by the definition of a *carry vector*). Due to the independence assumption among bits of $a[i]$ and $b[i]$ stated earlier, the independence condition holds here too for $a[i]$, $b[i]$ and $c[i]$. Note that we do not require the carry bits themselves to be independent, and this is certainly not true as shown in [7]. Even though we have provided the recursive formulation of $c[i]$, it will be clear shortly that the exact formulation of c is not necessary.

Now, let the function f contain k addition operations, with the remaining being XOR operations. This provides a count of the number of carry vectors we would require.

Each addition will require one carry vector, and hence let the carry vectors be denoted as $c_1, c_2, \cdots, c_k$. Then the function f, rewritten using carry vectors,

$$s = f(x_1, x_2, \cdots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n \oplus c_1 \oplus c_2 \oplus \cdots \oplus c_k$$

For the i^{th} bit of s, we obtain the bias using the piling up lemma

$$\varepsilon_{s[i]} = \varepsilon_{x_1[i]} \cdot \varepsilon_{x_2[i]} \cdots \varepsilon_{x_n[i]} \cdot \varepsilon_{c_1[i]} \cdot \varepsilon_{c_2[i]} \cdots \varepsilon_{c_k[i]} \tag{6}$$

Now, consider the linear approximation of f, f^L

$$s' = f^L(x_1, x_2, \cdots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

The bias here is,

$$\varepsilon^L_{s'[i]} = \varepsilon_{x_1[i]} \cdot \varepsilon_{x_2[i]} \cdots \varepsilon_{x_n[i]} \tag{7}$$

Since we only care about the absolute value of the bias, the ratio gives

$$\left| \frac{\varepsilon_{s[i]}}{\varepsilon^L_{s'[i]}} \right| = \frac{|\varepsilon_{s[i]}|}{|\varepsilon^L_{s'[i]}|} = \left|\varepsilon_{c_1[i]} \cdot \varepsilon_{c_2[i]} \cdots \varepsilon_{c_k[i]}\right| = |\varepsilon_{c_1[i]}| \cdot |\varepsilon_{c_2[i]}| \cdots |\varepsilon_{c_k[i]}|$$

Irrespective of how the vectors c_j are calculated, we know $|\varepsilon_i| \leq 1$. Hence,

$$\frac{|\varepsilon_{s[i]}|}{|\varepsilon^L_{s'[i]}|} = \left|\varepsilon_{c_1[i]} \cdot \varepsilon_{c_2[i]} \cdots \varepsilon_{c_k[i]}\right| = |\varepsilon_{c_1[i]}| \cdot |\varepsilon_{c_2[i]}| \cdots |\varepsilon_{c_k[i]}| \leq 1 \cdot 1 \cdots 1 = 1$$

$$\therefore |\varepsilon_{s[i]}| \leq |\varepsilon^L_{s'[i]}| \tag{8}$$

Equality is achieved *if* the original function is completely linear. □

If the independence assumption holds, the inclusion of bit-wise rotation operations in f does not alter the proof. We illustrate the above lemma with an example in line with the quarterround function of Salsa.

Example 1. Let $x_1, x_2, x_3 \in \{0,1\}^n$ and $f(x_1, x_2, x_3) = x_1 \oplus (x_2 + x_3)$. Alternatively, $f(x_1, x_2, x_3) = x_1 \oplus (x_2 \oplus x_3 \oplus c)$ where c is the carry vector for $(x_2 + x_3)$. The corresponding linearlized function is $f^L(x_1, x_2, x_3) = x_1 \oplus (x_2 \oplus x_3)$. Now, considering differentials, we get $\Delta f(\Delta x_1, \Delta x_2, \Delta x_3) = \Delta x_1 \oplus (\Delta x_2 \oplus \Delta x_3 \oplus \Delta c)$ and $f^L(\Delta x_1, \Delta x_2, \Delta x_3) = \Delta x_1 \oplus (\Delta x_2 \oplus \Delta x_3)$. Hence, from the piling-up lemma, for any bit i, the ratio of the biases $\frac{|\varepsilon_{f[i]}|}{|\varepsilon_{f^L[i]}|} = |\varepsilon_{c[i]}| \leq 1$.

Bit Dependencies. We now proceed to look at the linear approximations of Salsa and ChaCha and provide here the number of dependent bits in the linear approximation for both the ciphers. For the exact bit dependencies and bias equations, we refer the reader to Appendix C. In both Salsa and ChaCha, the linear approximation of quarterround is obtained by replacing all modular additions by XOR.

bit	# of dependent bits from previous round			
	b	c	d	a
$b'[i]$	1	0	1	1
$c'[i]$	1	1	1	2
$d'[i]$	2	1	3	3
$a'[i]$	3	2	4	6

(a) Salsa

bit	# of dependent bits from previous round			
	a	d	c	b
$a''[i]$	2	1	1	3
$d''''[i]$	3	2	1	4
$c''[i]$	4	3	2	5
$b''''[i]$	5	4	3	7

(b) ChaCha

Table 1: # of dependent bits from the previous round

When we discussed the motivation for a hybrid model earlier, we mentioned the necessity for the initial rounds to be non-linear. This initial non-linearity is to ensure that the absolute value of the bias at all the positions be strictly less than 1.

We present here the assumption, and refer the reader to a sketch of a possible proof in Appendix B.

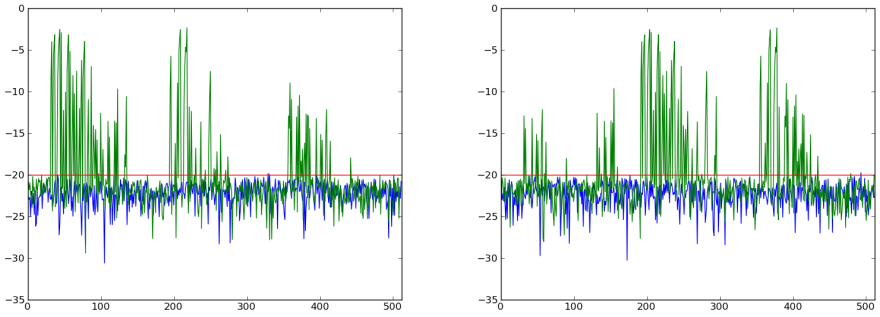
Assumption 2 *Let the bias after u rounds of the j^{th} bit of the i^{th} word be denoted as $\varepsilon_{i,j}^{(u)}$. After some rounds (say m), of the non-linear round function, $\exists \delta > 0$ such that $\forall i, j \;\; |\varepsilon_{i,j}^{(m)}| < 1 - \delta$.*

4 Calculations for Forward Biases

We use the result and assumptions stated in the previous section to provide a bound on the number of rounds required to achieve desired security against a distinguisher for forward biases. We like to identify the point at which the biases become infeasible for calculation experimentally. To this effect, we present without proof here the proposition made in [16].

Proposition 1 (Mantin and Shamir[16]). *Let X, Y be distributions, and suppose that 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.*

Hence, if it is the case that the absolute value of all the biases after some r rounds are less than $2^{-\frac{k}{2}}$, where k is the size of the key, the resultant time complexity for the distinguisher will be $\approx 2^k$, the same as a brute force key search. The lowest r for which this holds true should suffice for the the cipher in the forward direction. Intuitively, one would not expect R to be very high, as the biases would drop pretty rapidly. We present Figure 1 to show how the high biases after 4 rounds suddenly drops after 5 rounds. For explanations of these biases in more details, one may refer to [14].



(a) (b)

Fig. 1: Data corresponding to biases after 4^{th} (green) and 5^{th} (blue) rounds of Salsa. The X-axis represents the bits of a state, arranged linearly, and the Y-axis represents the $\log_2$ of the observed biases. The experiments were run for 2^{44} trials, with $\mathcal{ID}$s at (a)$\Delta^{(0)}_{7,31}$ and (b)$\Delta^{(0)}_{8,31}$.

We now proceed to the calculations for Salsa using our hybrid model. Let the bias at all positions (i, j) after m original rounds be $|\varepsilon^{\text{Salsa}}_{i,j}| < 1 - \delta$, for some $\delta > 0$. Consider the linear approximation of quarterround.

Since the bias bounds are same for all the variables after round m, application of piling-up lemma gives,

$$|\varepsilon^{\text{Salsa}}_{b^{(m+1)}}| < (1-\delta)^3,\ |\varepsilon^{\text{Salsa}}_{c^{(m+1)}}| < (1-\delta)^5,\ |\varepsilon^{\text{Salsa}}_{d^{(m+1)}}| < (1-\delta)^9,\ |\varepsilon^{\text{Salsa}}_{a^{(m+1)}}| < (1-\delta)^{15}$$

We see that bound for $|\varepsilon^{\text{Salsa}}_{b'}|$ is the largest, and hence the determining factor. If we were to stop at this point, we would require

$$(1-\delta)^3 \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{3}} \approx 0$$

This means, to stop after $m+1$ rounds, the variables should be almost perfectly random after m rounds, and hence not useful to us.

Now, let us proceed to the $(m+2)^{th}$ round. Note, unlike the previous round, the bounds for the biases are not the same throughout. Importantly, we should also note that the state matrix undergoes a transpose before the application of the quarterround function.

Having applied the transpose, the words taking the roles a and c remain unchanged, while those of b and d have swapped. This follows from the definition of the round function of Salsa. Calculating from the previously obtained biases,

$$|\varepsilon^{\mathsf{Salsa}}_{b^{(m+2)}}| < (1-\delta)^{1\cdot 9+0\cdot 5+1\cdot 3+1\cdot 15} = (1-\delta)^{27}$$
$$|\varepsilon^{\mathsf{Salsa}}_{c^{(m+2)}}| < (1-\delta)^{1\cdot 9+1\cdot 5+1\cdot 3+2\cdot 15} = (1-\delta)^{47}$$
$$|\varepsilon^{\mathsf{Salsa}}_{d^{(m+2)}}| < (1-\delta)^{2\cdot 9+1\cdot 5+3\cdot 3+3\cdot 15} = (1-\delta)^{77}$$
$$|\varepsilon^{\mathsf{Salsa}}_{a^{(m+2)}}| < (1-\delta)^{3\cdot 9+2\cdot 5+4\cdot 3+6\cdot 15} = (1-\delta)^{139}$$

Again b has the highest bias, to stop at this point, we would require

$$(1-\delta)^{27} \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{27}} = 0.037402$$

One can observe the bias requirement is already relaxing, and if the bias is less than `0.037402` after m rounds, we can stop after $m+2$ rounds.

For Salsa, since each bit of the word taking the role of b in every round is dependent on the least number of bits from the previous round, the bias bound of b is always the detrimental bias. Hence, we limit ourselves to calculating the bias of bits of b for $m+3$ rounds.

$$|\varepsilon^{\mathsf{Salsa}}_{b^{(m+3)}}| < (1-\delta)^{1\cdot 77+0\cdot 47+1\cdot 27+1\cdot 139} = (1-\delta)^{243}$$

The requirement to stop at this point is,

$$(1-\delta)^{243} \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{243}} = 0.694117$$

Here we only require the bias to be less that `0.694117`, which is considerably higher than the best known bias for 4 or 5 rounds of Salsa. If needed, calculations akin to the ones discussed here can be performed further.

Calculations for ChaCha, similar to Salsa, are provided in Appendix D and the results are summarized in the table below.

Here, it is important to note that m must be chosen such that the independence condition is valid for subsequent rounds. For Salsa , the multi-bit differentials for four rounds reported in [1] indicate dependence among the bits till the 4^{th} round at least. We conjecture that for both Salsa and ChaCha, the independence condition can be assumed starting with the 5^{th} round, and hence $m = 5$ is a reasonable assumption. The assumption would also imply that there would be no significant multi-bit differentials after m rounds.

rounds	bias bound requirement after m rounds	
	Salsa	ChaCha
$(m+2)$	0.037402	0.393008
$(m+3)$	0.694117	0.931587
$(m+4)$	0.960631	0.994635

Table 2: Summary of the bias bounds required to achieve computational indistinguishability after the mentioned rounds.

5 The Effect of PNBs

So far we have studied in details the bias in the forward direction. In this section we discuss how the idea of Probabilistic Neutral Bits (PNBs) applies to Salsa and ChaCha. We consider the known plaintext only attack model, where the 512-bit key stream of ChaChaR or Salsa20/R, i.e., $X + X^{(R)}$ is completely available to the attacker. The 256 key bits are not available, and the remaining 256 bits of X (i.e., the constants and IVs) are known. The strategy is to obtain the 256 secret key bits with a key search complexity less than exhaustive search, i.e., 2^{256}.

We have already discussed the forward biases in the previous section. We explain with a single bit $\mathcal{ID}$ and single bit $\mathcal{OD}$, though one can easily extend it for more than one bits. Let X, X' be two valid initial states with a given $\mathcal{ID}$ $\Delta_{i,j}^{(0)} = 1$, for which we observe a high bias ε_d in an $\mathcal{OD}$ $\Delta_{p,q}^{(r)}$ after $r < R$ ChaCha rounds. Thus, $\Pr(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \varepsilon_d)$, where $\Delta^{(r)} = X^{(r)} \oplus X'^{(r)}$. The two keystream blocks after R rounds are given by $Z = X + X^{(R)}$ and $Z' = X' + X'^{(R)}$.

Let us complement a particular key bit position κ, in both X and X', to yield the states $\overline{X}$ and $\overline{X'}$ respectively. Now consider the reversal of the states $Z - \overline{X}$ and $Z' - \overline{X'}$ by $R - r$ rounds to yield the states Y and Y' respectively. Let $\Gamma_{p,q} = Y_{p,q} \oplus Y'_{p,q}$. Given the $\mathcal{ID}$, if the bias in the event $(\Delta_{p,q}^{(r)} \oplus \Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)} = 1)$ is high, i.e., $\Delta_{p,q}^{(r)} = \Gamma_{p,q}$ with high probability, then we call the key bit κ a *Probabilistic Neutral Bit* (PNB). If $\Pr(\Delta_{p,q}^{(r)} \oplus \Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \gamma_\kappa)$, then γ_κ is called the *neutrality measure* of the key bit. One should run this experiment for each key bit several times over randomly chosen nonces and counter.

5.1 Identifying PNB Does Not Require Fixing $\mathcal{ID}$s

The main observation here, whether κ is a PNB or not, does not require fixing the $\mathcal{ID}$. This is supported by experiment only, and this is the reason certain independence between the biases in forward and reverse direction has been assumed in [1]. We have run detailed experiments in this direction that confirm this claim. The advantage of this observation is, while considering PNBs, such

a cipher described with a specific number of rounds can be characterized with much less effort.

In case that this had been dependent on the $\mathcal{ID}$s, for Salsa or ChaCha, we would have been require to experiment for 2^{128} different $\mathcal{ID}$s. However, in this case it is not required, and we can choose random $\mathcal{ID}$s and the average result of the experiments clearly identify the PNBs corresponding to the $\mathcal{OD}$s. This underlines the positions of $\mathcal{OD}$s one should try to mount the attack along the lines of [1]. For details of the attack one may refer to Appendix E.

One may note that the parameters for studying the PNBs are as follows:

- Fix the $\mathcal{OD}$ after r rounds where significant forward bias can be observed.
- Fix the number of reverse rounds $R - r$.
- Given those, for a key bit κ, what is the value of γ_κ.
- Finally, how many such key bits, say n, are there so that $\gamma_\kappa \geq \gamma$, for some previously fixed value γ.

These are related to the reverse bias which is denoted by ε_a, though the theoretical relationship seems elusive. As pointed out in [1], an attack seems feasible when $\varepsilon_d \varepsilon_a > 2^{-\frac{n}{2}}$ (see Appendix E).

Similar to our hybrid model for forward biases, one can consider nonlinear reverse rounds initially and then estimate using the linearized versions of the reverse rounds. However, we do not go for this analysis here in detail as we have already considered the number of forward rounds so that $|\varepsilon_d| < 2^{-\frac{k}{2}}$, where k is the number of secret key bits and naturally $n \leq k$. Further, it has been well studied that $|\varepsilon_a| < 1$ for $R - r = 4$ rounds given any single bit $\mathcal{OD}$ for both Salsa and ChaCha.

While there have been extensive studies for $R - r = 4$, we have performed additional experiments to observe that for $R - r = 5$ rounds the reverse biases reduce significantly and this we add while considering the total number of rounds in a safe design. Further details of the experiment will be made available in the full version of this paper.

6 Conclusion

In this paper we consider a hybrid model to evaluate Salsa and ChaCha under certain assumptions. Experimental evidences suggest we can assume certain independence assumptions after 5 rounds and then two more rounds are enough to reduce the biases to the order of 2^{-128}. Using the PNBs, experimental results suggest that one cannot obtain any significant bias for $R - r = 5$ reverse rounds that can be plugged with the forward biases. Thus, a total of $5 + 2 + 5 = 12$ rounds are sufficient for 256-bit security for both the ciphers. Combining these, using out heuristic arguments, we conclude that a total of 12 rounds would be sufficient to achieve desired security. Salsa20/12, as accepted in eStream, is of the same number of rounds. We thus suggest that ChaCha12 is sufficiently secure instead of deploying the proposed 20 rounds that substantially reduces the speed of ChaCha20. We believe that the way forward for cryptanalysis of Salsa

and ChaCha is to show multi-bit output differentials where bits from previous rounds cancel out and lead to better biases, but we have been unable to arrive at a case which would lead to drastic improvements over our analysis. Another alternative would be to demonstrate some linear dependencies among the bits to push up the value of m. This model also can have potential applications in other ARX based ciphers.

References

1. Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of Latin dances: analysis of Salsa, ChaCha, and Rumba. In *Fast Software Encryption*, pages 470–488. Springer, 2008.
2. Daniel Bernstein. Salsa20 security. 2005. `http://cr.yp.to/snuffle/security.pdf`.
3. Daniel Bernstein. Salsa20/8 and Salsa20/12. 2006. `http://cr.yp.to/snuffle/812.pdf`.
4. Daniel J Bernstein. Salsa20 specification. *eSTREAM Project algorithm description*, 2005. `http://www.ecrypt.eu.org/stream/salsa20pf.html`.
5. Daniel J Bernstein. ChaCha, a variant of Salsa20. In *Workshop Record of SASC*, volume 8, 2008.
6. Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Jean-Jacques Quisquater. On the salsa20 core function. In *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pages 462–469, 2008.
7. Joo Yeon Cho and Josef Pieprzyk. Multiple modular additions and crossword puzzle attack on nlsv2. In *Information Security, 10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007, Proceedings*, pages 230–248, 2007.
8. Paul Crowley. Truncated differential cryptanalysis of five rounds of Salsa20. *IACR Cryptology ePrint Archive*, 2005:375, 2005.
9. The ECRYPT stream cipher project. eSTREAM portfolio of stream ciphers. `http://www.ecrypt.eu.org/stream/`.
10. Simon Fischer, Willi Meier, Côme Berbain, Jean-François Biasse, and Matthew J. B. Robshaw. Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. In *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, pages 2–16, 2006.
11. Tsukasa Ishiguro, Shinsaku Kiyomoto, and Yutaka Miyake. Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha. In *Information and Communications Security - 13th International Conference, ICICS 2011, Beijing, China, November 23-26, 2011. Proceedings*, pages 255–266, 2011.
12. Gaëtan Leurent. Analysis of Differential Attacks in ARX Constructions. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 226–243, 2012.
13. Gaëtan Leurent. Construction of Differential Characteristics in ARX Designs Application to Skein. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 241–258, 2013.
14. Subhamoy Maitra. Chosen IV cryptanalysis on reduced round ChaCha and Salsa. *IACR Cryptology ePrint Archive*, 2015. `http://eprint.iacr.org/2015/698`.

15. Subhamoy Maitra, Goutam Paul, and Willi Meier. Salsa20 cryptanalysis: New moves and revisiting old styles. In *WCC 2015, the Ninth International Workshop on Coding and Cryptography, April 13-17, 2015, Paris, France.*, 2015. See also `http://eprint.iacr.org/2015/217`.
16. Itsik Mantin and Adi Shamir. A practical attack on broadcast RC4. In *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, pages 152–164, 2001.
17. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pages 386–397, 1993.
18. Nicky Mouha and Bart Preneel. A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:328, 2013.
19. `http://www.infosecurity-magazine.com/news/google-swaps-out-crypto-ciphers-in-openssl/`.
20. Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. In *Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers*, pages 337–351, 2012.
21. Yukiyasu Tsunoo, Teruo Saito, Hiroyasu Kubo, Tomoyasu Suzaki, and Hiroki Nakashima. Differential Cryptanalysis of Salsa20/8, 2007.

A Description of ChaCha

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

The rightmost matrix, similar to Salsa, shows the initial state, that takes four predefined constants $c_0, \ldots, c_3$ (similar to Salsa), 256-bit key $k_0, \ldots, k_7$, 32-bit block counter t_0 and 96-bit nonce v_0, v_1, v_2. Here, again, the basic nonlinear operation is the quarterround function. Each quarterround(a, b, c, d) consists of four ARX rounds, each of which comprises of addition (A), cyclic left rotation (R) and XOR (X) operation (one each) as given below:

$$\left.\begin{array}{l} a = a + b;\ d = d \oplus a;\ d = d \lll 16; \\ c = c + d;\ b = b \oplus c;\ b = b \lll 12; \\ a = a + b;\ d = d \oplus a;\ d = d \lll 8; \\ c = c + d;\ b = b \oplus c;\ b = b \lll 7; \end{array}\right\} \tag{9}$$

Each columnround works as four quarterrounds on each of the four columns of the state matrix and each diagonalround consists of four quarterrounds on each of the four diagonals. In ChaCha20, ten times the rowround, and ten times the diagonalround are applied alternatively to the initial state (total of 20 applications of quarterround).

In each of the odd rounds, we first apply quarterround to all the four columns in the following order: quarterround(x_0, x_4, x_8, x_{12}), quarterround(x_1, x_5, x_9, x_{13}), quarterround$(x_2, x_6, x_{10}, x_{14})$, and quarterround$(x_3, x_7, x_{11}, x_{15})$. This is a complete columnround. In each of the even rounds, we consider the order quarterround $(x_0, x_5, x_{10}, x_{15})$, quarterround$(x_1, x_6, x_{11}, x_{12})$, quarterround$(x_2, x_7, x_8, x_{13})$, and quarterround$(x_3, x_4, x_9, x_{14})$. This describes a complete diagonalround.

By $X^{(R)}$, we mean that R such rounds have been applied (in total, alternatively the columnround in odd rounds and diagonalround in even rounds, the initial round applied is considered as round 1) to the initial state $X^{(0)}$. For ChaCha20, there are 20 rounds, i.e., $R = 20$.

Each round of ChaCha is reversible and the reverse of each quarterround is defined as below:

$$\left.\begin{array}{l} b = b \ggg 7; \quad b = b \oplus c; \;\; c = c - d; \\ d = d \ggg 8; \quad d = d \oplus a; \; a = a - b; \\ b = b \ggg 12; \; b = b \oplus c; \;\; c = c - d; \\ d = d \ggg 16; \; d = d \oplus a; \; a = a - b; \end{array}\right\} \tag{10}$$

B Proof sketch for Assumption 2

The first step in this direction is to note, irrespective of the input differential, for a cipher to be deemed secure, after the initial few rounds, there should be at least one position with $|\varepsilon| < 1$. By the diffusion property of the ciphers, this bit would affect all the other bits in the state in a few subsequent rounds. Bernstein, in his security document on Salsa[2], illustrates a case where this happens in 4 rounds for a single bit differential. Lastly, once the absolute value of the bias drops below 1, it cannot be reinstated to the same. These would suffice to show that the assumption stated here is true.

An exact proof would involve considering the structure of each cipher, and would get rather tedious and messy. We believe, for our model, the exact proof is not of importance if the rationale behind why it should be true is understood.

C Bit Dependencies

C.1 Salsa

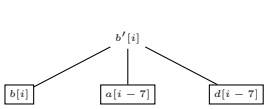
The exact bit dependencies for Salsa in the form of a tree, along with the bias equations are described below.

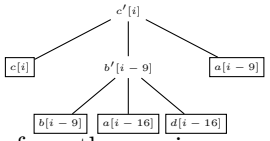
- Any bit $b'[i]$ is dependent on **three** bits from the previous round.

$$\varepsilon_{b'[i]} = \varepsilon_{b[i]} \cdot \varepsilon_{a[i-7]} \cdot \varepsilon_{d[i-7]}$$

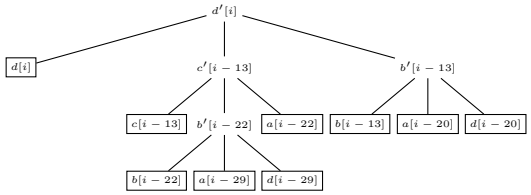
- Any bit $c'[i]$ is dependent on **five** bits from the previous round.

$$\varepsilon_{c'[i]} = \varepsilon_{c[i]} \cdot \varepsilon_{a[i-9]} \cdot \varepsilon_{b[i-9]} \cdot \varepsilon_{a[i-16]} \cdot \varepsilon_{d[i-16]}$$



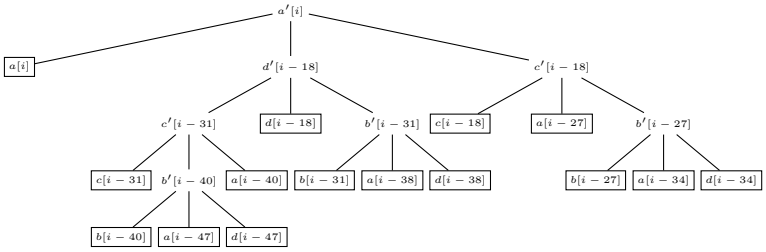


– Any bit $d'[i]$ is dependent on **nine** bits from the previous round.



$$\varepsilon_{d'[i]} = \varepsilon_{d[i]} \cdot \varepsilon_{c[i-13]} \cdot \varepsilon_{a[i-22]} \cdot \varepsilon_{b[i-22]} \cdot \varepsilon_{a[i-29]} \cdot \varepsilon_{d[i-29]} \cdot \varepsilon_{b[i-13]} \cdot \varepsilon_{a[i-20]} \cdot \varepsilon_{d[i-20]}$$

– Any bit $a'[i]$ is dependent on **fifteen** bits from the previous round.

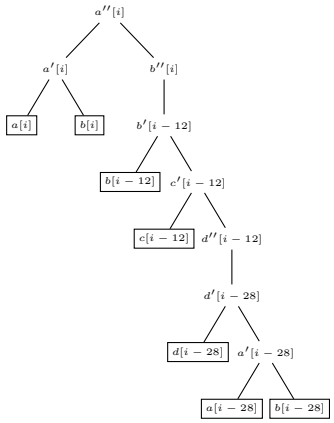


$$\varepsilon_{a'[i]} = \varepsilon_{a[i]} \cdot \varepsilon_{d[i-18]} \cdot \varepsilon_{c[i-31]} \cdot \varepsilon_{a[i-40]} \cdot \varepsilon_{b[i-40]} \cdot \varepsilon_{a[i-47]} \cdot \varepsilon_{d[i-47]} \cdot \varepsilon_{b[i-31]} \cdot \varepsilon_{a[i-38]} \cdot \varepsilon_{d[i-38]} \cdot \varepsilon_{c[i-18]} \cdot \varepsilon_{a[i-27]} \cdot \varepsilon_{b[i-27]} \cdot \varepsilon_{a[i-34]} \cdot \varepsilon_{d[i-34]}$$

C.2 ChaCha

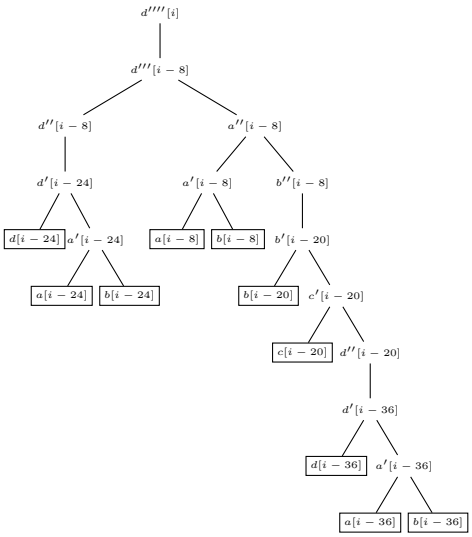
Making use of the above approximation, the bit dependency trees and the bias equations for ChaCha are provided below.

– Any bit $a''[i]$ is dependent on **seven** bits from the previous round.



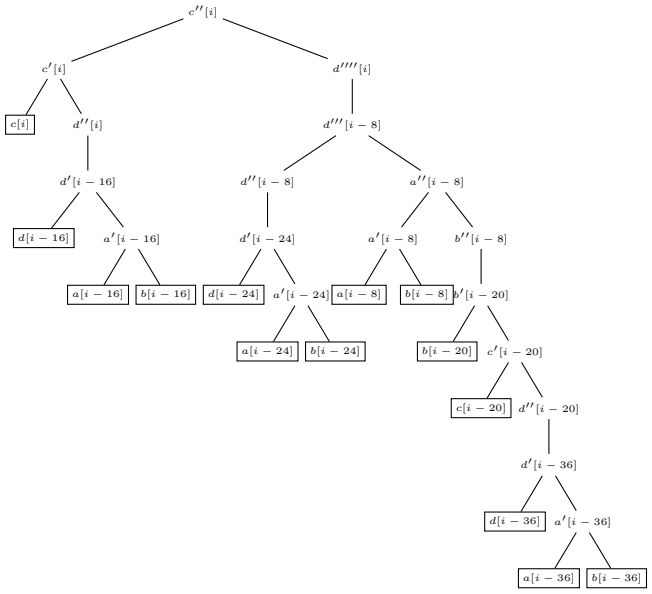
$$\varepsilon_{a''[i]} = \varepsilon_{a[i]} \cdot \varepsilon_{b[i]} \cdot \varepsilon_{b[i-12]} \cdot \varepsilon_{c[i-12]} \cdot \varepsilon_{d[i-28]} \cdot \varepsilon_{a[i-28]} \cdot \varepsilon_{b[i-28]}$$

– Any bit $d''''[i]$ is dependent on **ten** bits from the previous round.



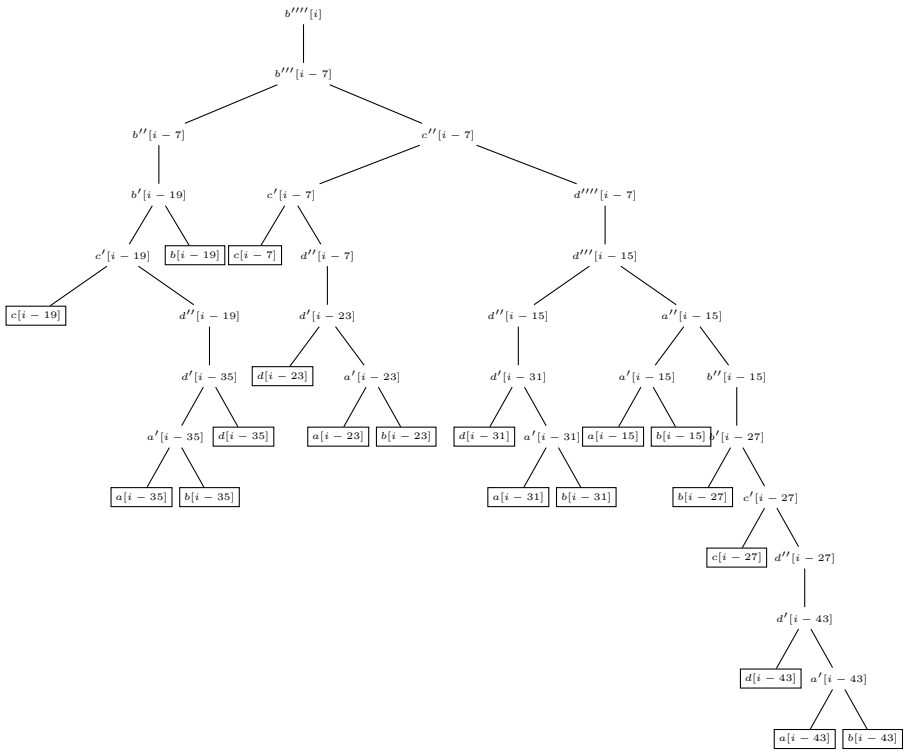
$$\varepsilon_{d''''[i]} = \varepsilon_{d[i-24]} \cdot \varepsilon_{a[i-24]} \cdot \varepsilon_{b[i-24]} \cdot \varepsilon_{a[i-8]} \cdot \varepsilon_{b[i-8]} \cdot \varepsilon_{b[i-20]} \cdot \varepsilon_{c[i-20]} \cdot \varepsilon_{d[i-36]} \cdot \varepsilon_{a[i-36]} \cdot \varepsilon_{b[i-36]}$$

– Any bit $c''[i]$ is dependent on **fourteen** bits from the previous round.



$$\varepsilon_{c''[i]} = \varepsilon_{c[i]} \cdot \varepsilon_{d[i-16]} \cdot \varepsilon_{a[i-16]} \cdot \varepsilon_{b[i-16]} \cdot \varepsilon_{d[i-24]} \cdot \varepsilon_{a[i-24]} \cdot \varepsilon_{b[i-24]} \cdot \varepsilon_{a[i-8]} \cdot \varepsilon_{b[i-8]} \cdot \varepsilon_{b[i-20]} \cdot \varepsilon_{c[i-20]} \cdot \varepsilon_{d[i-36]} \cdot \varepsilon_{a[i-36]} \cdot \varepsilon_{b[i-36]}$$

– Any bit $b''''[i]$ is dependent on **nineteen** bits from the previous round.



$$\begin{aligned}\varepsilon_{b''''[i]} = {} & \varepsilon_{c[i-19]} \cdot \varepsilon_{a[i-35]} \cdot \varepsilon_{b[i-35]} \cdot \varepsilon_{d[i-35]} \cdot \varepsilon_{b[i-19]} \cdot \varepsilon_{c[i-7]} \cdot \varepsilon_{d[i-23]} \cdot \varepsilon_{a[i-23]} \\ & \cdot \varepsilon_{b[i-23]} \cdot \varepsilon_{d[i-31]} \cdot \varepsilon_{a[i-31]} \cdot \varepsilon_{b[i-31]} \cdot \varepsilon_{a[i-15]} \cdot \varepsilon_{b[i-15]} \cdot \varepsilon_{b[i-27]} \cdot \varepsilon_{c[i-27]} \\ & \cdot \varepsilon_{d[i-43]} \cdot \varepsilon_{a[i-43]} \cdot \varepsilon_{b[i-43]}\end{aligned}$$

D Bias Calculations for ChaCha

Similar to Salsa, the bias at all positions (i, j) after m original rounds is assumed to be $|\varepsilon_{i,j}^{\text{ChaCha}}| < 1 - \delta$, for some $\delta > 0$. We use the linear approximation of quarterround.

The bias bounds are the same for each variable after round m, hence we get,

$$|\varepsilon_{a^{(m+1)}}^{\text{ChaCha}}| < (1-\delta)^7,\ |\varepsilon_{d^{(m+1)}}^{\text{ChaCha}}| < (1-\delta)^{10},\ |\varepsilon_{c^{(m+1)}}^{\text{ChaCha}}| < (1-\delta)^{14},\ |\varepsilon_{b^{(m+1)}}^{\text{ChaCha}}| < (1-\delta)^{19}$$

Unlike Salsa, the value of $|\varepsilon_{a^{(m+1)}}|$ is the largest and hence the determining factor. If we were to stop at this point,

$$(1-\delta)^7 \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{7}} \approx 0$$

As was the case with Salsa, stopping after $m+1$ rounds requires near perfect randomness after m rounds.

We now proceed to the $(m+2)^{th}$ round, but unlike Salsa, there is no transpose, and positions retain their roles irrespective of diagonalround or columnround being applied. Subsequently, we get

$$|\varepsilon^{\mathsf{ChaCha}}_{a^{(m+2)}}| < (1-\delta)^{2\cdot 7+1\cdot 10+1\cdot 14+3\cdot 19} = (1-\delta)^{95}$$
$$|\varepsilon^{\mathsf{ChaCha}}_{d^{(m+2)}}| < (1-\delta)^{3\cdot 7+2\cdot 10+1\cdot 14+4\cdot 19} = (1-\delta)^{131}$$
$$|\varepsilon^{\mathsf{ChaCha}}_{c^{(m+2)}}| < (1-\delta)^{4\cdot 7+3\cdot 10+2\cdot 14+5\cdot 19} = (1-\delta)^{181}$$
$$|\varepsilon^{\mathsf{ChaCha}}_{b^{(m+2)}}| < (1-\delta)^{5\cdot 7+4\cdot 10+3\cdot 14+7\cdot 19} = (1-\delta)^{250}$$

From the bias of a, we get,

$$(1-\delta)^{95} \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{95}} = 0.393008$$

As conjectured, the diffusion appears to be much faster in ChaCha than was observed in Salsa. Repeating this for a in round $(m+3)$ gives,

$$|\varepsilon^{\mathsf{ChaCha}}_{a^{(m+3)}}| < (1-\delta)^{2\cdot 95+1\cdot 131+1\cdot 181+3\cdot 250} = (1-\delta)^{1252}$$

The requirement now is,

$$(1-\delta)^{1252} \leq 2^{-128} \Rightarrow (1-\delta) \leq 2^{\frac{-128}{1252}} = 0.931587$$

A bias requirement for an upper bound 0.931587 should be achievable even with relatively few rounds of the original function.

E Background: Details of Attack Using PNBs

We explain here the ideas related to PNBs [1] in line with [15]. One can experiment this for sufficiently many samples corresponding to each key bit, which is enough to identify the biases. Repeating this for all the 256 key bits, a subset of the key bits can identified, which are called the PNBs. Typically, a threshold probability $\frac{1}{2}(1+\gamma)$ is chosen to filter the PNBs. If $\gamma_\kappa \geq \gamma$, then the key bit κ is included in the set of the PNBs. Suppose the size of this subset is n and therefore the number of non-PNB bits are $m = 256 - n$. The main idea behind the key recovery is to search these two sets separately.

After the set of PNBs is determined, the actual attack considers search over the key bits which are not PNBs. By considering a distinguisher, it is possible to identify when the correct keys have appeared. While studying the PNBs, in X and X', one complements a particular key bit position κ to yield the states $\overline{X}$ and $\overline{X'}$ respectively. However, for actual attack, we assign random values to all the PNBs. That is, we guess the key values to the m non-PNB key bits and assign random binary values to the n PNB key bits in both X and X' to yield the states $\hat{X}$ and $\hat{X}'$ respectively.

Then we reverse the states $Z - \hat{X}$ and $Z' - \hat{X}'$ by $R - r$ rounds to yield the states $\hat{Y}$ and $\hat{Y}'$ respectively. Let $\hat{\Gamma}_{p,q} = \hat{Y}_{p,q} \oplus \hat{Y}'_{p,q}$ and $\Pr(\hat{\Gamma}_{p,q} = 1 | \Delta^{(0)}_{i,j} = 1) =$

$\frac{1}{2}(1+\hat{\varepsilon})$. A higher absolute value of $\hat{\varepsilon}$ identifies that the non-PNBs have been chosen properly even without knowing the n PNBs.

Now consider the case when the guessed key values are correct. This is similar to assigning random binary values to all the 256 key bits in both X and X' to yield the states $\widetilde{X}$ and $\widetilde{X}'$ respectively. Then one can reverse the states $Z-\widetilde{X}$ and $Z'-\widetilde{X}'$ by $R-r$ rounds to yield the states $\widetilde{Y}$ and $\widetilde{Y}'$ respectively. Let $\widetilde{\Gamma}_{p,q}=\widetilde{Y}_{p,q}\oplus\widetilde{Y}'_{p,q}$ and $\Pr(\widetilde{\Gamma}_{p,q}=1|\Delta^{(0)}_{i,j}=1)=\frac{1}{2}(1+\widetilde{\varepsilon})$.

In actual key recovery attack, if the biases $\hat{\varepsilon}$ and $\widetilde{\varepsilon}$ can be efficiently distinguished, i.e., if the gap between the biases is significant with $\widetilde{\varepsilon}\approx 0$ (as it corresponds to a random event and should not have any bias), then we can conclude that the assignment $\hat{X}$ yields the correct values for the non-PNB bits. This needs to be experimented while choosing the set of PNBs.

In [1], the bias in the event $(\hat{\Gamma}_{p,q}=\Delta^{(r)}_{p,q})$ is denoted by ε_a. This needs to be studied while choosing the PNBs.

In the same work, the estimation of this bias is as follows. The key is fixed and one can vary the nonces and the counters to calculate one ε_a. Then it is possible to consider many randomly chosen keys to obtain a set of ε_a's and consequently compute the median ε^*_a's from this set. Similarly, the median ε^*_d can be estimated from the values of several ε_d's corresponding to different keys. Finally one can estimate ε^* as the median value[4] of ε's.

It was noted in [1] that ε^* can be approximated as $\varepsilon^*_d\cdot\varepsilon^*_a$. This underlines the fact that while estimating the PNBs and ε^*_a, the $\mathcal{ID}$ has no role. Thus, if one can come up with a set of IVs corresponding to a specific key for which ε_d can be increased substantially, then ε should also increase and thus the complexity of the attack will decrease as identified in [14].

As described in [1], given N, the number of samples used and $P_{fa}=2^{-\alpha}$, the probability of false alarm, the complexity of the attack is then given by

$$2^m(N+2^nP_{fa})=2^mN+2^{256-\alpha}, \tag{11}$$

where the required number of samples is

$$N\approx\left(\frac{\sqrt{\alpha\log 4}+3\sqrt{1-(\varepsilon^*)^2}}{\varepsilon^*}\right)^2 \tag{12}$$

for probability of non-detection $P_{nd}=1.3\times 10^{-3}$.

Roughly speaking, we need $2^mN<2^{m+n}$, i.e., $N<2^n$. Now N can be approximated by $(\varepsilon^*_d\varepsilon^*_a)^{-2}$. That is, we need $\varepsilon^*_d\varepsilon^*_a>2^{-\frac{n}{2}}$.

[4] The idea of using median is that, one can guarantee that the estimated probabilities will work for at least half of the keys.