# Improved Backward Bias computation for Salsa20 Cipher*

Sabyasachi Dey[1*][0000−0002−9984−4646], Gregor Leander[2**][0000−0002−2579−8587], and **Nitin Kumar Sharma**[1***][0000−0003−1009−7697]

[1] Department of Mathematics, Birla Institute of Technology and Science Pilani, Hyderabad, Jawahar Nagar, Hyderabad 500078, India.
[2] Ruhr University Bochum, Bochum, Germany
* sabya.ndp@gmail.com, **gregor.leander@rub.de, ***sharmanitinkumar685@gmail.com.

**Abstract.** In this paper, we present how to choose the actual value for so-called probabilistic neutral bits optimally. Because of the limited influence of these key bits on the computation, these are fixed to a constant value, often zero for simplicity. As we will show, despite the fact that their influence is limited, the constant can be chosen in significantly better ways, and intriguingly zero is the worst choice.

**Keywords:** Salsa20, Cryptanalysis, Probabilistic Neutral Bits, Backward bias.

## 1 Introduction

Salsa20 is, as part of the eSTREAM portfolio for software and is the base design for the cipher ChaCha used in TLS, one of the most important and analyzed stream ciphers today. It was designed by Daniel J. Bernstein in 2005 and was submitted to the ECRYPT Stream Cipher Project (eSTREAM).

The detailed structure and the use of various tools in its design have been discussed in [2]. Below we discuss the structure of 256-bit key version of Salsa20 cipher. The cipher is represented in a $4 \times 4$ matrix form consisting of 16 words, where each word is of 32 bits. The 256-bit key version of cipher takes 8 key words $(k_0, k_1, \ldots, k_7)$, 4 constants words $(c_0, c_1, c_2, c_3)$, 2 IV words $(t_0, t_1)$ and 2 counter words $(v_0, v_1)$ as input and generates a 512-bit output. The constant words $(c_0, c_1, c_2, c_3)$ for 256-bit key version have fixed value as: $c_0 =$ `0x61707865`, $c_1 =$ `0x3320646e`, $c_2 =$ `0x79622d32`, $c_3 =$ `0x6b206574`. The state matrix form of the Salsa20 is given by:

$$X = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} & X_{11} \\ X_{12} & X_{13} & X_{14} & X_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}.$$

* Admission year 2021, Guided By Dr. Sabyasachi Dey, Assistant Prof., Department of Mathematics, Birla Institute of Technology and Science Pilani, Hyderabad, Jawahar Nagar, Hyderabad 500078, India.

In Salsa20 cipher algorithm, the function operated in each round is a nonlinear operation which transforms a vector $(a, b, c, d)$ into $(a', b', c', d')$ by performing the $\oplus, \boxplus$ and $\lll$ operation for each round. Here, $\oplus$ denotes XOR operation between the bits, $\boxplus$ is the addition modulo $2^{32}$, $\lll$ is left cyclic rotation operation.

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

**Finding Probabilistic Neutral Bits** Now we explain the probabilistic neutral bits (also called as non-significant key bits) and the procedure of finding those bits. Using this procedure we partition the key bits into significant key bits and non-significant key bits. Non-significant key bits are those which influences the output difference bit with low probability. The aim of partitioning the key bits is that, instead of searching over all $2^{256}$ feasible possibilities of the key bits (for 256 bit key), if, for example $m$ bits are significant and the remaining $(256 - m)$ bits are non-significant, at first we aim to search the $m$ significant bits only. As a result, the maximum number of guesses is reduced to $2^m$. Once we achieve these bits, we can find the remaining bits by exhaustive search.

Let us define the non-significant bits or PNBs formally. For an initial state matrix $X$, after introducing a suitable non-zero input difference ($\mathcal{ID}$) we get another state $X'$. Running $X, X'$ for $r$-rounds $(1 \leqslant r < n)$ we observe the output difference ($\mathcal{OD}$) at position $(p, q)$, i.e., $\Delta X_p^{(r)}[q]$. The bias is given by $\epsilon_d$. After completing the n-rounds we obtain the final state $X^{(n)}$ and $X'^{(n)}$ which are operated with the respective initial states $X$ and $X'$ to obtain keystream blocks $Z$ and $Z'$. Considering figure 2, we get $X^{(r)} = F^{-1}(Z - X)$ and $X'^{(r)} = F^{-1}(Z' - X')$.

In the procedure of finding the PNB we will alter one key-bit say $l$ among the total keys (128 or 256) in the initial states $X$ and $X'$. $\bar{X}$ and $\bar{X}'$ are the new altered states. We apply the reverse round function on $Z - \bar{X}$ and $Z' - \bar{X}'$ by $(n - r)$-rounds and obtain the state matrices $\bar{M}$ and $\bar{M}'$ i.e., $\bar{M} = F^{-1}(Z - \bar{X})$ and $\bar{M}' = F^{-1}(Z' - \bar{X}')$. For a non-significant bit, the probability of the event $\Delta \bar{M}_p[q] = \Delta X_p^{(r)}[q]$ is expected to be high. We denote $\gamma_l$ to be the bias of this event, i.e.,

$$\Pr_{v,t}\left[\Delta X_p^{(r)}[q] \oplus \Delta \bar{M}_p[q] = 0 | \Delta X = D_{ID}\right] = \frac{1}{2}(1 + \gamma_l). \tag{2}$$

To construct the set of PNBs, in the work of [1], at first a threshold $\gamma$ is chosen. Each of the key bits for which $\gamma_i \geq \gamma$ are considered to be probabilistically neutral.

## 2 Improving the backward bias

We observe that the PNBs are located in the form of blocks, i.e., collections of consecutive bits. We decompose the subspace $k_s$ in the form of subspaces $k_{s_i}$, where each $k_{s_i}$ corresponds to a block. Now, consider one such block of PNBs of size $b$ at $X_i$ from $X_i[j]$ to $X_i[j+b-1]$. In our attack, after assigning the vector $\beta$ at the PNBs, we compute $Z-\tilde{X}$ and $Z'-\tilde{X}'$. We know that $\tilde{M}=F^{-1}(Z-\tilde{X})$ and $X^{(r)}=F^{-1}(Z-X)$. Therefore, in order to get a good backward bias, we aim to find out how closely $Z-\tilde{X}$ replicates $Z-X$. In simple words, more the number of bits of $Z-X$ matches with $Z-\tilde{X}$, more is the backward bias.

During the subtraction $Z-\tilde{X}$, the difference between $X$ and $\tilde{X}$ can also propagate to the bit at the position $j+b$ and onwards of $Z-\tilde{X}$ due to carry-propagation. Now, we observe that this probability of propagation varies based on the assigned values at the PNBs. So, if the assigned values at $\tilde{X}[j]\cdots\tilde{X}[j+b-1]$ can be chosen in such a way that the probability of this propagation can be reduced, we achieve a higher backward bias. We next analyze how the values can be chosen so that this probability of propagation is minimal.

**Which values of $\beta$ minimizes the probability of propagation** For each of $Z_i, X_i$ and $\tilde{X}_i$, if we consider the PNB block from $j$ to $j+b-1$ bit, each of them is a number between 0 to $2^b-1$. Let us denote them $z$, $k_n$ and $\beta$ respectively. From attackers prospective, $z$ and $k_n$ are unknown, and $\beta$ is decided by him/her. We aim to find which value of $\beta$ would be most suitable. The difference between $Z-X$ and $Z-\tilde{X}$ would propagate to $j+b$-th bit if either $k_n > z \geq \beta$ or $\beta > z \geq k_n$.

**Theorem 1.** *Let for some $\beta \in \{0,1,2,\cdots 2^b-1\}$, $S_\beta$=\{(z, x): either $x > z \geq \beta$ or $\beta > z \geq x$\}. Then $|S_\beta|$ is minimum if $\beta = 2^{b-1}-1$ or $\beta = 2^{b-1}$, and maximum if $\beta = 0$ or $2^b-1$.*

*Proof.* Possible values of $(x,z)$ such that $x > z \geq \beta$ is ${}^{2^b-\beta}C_2$, since $x, z$ can be any integer in the range $[\beta, 2^b-1]$. Similarly, possible values of $(x,z)$ such that $\beta > z > x$ is ${}^{\beta}C_2$, and possible values such that $\beta > z = x$ is $\beta$. Therefore,

$$\begin{aligned}|S_\beta| = {}^{2^b-\beta}C_2 + {}^{\beta}C_2 + \beta &= \frac{(2^b-\beta)!}{(2!)(2^b-\beta-2)!} + \frac{\beta!}{(2!)(\beta-2)!} + \beta \\ &= 2^{2b-1} - 2^b \times \beta - 2^{b-1} + \beta^2 + \beta.\end{aligned}$$

We aim to find the $\beta$ for which $|S_\beta|$ is minimum. For this, we can write it as:

$$\begin{aligned}|S_\beta| &= 2^{2b-1} - 2^{b-1} + \beta^2 - (2^b-1)\times\beta \\ &= 2^{2b-1} - 2^{b-1} - \left(\frac{2^b-1}{2}\right)^2 + \left(\beta - 2^{b-1} + \frac{1}{2}\right)^2.\end{aligned}$$

The term $(\beta - 2^{b-1} + \frac{1}{2})^2$ is non-negative. Since $\beta$ is an integer, $(\beta - 2^{b-1} + \frac{1}{2})^2$ gives minimum value either at $\beta = 2^{(b-1)} - 1$ or at $\beta = 2^{b-1}$, and at both of them $(\beta - 2^{b-1} + \frac{1}{2})^2 = \frac{1}{4}$. So, in both these cases, we get $g(2^{(b-1)} - 1) = 2^{2(b-1)}$ and $g(2^{b-1}) = 2^{2(b-1)}$. Similarly, to find the $\beta$ for which $|S_\beta|$ is maximum, we focus on the term $\left(\beta - 2^{b-1} + \frac{1}{2}\right)^2$. It gives the maximum value when $\beta = 0$ or $2^b - 1$.

**Observation for 7-round 128-bit key version of Salsa20** We experiment on a set of four consecutive PNBs $\{13, 14, 15, 16\}$ for 7 round 128-bit Salsa20. We know that there can be $2^4$ possible values of this PNB set, i.e., $\{0, 1, 2, \cdots 15\}$. For each of these values, we will find the value of $\epsilon_a$ (backward bias). We observe that we got a maximum backward bias for the values $7, 8$ and a minimum backward bias at $0, 15$. In this case b=4, hence from above theorem $|S_\beta|$ is minimum at $\beta = (2^{b-1} - 1) = (2^3 - 1) = 7$ or $\beta = 2^{b-1} = 2^3 = 8$.

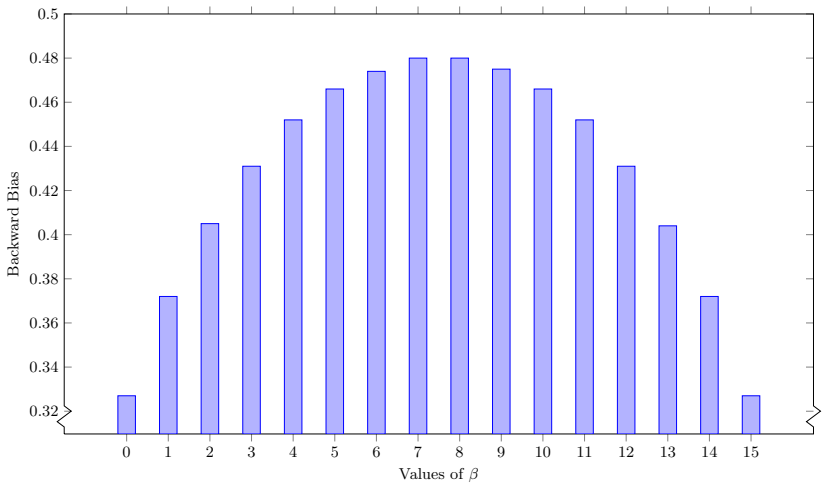


Fig. 1: Graphical Representation of the backward biases for different values of $\beta$ for the PNBs $\{13, 14, 15, 16\}$

## References

1. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: Analysis of Salsa20, ChaCha and Rumba. In: FSE 2008, Revised Selected Papers. LNCS, vol. 5086, pp. 470–488. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4_30.
2. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.). New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 84–97. Springer (2008). https://doi.org/10.1007/978-3-540-68351-3_8.