Proving the Forward Bias of Salsa

Sabyasachi Dey and Santanu Sarkar

Department of Mathematics,
Indian Institute of Technology Madras,
Sardar Patel Road, Chennai 600036, India
{sabya.ndp,sarkar.santanu.bir}@gmail.com

Abstract. Salsa20 is one of the most famous stream ciphers in recent times. Most of the attacks available so far against Salsa are differential attacks, where a difference is given as an input in the initial state of the cipher and in the output, some correlation is investigated, which works as a distinguisher. All the key recovery attacks in the last decade against Salsa are based on this observed distinguisher. However, the distinguisher in the differential attack was purely an experimental observation, and the proper reason for this bias was unknown so far. In this paper, we provide a full theoretical proof of this observed distinguisher. This is the first attempt to provide a theoretical justification of this bias which plays the major role in all the key recovery attacks so far against this cipher.

Keywords: Salsa, Bias, Theoretical justification

1 Introduction

Salsa20 was designed by D. Bernstein in the year 2005 as a candidate for eStream [2]. It was one of the finalists in the competition. The original version of Salsa has 20 rounds. However, the designer submitted the 12 rounds version in eStream. The first cryptanalysis against Salsa was presented by Crowley in 2005 [4], who attacked it upto five rounds. Later, six rounds and seven rounds Salsa were attacked respectively by Fischer et al. [6] and Tsunoo et al. [10]. So far, this cipher has been attacked only upto 8 rounds. The central ideas of most of the attacks are based on differential distinguisher. Fischer [6] used it in a key recovery attack upto sixth round. In 2008, Aumasson et al. [1] produced a significant improvement in this attack by introducing the idea of probabilistically neutral bits. By using the differential in the 4th round, they attacked Salsa upto 8th round. Afterwards, there has been further improvement in the attack complexities in last decade [9, 8, 7]. In [3] a distinguisher has been found upto 5 rounds by a linear combination of multiple bits of the output. But still, no work has been able to extend the attack upto 9th round in the last decade.

For a major improvement in this attack method, we need a detailed analysis of the whole attack procedure. The whole idea is so far mostly based on experimental observations only. Starting from the distinguisher, the idea of probabilistic neutral bits, meet-in-the middle attack etc. are mostly relying on experiments. In this attempt, we go through the internal mechanism of how the differential distinguisher is created. Our aim is not to reduce the complexity of the attack, but find out the actual reason for the already observed results.

2 Structure

Salsa uses a 256 bit key in its algorithm. Another 128 bit key version is there, which replicates its 128 bit key to another copy and produces total 256 keybits. The internal state of Salsa is basically a matrix of size 4×4. Each of the 16 cells contains a 32-bit binary number, which is called a 'word'. These 16 words or cells can be divided into three categories:

1. **Constant cells:** These cells lie in the diagonal of the matrix. These contain some predefined constant 32-bit numbers and are denoted by c_i's. The values of these cells are given below in hexadecimal form:
   $c_0 = 0x61707865, c_1 = 0x3320646e, c_2 = 0x79622d32, c_3 = 0x6b206574$.
2. **Key cells:** There are eight cells in the matrix which contain the keybits. These are denoted by $k_0, k_1, \ldots, k_7$.
3. **Counters and nonces:** There are four cells t_0, t_1, v_0 and v_1 which are counters and nonces. These are also called IVs.

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

The initial matrix is denoted by X or $X^{(0)}$. This matrix is updated by a function called quarterround. After completion of r rounds, we denote the updated matrix as $X^{(r)}$. The quarterround function works on a 4-tuple (a, b, c, d). It uses three operations: addition modulo 2^{32} $(+)$, left rotation $(\lll)$ and XOR $(\oplus)$. The function is as follows:

$$b = b \oplus ((a + d) \lll 7),$$

$$c = c \oplus ((b + a) \lll 9),$$

$$d = d \oplus ((c + b) \lll 13),$$

$$a = a \oplus ((d + c) \lll 18).$$

This function works on the columns and rows of the matrix in alternative round. In the odd rounds it works on the columns, and this is called columnround. Application on rows are performed in the even rounds and called rowround.

Notations: X_i denotes the ith cell of the matrix X; $X_i[j]$ denotes the jth bit of the cell X_i; $X^{(r)}$ denotes the matrix after r rounds; $X_i^{(r)}[j]$ denotes the jth bit of the ith cell of the matrix after r rounds; location (p, q) denotes the qth bit of the pth cell; $\Pr(A)$ denotes the probability of the event A.

Differential attack and our contribution: Here we discuss in brief the idea of a differential attack on Salsa. A single bit of a IV is changed in the initial matrix X producing a new matrix X'. As in the work of Aumasson et al. [1], the difference is given in 31st bit of X_7. Then, the Salsa algorithm is applied for four rounds on both X and X'. After four rounds, a correlation is observed on 14th bit of X_1 and X_1'. This correlation is that the probability that the bit $(1, 14)$ of X and X' (after four rounds) are equal with probability 56%. This is called the forward bias. This is an experimental observation, and no theoretical approach has been made to justify the proper reason for this bias. In this paper, we attempt to prove this bias. We track the propagation

of the input difference after each round and reach the fourth round to find the bias in the position $(1, 14)$, (14th bit of the cell X_1). In this context, we clearly mention here that throughout paper whenever we say that a cell or bit is "not affected" or "not influenced" by the difference, we mean that upto that round, that cell or that bit position has the same value for X and X'.

3 Some mathematical results

Here, at first we state without proof a lemma from [5]. For the proof, one can check [5].

Lemma 1. *$a = a_{31}a_{30}a_{29}\cdots a_0$ and $b = b_{31}b_{30}b_{29}\cdots b_0$ be two independent arbitrarily chosen numbers of 32 bits. Let $b' = b'_{31}b'_{30}b'_{29}\cdots b'_0$ be a number which differs at exactly one bit (say n-th, $n \leq 31$) from b. Consider $S = a + b \bmod 2^{32}$ and $S' = a' + b' \bmod 2^{32}$. Then for any $k \geq 0$ such that $n + k \leq 31$, the probability that S and S' will differ at $(n + k)$th bit is $\frac{1}{2^k}$.*

We will use this result when we prove the observed bias for Salsa.
Next, we deal with a case which is slightly more complicated and generalised.

Theorem 1. *Let a, b are two independent randomly chosen single bit number. Let a', b' be single bit numbers such that $\Pr(a = a') = p$ and $\Pr(b = b') = q$. Let $s = a + b$ and $s' = a' + b'$. Then the probability $\Pr(s_i = s'_i)$ can be given by:*

1. *$pq + (1 - p)(1 - q)$ for $i = 0$.*
2. *$\frac{1+pq}{2}$ for $i = 1$.*

Proof. **For $i = 0$:** In this case, $s_0 = a \oplus b$ and $s'_0 = a' \oplus b'$. So, $s_0 = s'_0$ implies $s_0 \oplus s'_0 = 0$. Thus $(a \oplus a') \oplus (b \oplus b') = 0$. So, either $(a \oplus a') = (b \oplus b') = 0$ or $(a \oplus a') = (b \oplus b') = 1$.
In the first case, the probability is:

$$\begin{aligned}\Pr((a = a') \cap (b = b')) &= \Pr(a = a') \cdot \Pr(b = b') \text{ (independence)} \\ &= pq.\end{aligned}$$

In the second case, the probability is

$$\Pr((a \neq a') \cap (b \neq b')) = \Pr(a \neq a') \cdot \Pr(b \neq b') = (1 - p)(1 - q).$$

For $i = 1$: Since a and b are single bits, so s_1 and s'_1 are basically the carries generated in the previous sums. So, we find out the possible ways of generation of a different carry in $a + b$ and $a' + b'$.
Event 1: One of (a, b) or (a', b') is $(0, 0)$ and the other one is $(1, 1)$. In this case, the tuple $(1, 1)$ generates a carry 1, but $(0, 0)$ does not. So, $s_i \neq s'_i$. Now, the probability of this event we calculate in the following way.

We first focus on the tuple (a, b). The probability that it takes the value either $(0, 0)$ or $(1, 1)$ is $\frac{1}{2}$, since there are total four possible options. Once this value is assigned, (a', b') must choose the other possible value. This means, a' must not be equal to a and b' must not be equal to b. This probability should be $(1-p)(1-q)$. Therefore, the probability of this whole event is $\frac{1}{2}(1 - p)(1 - q)$.
Event 2: $a = a' = 1$ and $b \neq b'$. In this case, one of the tuples (a, b) and (a', b') is $(1, 1)$, which generates carry 1, and the other one is $(1, 0)$, which generates carry 0.

Now, $\Pr(a = a' = 1) = \Pr(a = a') \cdot \Pr(a = 1) = p \cdot \frac{1}{2} = \frac{p}{2}$. Then, $\Pr(b \neq b') = (1 - q)$. Therefore, the probability of this event is $\frac{p}{2}(1 - q)$.
Event 3: $b = b' = 1$ and $a \neq a'$. This event is similar to the previous one. By similar arguments it can be shown that the probability is $\frac{q}{2}(1 - p)$.
These three events are mutually disjoint and other than these three events there is no way of producing $s_i \neq s'_i$. Therefore the probability of $s_i \neq s'_i$ is the sum of these three probabilities, which is

$$\frac{1}{2}(1-p)(1-q) + \frac{p}{2}(1-q) + \frac{q}{2}(1-p) = \frac{1-pq}{2}.$$

Therefore, the probability of equality can be given by $1 - \left(\frac{1-pq}{2}\right) = \frac{1+pq}{2}$.

■

Now, let us generalise this result little bit further. Suppose a, b, a', b' are not single bit numbers. We focus on some ith bit of them. Suppose we know the correlation between the corresponding bits of a and a' (and similarly for b and b'). The question is whether we can find a probabilistic relation between the ith bit of s and s'. One important point that we have to keep in mind is that the ith bit of s does not depend only on a_i and b_i. There is an involvement of a carry also, which is produced from the previous bits. Suppose, by c_i and c'_i we denote the carries generated at $(i-1)$th bits of s and s' respectively, which are to be added in ith bit. So, $s_i = a_i \oplus b_i \oplus c_i$ (same for s').

Theorem 2. *Suppose a, b be two independent and randomly chosen n-bit numbers. Let a', b' be n-bit numbers such that for all $j = \{0, 1, \ldots, (n-1)\}$, $\Pr(a_j = a'_j)$ is given by p_j and $\Pr(b_j = b'_j)$ is given by q_j. Suppose $s = a + b$ and $s' = a' + b'$. Then,*

$$\Pr(c_{i+1} \neq c'_{i+1}) = \Pr(c_i \neq c'_i) \cdot \left(1 - p - q + \frac{3pq}{2}\right) + \Pr(c_i = c'_i)\frac{(1-pq)}{2}.$$

Proof. Suppose $c_i \neq c'_i$. Without loss of generality, let us assume that $c_i = 1$ and $c'_i = 0$. Let us find the possible cases where c_{i+1} and c'_{i+1} are different.
Event 1: One of (a_i, b_i) and (a'_i, b'_i) is $(0, 0)$ and the other one is $(1, 1)$. If $(a_i, b_i) = (1, 1)$ and $(a'_i, b'_i) = (0, 0)$, then $a_i + b_i + c_i = 3$, which generates a carry 1. On the other side, $a'_i + b'_i + c'_i = 0$, which generates carry 0. Therefore $c_{i+1} \neq c'_{i+1}$. If $(a'_i, b'_i) = (1, 1)$ and $(a_i, b_i) = (0, 0)$, then $a_i + b_i + c_i = 1$, which generates carry 0 and $a'_i + b'_i + c'_i = 2$, which generates carry 1. Therefore, again $c_{i+1} \neq c'_{i+1}$. The probability of this event is $\frac{(1-p)(1-q)}{2}$, as computed in the previous theorem.
Event 2: $(a_i, b_i), (a'_i, b'_i) \in \{(0, 1), (1, 0)\}$ and $(a_i, b_i) = (a'_i, b'_i)$. In this case, $a_i + b_i + c_i = 2$, which generates carry 1 and $a'_i + b'_i + c'_i = 1$, which generates carry 0. So, $c_{i+1} \neq c'_{i+1}$. The probability of this event can be calculated as $\Pr(a_i = a'_i) \cdot \Pr(b_i = b'_i) \cdot \Pr(a_i \neq b_i) = \frac{pq}{2}$.
Event 3: $(a_i, b_i), (a'_i, b'_i) \in \{(0, 1), (1, 0)\}$ and $(a_i, b_i) \neq (a'_i, b'_i)$. Similar to the previous event, one can easily verify that in this event $c_{i+1} \neq c'_{i+1}$. Also, by similar arguments as the previous event, the probability can be calculated as $\frac{(1-p)(1-q)}{2}$. These three events are mutually disjoint. So, given $c_i \neq c'_i$, the total probability of the event $(c_{i+1} = c'_{i+1})$ can be computed by adding them. This sum is $1 - p - q + \frac{3pq}{2}$. On the other hand, if $(c_i = c'_i)$, then both of them can be either 0 or 1. If 0, then we can treat a_i, b_i, a'_i, b'_i like the least significant bits as in Theorem 1. So, the probability is $\frac{1-pq}{2}$. Similarly, if both the carries are 1, it can be shown by similar method that the probability is $\frac{(1-pq)}{2}$.

So, the total probability of $\Pr(c_{i+1} \neq c'_{i+1})$ can be calculated as $\Pr(c_i \neq c'_i) \cdot (1 - p - q + \frac{3pq}{2}) + \Pr(c_i = c'_i)\frac{(1-pq)}{2}$. ■

Corollary 1. *Supose a, a', b, b' are as in Theorem 2. Then, if $q_i = \frac{1}{2}$, $\Pr(c_{i+1} \neq c'_{i+1}) = \frac{1}{2} - \frac{p_i}{4}$.*

Proof. Let us prove it for $i = 0$. So, the probability $\Pr(c_1 \neq c'_1)$ can be given by Theorem 1, which is $\frac{1-p_0q_0}{2}$. Putting $q_0 = \frac{1}{2}$, we get $\frac{1}{2} - \frac{p_0}{4}$. Therefore the result is true for $i = 0$. Now, for $i = 1$, we use these two results in Theorem 2 and putting $q_1 = \frac{1}{2}$, we get the probability as $\frac{1}{2} - \frac{p_1}{4}$. Similarly, using this result we can prove the result for $i = 2$. Proceeding this way the result can be proved for any i. ■

Theorem 3. *Consider a, b, a', b' as n-bit numbers similar to 2. Also, s, s', c_j and c'_j are as mentioned. Now, suppose at ith bit, $c_i \neq c'_i$. The probabilities $\Pr(a_i = a'_i)$ and $\Pr(b_i = b'_i)$ are given by p and q respectively. Then, the probability that the ith bit of the sums s and s' matches is $p(1 - q) + q(1 - p)$.*

Proof. Without loss of generality, let us assume that $c_i = 1$ and $c'_i = 0$. Now, $s_i = a_i \oplus b_i \oplus c_i = a_i \oplus b_i \oplus 1$ and $s'_i = a'_i \oplus b'_i \oplus c'_i = a'_i \oplus b'_i$. Therefore $s_i = s'_i$ implies $a_i \oplus b_i \oplus a'_i \oplus b'_i = 1$. Therefore, either $a_i \oplus a'_i = 0$ and $b_i \oplus b'_i = 1$, or $a_i \oplus a'_i = 1$ and $b_i \oplus b'_i = 0$. Probability of the first event is $\Pr(a_i = a'_i) \cdot \Pr(b_i \neq b'_i) = p(1 - q)$. Similarly, probability of the second event is $q(1 - p)$. Adding these two probabilities, we get the desired result. ■

4 Salsa

In this section we prove the forward bias for Salsa, i.e., the probability that the bit $(1, 14)$ of X and X' after four rounds matches with probability approximately 0.56. It is expected that as the algorithm progresses, the correlation between X and X' decreases and randomness increases. So, one important point is that our aim is to provide theoretical justification only wherever we observe bias. If in any case, we observe that there is no bias, obviously we do not attempt to justify the reason for the unbiasedness.

First round:

In the first round, at first we find $\Pr(X_{15}[0] = X'_{15}[0])$. In that context, we have the following result.

Lemma 2. *After the completion of the first round, the probability that the 0th bit of X_{15} and X'_{15} matches is 0.75.*

Proof. In the columnround operation on the last row of X, we have: $(a, b, c, d) = (X_{15}, X_3, X_7, X_{11})$. The input difference is given in 31st bit of X_7. Since in the quarterround function b (here X_3 and X'_3) is updated first, and X_7 (here it is c) is not involved in the update function of b, there is no difference between X_3 and X'_3 after the update, i.e., $X_3 = X'_3$. In the next step, c (X_7 and X'_7) is updated. It involves the variables a and b, which are X_3 and X_{15}. Both of them are same for X and X' so far. Therefore, these variables do not bring any difference between X and X'. The only difference is caused by the XOR of c, which is X_7. So, after the update, X_7 and X'_7 has only one difference, which is at 31st bit.

$\boldsymbol{d = d \oplus ((c + b) \lll 13)}$: In the update of d, the involved variables are d, c, b. $d(X_{11})$

and $b(X_7)$ do not have any difference between X and X'. c has difference in 31st bit. So in the addition $c+b$, exactly one difference is there between X and X', which is at 31st bit. After the rotation by 13 bits, that difference comes at 12th bit. So, X_{11} and X'_{11} has only one bit difference, which is at 12th bit.
$\boldsymbol{a = a \oplus ((d + c) \lll 18)}$: In the final update, a has no difference between X and X'. d has a difference at 12th position, and c has difference at 31st. In the sum $c + d$, difference will be at 12th and 31st bit. But, since this is an addition operation, the difference at 12th bit may propagate to the next bit. The probability of this propagation we calculate using Lemma 1. Here, $n = 12, k = 2, n + k = 14$. So, the probability of the propagation is $\frac{1}{2^2} = \frac{1}{4}$. After left rotation by 18 bits, this difference comes to the 0th bit. So, $\Pr(X_{15}[0] = X'_{15}[0]) = 1 - \frac{1}{4} = 0.75$. ■

Second round: In the second round, we prove a small but important result which we will use later. It says that upto the second round, the least significant bit of the cell X_{11} is not affected by the difference propagation.

Lemma 3. *After the completion of the second round,* $X_{11}[0] = X'_{11}[0]$.

Proof. $(a, b, c, d) = (X_{10}, X_{11}, X_{12}, X_{13})$. In the first step, we have: $b = b \oplus ((a + d) \lll 7)$. Here, a and d have no difference in X and X', because after the first round, the propagation of the difference is within the fourth column only. So, the only difference can be produced by the XOR of b, which has only one difference at 12th bit. Therefore, 0th bit is not affected by this operation. ■

Third Round: Now we provide the proof of some biases that we observe in the third round. In these proofs, we are going to use the results obtained in the first and second round.

First column: In this column, the tuple (a, b, c, d) is (X_0, X_4, X_8, X_{12}).

Theorem 4. $\Pr(X_4[8] = X'_4[8]) \approx 0.998$.

Proof. In the first round, the cell X_4 is not affected by the propagation of input difference, since the propagation stays within the fourth column only. In the second round, the rowround operation on the second row helps the propagation of the difference into X_4 during the operation $d = d \oplus ((b + c) \lll 13)$. In this operation, since c (in this case, X_7) has a difference at 31st bit, it brings difference in 12th bit of d (because of the left rotation by 13 bits). The bits on the left side of 12th bit are also affected probabilistically, which decreases gradually as the distance increases. But, since the addition does not have any effect on the bits which are on the right side of 12th bit are not affected by this operation. Therefore, even after completion of the second round, we have $X_4[i] = X'_4[i]$ for all $i < 12$. Now, in the third round, X_4 is updated as by the function: $b = b \oplus ((a + d) \lll 7)$, where a, d are respectively X_0 and X_{12} from the second round. We focus on the 1st bit of $(a + d)$, since it reaches the 8th bit position after rotation by 7 bits. Now, since X_0 is on the first row, it is not affected by the propagation upto the 2nd round. Therefore we only focus on X_{12}. In the first round, $X_{12} = X'_{12}$. In the second round, it is updated by the function:

$$b = b \oplus ((a + d) \lll 7),$$

where a, d are X_{15} and X_{14} from the 1st round. After the left rotation, the 26th bit comes in place of 1st bit. Therefore, we focus on 26th bit of $X_{15}+X_{14}$. X_{14} is unaffected

in the first round (3rd column). So, the difference may generate only from X_{15}. Now, X_{15} has difference at 17th bit, but no difference afterwards upto 26th bit. Therefore the probability that during the addition $a+d$, the difference would propagate to 26th bit is $\frac{1}{2^{26-17}} = \frac{1}{2^9}$. Therefore, the probability that after the second round, $(X_{12}[1] = X'_{12}[1])$ is $1 - \frac{1}{2^8} \approx 0.998$. Therefore, after the 3rd round, $\Pr(X_4[8] = X'_4[8]) \approx 0.998$. ■

Theorem 5. *After the 3rd round,* $\Pr(X_{12}[21] = X'_{12}[21]) \approx 0.025$.

Proof. X_{12} is updated by the function:

$$d = d \oplus ((c + b) \lll 13),$$

where c and b are respectively X_8 and X_4. Since the rotation is by 13 bits, $X_{12}[21]$ is basically the XOR of $X_{12}[21]$ (after the second round) and 8th bit of $(X_8 + X_4)$. Here, X_8 and X_4 are already updated upto the third round.

X_{12}[21] after the second round: This cell is unaffected in the first round. In the second round it is updated by

$$b = b \oplus ((a + d) \lll 7),$$

where a, d are X_{15} and X_{14} respectively. So, we focus on the 14th bit of $X_{15} + X_{14}$. The only difference can come by the propagation of the difference of X_{15} at 0th bit (Lemma 2) and next few bits, whose effect is neglible at 14th bit. Therefore, the probability of difference is approximately 0. So, after the second round, $\Pr(X_{12}[21] = X'_{12}[21]) \approx 1$.

$(X_8 + X_4)$[8] after third round: Upto 2nd round, the 8th bit of X_8 and X_4 are not influenced by the difference. In the third round, X_8 is XORed with $X_4 + X_0$. Here, 31st bit of X_4 is different from X'_4 with probability almost 1, because it was XORed with X_7 in the previous step. Therefore, 8th bit of X_8 differs from X'_8 with probability almost 1. This means, $\Pr(X_8[8] = X'_8[8]) \approx 0$. On the other hand $X_4[8]$ is affected by the difference with probabilty ≈ 0.025. In their sum, neglecting the small probabilities of propagation from the previous bits, we can say $\Pr((X_8+X_4)[8] = (X'_8+X'_4)[8]) \approx 0.025$. Therefore the final XOR will produce equality with probability ≈ 0.025. ■

Theorem 6. *After the completion of the third round,* $\Pr(X_0[7] = X'_0[7]) \approx 0.94$.

Proof. In the third round, X_0 is updated by the function: $a = a \oplus ((c + d) \lll 18)$, where c and d are X_8 and X_{12} respectively which are already updated in the third round. Since the rotation is by 18 bits, the 7th bit of updated X_0 can be given by: $X_0[7] \oplus (X_8 + X_{12})[21]$. Now, $X_0 = X'_0$ upto the second round. So, the difference in the updated X_0 and X'_0 can be caused only by the difference of $(X_8 + X_{12})[21]$ and $(X'_8 + X'_{12})[21]$. The probability of the equality of these two bits can be found by partitioning it into two events: If there is no difference of the carry from previous bit, and if there is adifference in the carry. In the first case, the probability of equality can be found by Theorem 1, where $p = \Pr(X_{12}[21] = X'_{12}[21]) = 0.025$ (Theorem 5). This value is approximately 0.975.

Next we focus on $\Pr(X_8[21] = X'_8[21])$ to get q. In the second round, this bit is updated as $d = d \oplus ((b + c) \lll 13)$, where b is X_0 and c is X_4. Since X_0 is not influenced by the difference so far, we focus on 8th bit of X_4 (since rotation is by 13 bits). This bit has a direct influence of the difference given in 31st bit of X_7, which after rotation by 9 bits, reaches this position. So, this bit is different with probability

1. So, in the 2nd round, $\Pr(X_8[21] = X'_8[21]) = 0$. In the third round, one can check that the term $(b + c)$ has influence over this position with very negligible probability. Therefore, this probability q is approximately 0. In the second case, the probability can be found by Theorem 3, which gives the probability as approximately 0. Next we find the probability of the difference of the carry. Though there are 20 bits on the right side, we ignore the 19 bits of the right side since their influence is negligible and consider only the 20th bit. So, the probability of equality of the carries can be computed by Theorem 2, where $p = 0.996, q = 0.95$. Using the formula, we have the probability of equality of the carries approximately as 0.97. So, the total probability: $0.97 \times 0.975 + 0.03 \times 0 \approx 0.94$. ■

Second column: In this column, the tuple (a, b, c, d) is (X_1, X_5, X_9, X_{13}).

Theorem 7. *After third round,* $\Pr(X_1[14] = X'_1[14]) = 0.96$.

Proof. In third round, X_1 is updated by $d = d \oplus ((b + c) \lll 13)$. Here, upto the second round, X_1 (or d in the second round) is not influenced by the difference. b and c are respectively X_9 and X_{13} from the 3rd round. So, we focus on the 1st bit of their sum (since the rotation is by 13 bits). Now, X_{13} is updated in the 2nd round by $c = c \oplus ((a + b) \lll 9)$. In that update, on the 24th bit we focus, due to the rotation. Since the 19th bit contains a difference during its update (can be checked very easily), this bit is affected with probability $\frac{1}{2^5}$ (Lemma 1). So, the probability of equality is $1 - \frac{1}{2^5} \approx 0.97$. The differences in the addition $a + b$ in the third round do not bring any significant change in this probability. By similar argument one can very easily check that $\Pr(X_9[1] = X'_9[1]) \approx 0.99$. Therefore, the probability of the equality of 1st bit of $(X_9 + X_{13})$ can be given by $0.97 \times 0.99 \approx 0.96$. (here we ignore the probability of the difference in the carry from the previous bit, since it is negligible). Therefore, the final probability of the given event is also 0.96. ■

Fourth column:

Theorem 8. *After the third round,* $\Pr(X_3[7] = X_3[7]) = 0.75$.

Proof. $(a, b, c, d) = (X_{15}, X_3, X_7, X_{11})$. In the first step, $X[3]$ is updated by the function $b = b \oplus ((a + d) \lll 7))$. After the first round, the first row is not influenced by the difference. Therefore in the second round, the rowround operation on the first row also does not bring any difference. So, $X[3]$ and $X'[3]$ is exactly same after the second round. Also, from Lemma 3, we know that the 0th bit of X_{11} and X'_{11} are equal after the second round. Therefore the difference at 0th bit of $a + d$ occurs iff the 0th bit of a has the difference. Therefore, the probability of the equality in the least significant bit of $a + d$ for X and X' is equal to the difference at the least significant bit of a i.e., $\Pr(X_{15}[0] = X_{15}[0])$, which is 0.75, from Lemma 2. Therefore, this probability is 0.75. Now, after left rotation by 7 bits, this bit shifts to the 7th bit. And this is XORed with b, which does not have any difference. Therefore, the probability that the 7th bit of the updated b is equal for X_3 and X'_3 is 0.75. ■

Fourth Round:

Theorem 9. *After fourth round,* $\Pr(X_1[14] = X'_1[14]) \approx 0.56$.

Proof. In the rowround on the first row of X, we have: $(a, b, c, d) = (X_0, X_1, X_2, X_3)$. So, $b = X_1$ is updated at first. We focus on the 14th bit of X_1 and X_1'. Let us denote the 14th bit of b as $b[14]$. So $b[14] = b[14] \oplus ((a + d)[7])$ (since the 7th bit of $a + d$ is left shifted by 7 bits and reaches the 14th bit position).

Let us first focus on the 7th bit of $a + d$. The probability of the equality of this bit can be computed for two separate cases: If there is no difference in the carry from the previous bit, and if there is difference in the carry.

Event 1: If there is no difference in the carry, the probability of equality at $(a + d)[7]$ can be calculated by the formula derived from Theorem 1, which is $pq(1 - p)(1 - q)$. Here, $p = 0.94$ (Theorem 6). In the 3rd round, X_3 is updated by $b = b \oplus ((a+d) \lll 7)$. So, the 7th bit of X_3 can be given by XOR of the previous X_3 and 0th bit of X_{11} and X_{15}. Since the 0th bit of X_3 and X_{11} unaffected by the difference upto the second round, so the difference can only come due to the difference at X_{15}. Now, the 0th bit of X_{15} possesses equality with probability 0.75 after the first round (Lemma 2). It remains the same after the second round since in this rowround all the other key cells involved are not influenced by the difference. So the probability is $q = 0.75$. So the probability of the equality if there is difference in the carry is $(0.94 \times 0.75 + .06 \times 0.25) \approx 0.71$.

Event 2: If there is difference in the carry, the probability can be calculated by the formula from Theorem 3, which is $p(1 - q) + q(1 - p)$. So, the probability is: $(0.94 \times 0.25 + 0.06 \times 0.75) \approx 0.29$.

Now, we find the probabilities of event 1 and event 2. We experimentally observe that in d, the 6th bit does not possess any bias, i.e., $\Pr(X_3[6] = X_3'[6]) = \frac{1}{2}$. Therefore, as already mentioned in the beginning of this section, we do not attempt to prove this probability. Suppose we denote this probability by q_6. So, $q_6 = \frac{1}{2}$. Therefore we can use Corollary 1 here. By a similar argument as in the proof of $\Pr(X_3[7] = X_3'[7]) = 0.94$, it can be proved that $p_6 = \Pr(X_3[6] = X_3'[6]) = 0.9$. Therefore, the probability that there is difference in the carry is $\frac{1}{2} - \frac{0.9}{4} \approx 0.28$. And the probability that there is no difference in the carry is $1 - 0.28 = 0.72$. So, calculating the total probability, we get 0.59. Next, this bit is XORed with $b[14]$. The probability that $b[14]$ differs is 0.96, from Theorem 7. So, the final probability can be given by the formula $(pq + (1 - p)(1 - q))$ (Theorem 1), where $p = 0.96$ and $q = 0.59$. So, we have the probability approximately as 0.56. ■

In Table 1, we provide a comparison between the probabilities achieved theoretically and by experiment side-by-side in second column and third column. We perform experiment over 10,000 random key-IV pairs to produce this result.

Chosen IV Cryptanalysis: In [7], Maitra improved the bias for same input-output pair by suitably choosing the IVs. Using this idea, he improved the probability $\Pr(X_1[14] = X_1'[14])$ to 0.62. This probability can also be theoretically proved by our approach. The tracking of the biases will be exactly similar, the only difference being that the biases would be higher in this case. For example, in the first round, the probability $\Pr(X_{15}[0] = X_{15}'[0])$ would become 1, instead of 0.75 (Lemma 2). In Table 1, we provide the experimental probabilities based on 10,000 random key-IV pairs for chosen IV in the last column, which gives a clear idea about how the difference propagates.

5 Conclusion

Our work finds out the exact reason for an observed result, which has been exploited for one decade in the key recovery attacks against Salsa. This work provides a deep

Table 1. Comparison between theoretical and experimental results

Output bits	Theoretical result	Experimental result	Chosen IV Experimental result [7]
$X_{15}^{1}[0]$	0.75	0.75	1.0
$X_{4}^{3}[8]$	0.998	0.996	0.997
$X_{12}^{3}[21]$	0.025	0.022	0.021
$X_{0}^{3}[7]$	0.94	0.95	0.95
$X_{1}^{3}[14]$	0.96	0.98	0.98
$X_{3}^{3}[7]$	0.75	0.74	0.93
$X_{1}^{4}[14]$	0.56	0.56	0.62

insight of the distinguisher observed for Salsa. In the differential attack idea against Salsa, it has always been an aim for the researchers to increase the observed bias and if possible, extend it to next rounds. This theoretical explanation will help to get a clear picture of the propagation of the bias from the beginning, which may help to find some better distingisher by some suitable method.

References

1. J. P. Aumasson, S. Fischer, S. Khazaei, W. Meier and C. Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. FSE 2008, LNCS 5086, pp. 470–488, 2008.
2. D. J. Bernstein. Salsa20 specification. eSTREAM Project algorithm description, `http`//www.ecrypt.eu.org/stream/salsa20pf.html, 2005.
3. A. R. Choudhuri and S. Maitra. Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. IACR Trans. Symmetric Cryptol 2016(2) pp. 261–287, 2016. Available at `http://eprint.iacr.org/2016/1034`.
4. P. Crowley. Truncated differential cryptanalysis of five rounds of Salsa20. IACR 2005. Available at `http://eprint.iacr.org/2005/375`.
5. S. Dey and S. Sarkar. Improved analysis for reduced round Salsa and ChaCha. Discrete Applied Mathematics 227(2017) pp. 58-69, 2017.
6. S. Fischer, W. Meier, C. Berbain, J. F. Biasse. Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. Indocrypt 2006, LNCS 4329, pp. 2–16, 2006.
7. S. Maitra. Chosen IV Cryptanalysis on Reduced Round ChaCha and Salsa. Discrete Applied Mathematics, volume 208, pp. 88–97, 2016.
8. S. Maitra, G. Paul, W. Meier. Salsa20 Cryptanalysis: New Moves and Revisiting Old Styles. WCC 2015. Available at `http://eprint.iacr.org/2015/217`.
9. Z. Shi, B. Zhang, D. Feng, W. Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. ICISC 2012, LNCS 7839, pp. 337–351.
10. Y. Tsunoo, T. Saito, H. Kubo, T. Suzaki and H. Nakashima. Differential Cryptanalysis of Salsa20/8. SASC 2007. `http`//www.ecrypt.eu.org/stream/papersdir/2007/010.pdf