Revisited Diffusion Analysis of Salsa and ChaCha

Yusuke MATSUOKA and Atsuko MIYAJI
Graduate School of Engineering, Osaka University
2-1 Yamadaoka Suita Osaka, Japan

***Abstract*—Both ChaCha and AES are standardized as symmetric ciphers in TLS 1.3; AES is a block cipher, whereas ChaCha is a stream cipher. The security of AES has been studied by many researchers. ChaCha, however, needs more security analysis because it has been proposed more recently, compared with AES. Furthermore, ChaCha is improved from Salsa from the point of view of *diffusion* and thus, diffusion analysis of Salsa and ChaCha is important to understand their security-design criteria. In this study, we revisit diffusion analysis and investigate weak bits and weak columns of Salsa and ChaCha. To the authors' knowledge, this is the first detailed diffusion analysis of Salsa and ChaCha.**

I. Introduction

Salsa [2] and ChaCha [1] were proposed by Dan Bernstein. Recently, ChaCha has received much attention owing to the following reasons. ChaCha and AES are standardized as symmetric ciphers in TLS 1.3; AES is a block cipher, whereas ChaCha is a stream cipher. Furthermore, Google supports ChaCha because ChaCha can speed up and strengthen HTTPS connections for Chrome on Android without AES hardware acceleration [3]. The security of AES has been studied by many researchers. ChaCha20 was proposed by Dan Bernstein in 2008 after he proposed Salsa20 [2] in 2005. To date, few studies have analyzed the security [4], [5], [6], [7]. To attain a reasonable level of confidence regarding the security of ChaCha20, it is necessary to perform more security analyses.

Furthermore, ChaCha20 is improved from Salsa [2] to increase *diffusion*. Therefore, diffusion analysis of Salsa20 and ChaCha20 is important for understanding their security-design criteria. In [6], diffusion analysis on Salsa and ChaCha is performed. However, the results obtained did not deal with weak-bit and the 2nd-round diffusion, that is, bit-dependent and round-dependent diffusion, respectively.

In this study, we re-examine diffusion on Salsa and ChaCha Core in terms of bit-dependent and round-dependent diffusion. Our detailed analysis find that weak bits in the 1st round and weak columns in the 2nd round of Salsa and ChaCha when given one-bit different inputs.

This paper is organized as follows. Section II summarizes both Salsa20 and ChaCha20 and explains the previous studies on diffusion analysis [6]. Then, we revisit diffusion analysis in Section III. Experimental results and discussion are presented in Section IV. Finally, we present our conclusions in Section V.

II. Previous Results

In this section, we first describe the stream ciphers Salsa and ChaCha. Then, we describe the existing security analysis ([6]).

A. Salsa20 and ChaCha

Salsa20 [2] is a stream cipher that was developed in 2005 by D. J. Bernstein. This cipher generates pseudorandom numbers for performing operations based on 32-bit addition, XOR, and rotation. This operation is performed for 20 rounds that are repeated 10 times. The operation is performed on an input word, specifically, on the rows and columns of a 4×4 matrix. Salsa20/12 and Salsa20/8 are variants of Salsa20 with a reduced number of rounds; they use 12 and 8 rounds, respectively, compared to 20 in the case of Salsa20.

ChaCha 20 [1] is a variant of Salsa20 that was developed in 2008 by D. J. Bernstein. It aims to achieve improved performance by increasing the diffusion in each round. Corresponding to Salsa20/8, Salsa20/12, and Salsa20, ChaCha8, ChaCha12, and ChaCha20 have been developed. Salsa20 and ChaCha20 use different round-to-round operations and initial states. However, their encryption structure is the same; both use 32-bit addition, XOR, and rotation operations. Both of them have an initial state with 16 4-byte words comprising the secret key, block number, nonce, and constants. Each round consists of four quarter rounds of 4 words each.

The encryption is performed on 64 bytes of plaintext m to produce 64 bytes of ciphertext c.
$m_n : m_0, m_1, \cdots, m_k (m = m_0 \parallel m_1 \parallel \cdots \parallel m_k)$
$c_n : c_0, c_1, \cdots, c_k (c = c_0 \parallel c_1 \parallel \cdots \parallel c_k)$
Here, we outline the encryption procedure that is performed for 20 rounds.

Encryption procedure

1) Make the 64-byte initial state $X_n^{(0)}$ by using constants, a secret key, block number, and nonce.
2) After performing 20 rounds of the operation on the initial state to obtain the final state, perform 32-bit addition of the initial state $X_n^{(0)}$ and final state $X_n^{(20)}$ to yield the 64-byte output stream X_n^{out}.
3) Perform encryption (decryption) for every 64 bytes by XOR-ing the 64-byte plaintext m_n and X_n^{out}.
4) Obtain the ciphertext c by performing the procedure 1–3 for each n.

Salsa20 and ChaCha20 yield different initial states. In Salsa and ChaCha, the secret key, block number, nonce,

and constants are $k_j, i_j, v_j, t_j (j = 0, 1, \cdots, 7)$, respectively. Each $k_j, i_j, v_j, t_j (j = 0, 1, \cdots, 7)$ has 32-bit size. Salsa20 and ChaCha 20 have the same constant values as mentioned above: $t_0 = 0\text{x}6170786e, t_1 = 0\text{x}3320646e, t_2 = 0\text{x}79622d32, t_3 = 0\text{x}6b206574$. The block number i is counted up for each 64-byte encryption. The nonce is also updated for each 64-byte encryption. Initial state of Salsa:

$$X^{(0)} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} t_0 & k_0 & k_1 & k_2 \\ k_3 & t_1 & v_0 & v_1 \\ i_0 & i_1 & t_2 & k_4 \\ k_5 & k_6 & k_7 & t_3 \end{pmatrix}$$

Next, we describe how to update the state $X^{(r)}$. We call the operations of Salsa20 and ChaCha20 for every round of Salsa Core and ChaCha20, respectively. The round function `QuarterRD` operation includes four quarter round operations that each process 4 words. The round function in Salsa20 and ChaCha20 is denoted as $\texttt{QuarterRD}_{\texttt{Salsa}}$ and $\texttt{QuarterRD}_{\texttt{ChaCha}}$, respectively. Next, we discuss Salsa Core. By using 4 32-bit words as an input for the $\texttt{QuarterRD}_{\texttt{Salsa}}$ function, we obtain a 4-word output. Assuming that the input is x and output is y, the output y_j corresponding to each word x_j is generated by an arithmetic operation (ARX) performed by combining 32-bit addition, rotation, and XOR operations. In addition, the order of the input and output words is fixed. In order, we call the $\texttt{QuarterRD}_{\texttt{Salsa}}$ function for 16 words in the row direction of the 4×4 matrix as the input and output **row round** and call the $\texttt{QuarterRD}_{\texttt{Salsa}}$ function for 16 words in the column direction of the 4×4 matrix as the input and output **column round**. The input word is $x = (x_a, x_b, x_c, x_d)$, and the output word is $y = (y_a, y_b, y_c, y_d)$. Then, $y = \texttt{QuarterRD}_{\texttt{Salsa}}(x)$ operates according to the following formula.

$$\begin{aligned}(y_a, y_b, y_c, y_d) &= \texttt{QuarterRD}_{\texttt{Salsa}}(x_a, x_b, x_c, x_d) \\ &= \begin{cases} y_b &= x_b \oplus ((x_a + x_c) \lll 7) \\ y_c &= x_c \oplus ((y_b + x_a) \lll 9) \\ y_d &= x_d \oplus ((y_c + y_b) \lll 13) \\ y_a &= x_a \oplus ((y_c + y_d) \lll 18) \end{cases}\end{aligned}$$

Next, we discuss ChaCha Core. In ChaCha Core, as in Salsa Core, the 4-word output is generated by ARX using 4 words. The order of the input and output words in ChaCha Core differs from that in Salsa Core. A **diagonal round** is used instead of a **row round**. The input word is $x = (x_a, x_b, x_c, x_d)$, and the output word is $y = (y_a, y_b, y_c, y_d)$. Then, $y = \texttt{QuarterRD}_{\texttt{Salsa}}(x)$ operates according to the following formula.

$$\begin{aligned}(y_a, y_b, y_c, y_d) &= \texttt{QuarterRD}_{\texttt{ChaCha}}(x_a, x_b, x_c, x_d) \\ &= \begin{cases} u_a &= x_a + x_b \\ u_d &= x_d \oplus u_a; u_d = u_d \lll 16 \\ u_c &= x_c + u_d \\ u_b &= x_b \oplus u_c; u_b = u_c \lll 12 \\ y_0 &= u_0 + u_1 \\ y_d &= u_d \oplus y_a; y_d = y_d \lll 8 \\ y_c &= u_c + y_d \\ y_b &= u_b \oplus y_c; y_b = y_b \lll 7 \end{cases}\end{aligned}$$

TABLE I: Diffusion of Salsa [6]

IP \OP	a	b	c	d	
a	12.105	1.955	4.316	6.430	μ: 5.0992
b	7.255	1.0	1.946	4.330	σ: 3.1887
c	4.333	0	1.0	1.958	
d	8.784	1.958	2.468	5.649	

TABLE II: Diffusion of ChaCha [6]

IP \OP	a	b	c	d	
a	4.047	11.285	9.331	5.989	μ: 6.6424
b	5.987	13.418	10.825	7.794	σ: 3.2731
c	2.399	6.781	4.803	2.47	
d	2.399	8.528	6.751	3.399	

For the column and diagonal rounds, see [1].

B. Diffusion Property of Quarter Rounds

In [6], **diffusion** was used for Salsa Core and ChaCha Core to evaluate the safety of Salsa and ChaCha, respectively. Diffusion is considered one of the main properties of the operation of a secure cipher; it refers to the property of a function to quickly spread a small change in the input to the maximum possible number of bits in the output. In [6], diffusion was attributed to differences in one-bit of an output word and one-bit of an input word. Higher diffusion implies a better quarter round function. We describe these experiments [6].

The input word is $x = (x_a, x_b, x_c, x_d)$, and the output word is $y = (y_a, y_b, y_c, y_d)$ (each word contains 32 bits). Then, for all shifts (i, j, k, l), the **diffusion matrix** D is computed. (i, j, k, l) represents all shifts of the rotation operation in Salsa Core and ChaCha Core. Because $0 \le i, j, k, l \le 31$, the number of diffusion matrixes in each Core is $32 \times 32 \times 32 \times 32 = 2^{20}$. **Diffusion matrix** D is the matrix generated by each D_{vw}; this is the number of difference bits in the output word $y_w (w = a, b, c, d)$ caused by flipping one bit of an input word $x_v (v = a, b, c, d)$.

$$D = \begin{pmatrix} D_{aa} & D_{ab} & D_{ac} & D_{ad} \\ D_{ba} & D_{bb} & D_{bc} & D_{bd} \\ D_{ca} & D_{cb} & D_{cc} & D_{cd} \\ D_{da} & D_{db} & D_{dc} & D_{dd} \end{pmatrix}$$

Experiments are performed to obtain D for each (i, j, k, l). In Salsa Core and ChaCha Core, (μ, σ), that is, the mean and standard deviation , respectively, of the diffusion matrix D of each (i, j, k, l) are shown in Tables I and II.

III. Detailed Diffusion Analysis

The results of a previous study [6] compute how many output bits y differ for y', where y and y' are outputs for inputs x and x', respectively. Here, a one-bit difference exists between x and x'. However, their results did not deal with bit-dependent and round-dependent diffusion. In this research, we re-examine diffusion on `Salsa` and `ChaCha` Core in terms of bit-dependent and round-dependent diffusion, which implies weak-bit and the 2nd-round diffusion, respectively.

A. Bit-Dependent Diffusion

When we compute the diffusion for 1 word, there are 32 one-bit differences. Diffusion is exactly dependent on an address where a one-bit difference exists. This is because both Salsa and ChaCha are ARX-based ciphers and thus non linearity introduced by a modular addition would cause differences in diffusion. Unfortunately, diffusions are evaluated over a random one-bit difference of a 4-word input (x_a, x_b, x_c, x_d) in [6]. It is important to compute diffusions on each 32 one-bit differences for 4 columns. However, a straightforward exhaustive examination would require a large number of trials. Therefore, we evaluate diffusion for each one-bit difference of input by the following two procedures:

1) **weak-bit search** For a given i-th bit difference of each 4-word input (x_a, x_b, x_c, x_d), compute the average of each diffusion matrix, and determine a set of weak bits $S_W = \{i\}$ that have a diffusion smaller than that of other bits.
2) **column-dependent weak-bit diffusions** In the previous procedure, the *average* of diffusions of each 4-word input with the same i-th bit difference is computed. In other words, we have investigated conditions dependent to column-dependent differences. In this procedure, for each weak bit $i \in S_W$, compute a diffusion matrix $D_{v,w}(v, w = a, b, c, d)$ individually and analyze behaviors dependent on each column for each weak bit.

B. Round-Dependent Diffusion

It is reported that positions of input and output such as $\{(b,b),(c,b),(c,c)\}$ and $\{(c,a),(c,d),(d,a)\}$ shows rather small diffusions in the first round for Salsa and ChaCha, respectively [6], as seen from Tables I and II. Furthermore, it is necessary to compute the second-round diffusion to estimate the minimum round to achieve enough diffusion. However, it is not clear whether the same position of output word in the second-round as that in the first round showss the low diffusion or not. The previous work evaluates only the first round diffusion. In fact, it is easy to compute the first-round diffusion because we need to focus on only 1 column, that is, an input and an output of 1 `QuarterRD`-function. However, it is rather complicated to compute the second-round diffusion: Salsa and ChaCha consist of 4 columns, A, B, C, and D, containing 4 words each, and every 4 columns are permutated each other after execution of `QuarterRD`-functions.

Considering the above issues, we evaluate the second-round diffusion as follows.

1) Set an input $(x_A, x_B.x_C, x_D)$ for 4 `QuarterRD`.
2) Flip one bit for the first-column input x_A and set the 1-bit difference input to $(\overline{x}_A, x_B.x_C, x_D)$
3) Execute `QuarterRD`-functions until the second round.
4) Evaluate diffusions for each 4-column output. Note that the outputs in the 4 columns of $(x_A, x_B.x_C, x_D)$ are different from each other. Thus, we need to evaluate each 4-column input and output.

From the above 4), we compute each diffusion matrix for every 4-column input. As a result, this examination aims to search for weak columns which show low diffusion compared with other columns.

IV. Experiments and Security Analysis

A. Experimental Conditions and Algorithms

In this chapter, we begin by explaining the experimental conditions and then show Algorithms 1 and 2, which correspond to Sections III-A and III-B. All the experiments were executed on an Intel(R) Core(TM) i3-4160 CPU@3.60GHz. Algorithms 1 and 2 compute the diffusion for only the original rotations of $(7, 9, 13, 18)$ and $(16, 12, 8, 7)$ for both Salsa and ChaCha, respectively. The number of trials was 10^6 and thus the diffusion matrix was computed as the average of 10^6 trials. Experimental times for Algorithms 1 over Salsa (resp. Chacha) and 2 over Salsa (resp. ChaCha) are 210.53 (resp. 213.70) and 63.94 (resp. 65.74) seconds.

Let us present experimental Algorithm 1 in the below.

Algorithm 1 Bit Dependency Examination

1: Let an M-sequence input be $x = (x_a, x_b, x_c, x_d)$.
2: Compute $y = (y_a, y_b, y_c, y_d) = \texttt{QuarterRD}(x)$.
3: **for all** $v \in \{0, 1, 2, \cdots, 31\}$ **do**
4: Set the v-th bit complement of x_a to $\overline{x}_a$.
5: Execute $y' = (y'_a, y'_b, y'_c, y'_d) = \texttt{QuarterRD}(\overline{x}_a, x_b, x_c, x_d)$.
6: Compute $D_a = (D_{aa}, D_{ab}, D_{ac}, D_{ad}) = (\mathrm{HW}(y_a \oplus y'_a), \mathrm{HW}(y_b \oplus y'_b), \mathrm{HW}(y_c \oplus y'_c), \mathrm{HW}(y_d \oplus y'_d))$.
7: Repeat steps 3 to 6 for x_b, x_c, and x_d, and get $D = (D_a, D_b, D_c, D_d)^t$
8: Go to step 1 for the next x (10^6 trials).
9: Compute an average of $D = (D_a, D_b, D_c, D_d)^t$ for v.
10: **end for**

In Algorithms 2, we need to compute diffusion matrixes for each columns from A to D, which is represented by D^A, D^B, D^C, D^D, where:

$$D^A = \begin{pmatrix} D^A_{aa} & D^A_{ab} & D^A_{ac} & D^A_{ad} \\ D^A_{ba} & D^A_{bb} & D^A_{bc} & D^A_{bd} \\ D^A_{ca} & D^A_{cb} & D^A_{cc} & D^A_{cd} \\ D^A_{da} & D^A_{db} & D^A_{dc} & D^A_{dd} \end{pmatrix}.$$

D^B, D^C, and D^D are defined in the same way as D^A. Algorithms 2 is given as follows.

B. Security Analysis

In this chapter, we describe the experimental results obtained with Algorithms 1 and 2 and revisit diffusion analysis. Figures 1 and 2 show the results obtained for `Salsa` and `ChaCha` with Algorithms 1, respectively. The vertical axis shows the average for D_{vw} ($2^5 \times 10^6$), while the horizontal axis shows the address bit with a one-bit difference from MSB= 31 to LSB= 0. Each line shows the average for all the diffusion matrixes, that being 2.195 and 3.452 in Figures 1 and 2, respectively. From these results, we can determine each set of weak bits, $S_W^{\texttt{Salsa}} = \{21, 22, 29, 30, 31\}$ and $S_W^{\texttt{ChaCha}} = \{14, 15, 29, 30, 31\}$.

Algorithm 2 2nd-Round Examination

1: Let an M-sequence input be $x=(x_a, x_b, x_c, x_d)$.
2: Compute $y=(y_a, y_b, y_c, y_d)=\texttt{QuarterRD}(x)$.
3: Set a random one-bit complement of x_a to $\overline{x}_a$.
4: Execute $y'=(y'_a, y'_b, y'_c, y'_d)=\texttt{QuarterRD}(\overline{x}_a, x_b, x_c, x_d)$.
5: Set $(D_{aa}, D_{ab}, D_{ac}, D_{ad}) = (\mathrm{HW}(y_a \oplus y'_a), \mathrm{HW}(y_b \oplus y'_b), \mathrm{HW}(y_c \oplus y'_c), \mathrm{HW}(y_d \oplus y'_d))$.
6: Execute $z=(z_a, z_b, z_c, z_d)=\texttt{QuarterRD}(y_a, y_b, y_c, y_d)$, $z^A=(z^A_a, z^A_b, z^A_c, z^A_d)=\texttt{QuarterRD}(y'_a, y_d, y_c, y_b)$, $z^B=(z^B_a, z^B_b, z^B_c, z^B_d)=\texttt{QuarterRD}(y_a, y_d, y_c, y'_b)$, $z^C=(z^C_a, z^C_b, z^C_c, z^C_d)=\texttt{QuarterRD}(y_a, y_d, y'_c, y_b)$, $z^D=(z^D_a, z^D_b, z^D_c, z^D_d)=\texttt{QuarterRD}(y_a, y'_d, y_c, y_b)$).
7: Compute $((D^A_{aa}, D^A_{ab}, D^A_{ac}, D^A_{ad}) = (\mathrm{HW}(z_a \oplus z^A_a), \mathrm{HW}(z_b \oplus z^A_b), \mathrm{HW}(z_c \oplus z^A_c), \mathrm{HW}(z_d \oplus z^A_d)))$.
8: Repeat steps 3 to 6 for x_b, x_c, and x_d, and get $D_b, (D^A_b, D^B_b, D^C_b, D^D_b)$, $D_c, (D^A_c, D^B_c, D^C_c, D^D_c)$, $D_d, (D^A_d, D^B_d, D^C_d, D^D_d)$.
9: Go to step 1 for the next x (10^6 trials).
10: Compute an average of D^A, D^B, D^D, and D^D.

We went on to analyze the behaviors of $S^{\texttt{Salsa}}_W$ in detail. For $S^{\texttt{ChaCha}}_W$, we will show the results obtained with our final version due to space restrictions. Figure 3 shows the diffusion average for $S^{\texttt{Salsa}}_W$ for each input a, b, c, d. Figures 4 and 5 show the diffusion averages in $S^{\texttt{Salsa}}_W$ for outputs a, b, c, d for a 1-bit difference in input b and input c, respectively. From these results, we acquired the 31st bit difference for each input a, b, c, and d outputs a significantly small diffusion as would be expected from the features of addition. In particular, there are major differences between the 31st bit and other weak bits in the case of (input, output)= $(b, c), (c, d)$, that is $D_{bc} = D_{cd} = 1$, while the bits of outputs 21, 22, 29, and 30 are greater than 1. We should note that, in addition to 29, 30, and 31, bits 21 and 22 are also weak bits.

For the 2nd-round diffusion in Algorithm III-B, we need to analyze the four columns A, B, C, and D independently. Here we focus on `Salsa`. A detailed analysis of `ChaCha` in our final version is given by showing only the average diffusion for all the columns in Table VII, given the space restrictions. Figures III, IV, V, and VI show each diffusion of 1st-/2nd-round and increasing ratio in each column A, B, C, and D, respectively.

Table IV shows that $(D^B_{ca}, D^B_{cb}, D^B_{cc}, D^B_{cd}) = (0, 0, 0, 0)$. This means that the 2nd-round diffusion in column B for the 1st-round one-bit difference c has disappeared. We investigated why the 2nd-round diffusion disappears. By using the result obtained for the 1st-round diffusion (Table I), we get $D_{cb} = 0$. By combining the fact that an input to column B in the 2nd round is equal to (y_a, y_d, y_c, y'_b) to $D_{cb} = 0$, we get $y'_b = y_b$. Therefore, both inputs become equal to each other. Thus, we can get $(D^B_{ca}, D^B_{cb}, D^B_{cc}, D^B_{cd}) = (0, 0, 0, 0)$. In fact, our results reflect the previous result that stated that high bias exists in output y_b in [5].

Table V shows that $(D^C_{ab}, D^C_{bb}, D^C_{cb}, D^C_{db}) = (0, 0, 0, 0)$. This implies that, for the 2nd-round diffusion in column C on output b, any 1st-round one-bit difference disappears. We considered why this had happened. The input of the column C is (y_a, y_d, y'_c, y_b) and, thus, D^c depends only on $(D_{ac}, D_{bc}, D_{cc}, D_{dc})$ as induced by differences in y_c and y'_c. Table I shows that $D_{ab}, D_{bb}, D_{cb}, D_{db}$ varies from 1 to 4.3 on average. From this fact, we deduce that output z_b of column C in the 2nd round is independent of the input difference in the 1st round.

In summary, `Salsa` still has a high bias on some columns even in the 2nd round, that is column-dependent bias exists for given one-bit difference inputs.

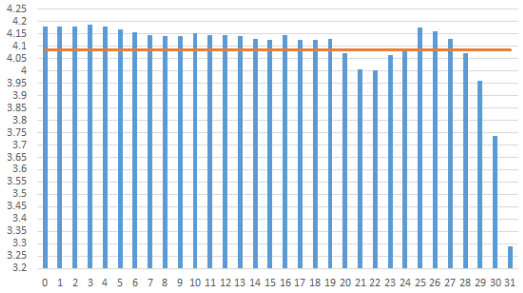

Fig. 1: Input-dependency diffusion (`Salsa`)

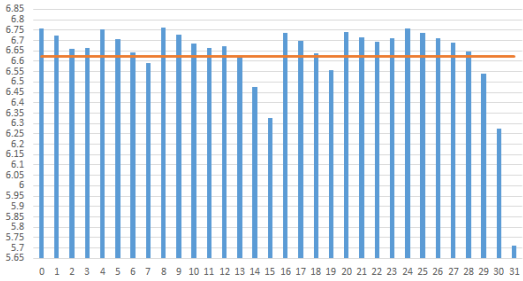

Fig. 2: Input-dependency diffusion (`ChaCha`)

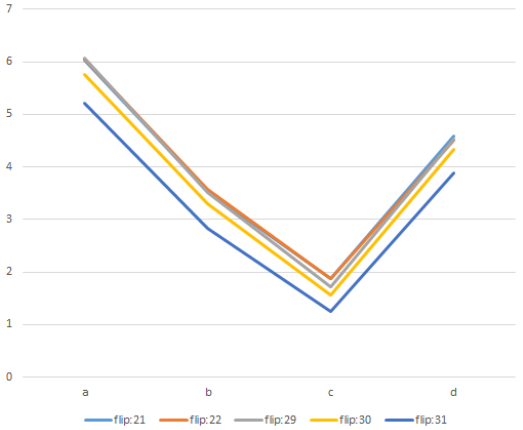

Fig. 3: Diffusion average for inputs a, b, c, d ($S^{\texttt{Salsa}}_W$)

TABLE III: 1/2-round Diffusions in Column A (`Salsa`)

IP\OP	a	b	c	d
a	12.324/15.917 (1.292)	2.018/13.134 (6.508)	4.414/15.617 (3.538)	6.417/15.831 (2.467)
b	7.208/15.882 (2.203)	1/8.089 (8.089)	1.969/13.381 (6.796)	4.422/14.367 (3.249)
c	4.453/15.958 (3.584)	0/5.282 (-)	1/10.799 (10.799)	1.982/13.271 (6.696)
d	8.937/16.007 (1.791)	1.988/9.435 (4.746)	2.424/14.707 (6.067)	5.691/15.370 (2.701)

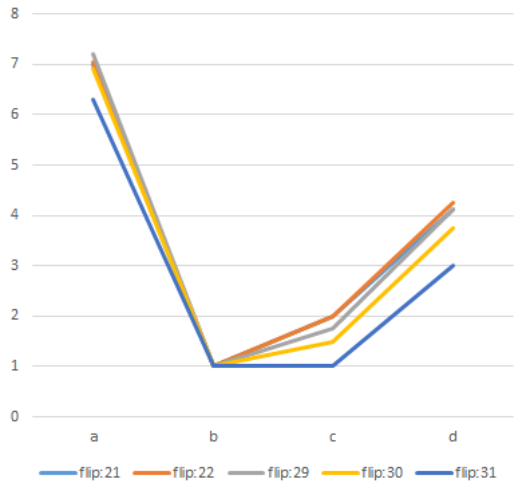

Fig. 4: Diffusion on outputs a, b, c, d for 1-bit-difference input b

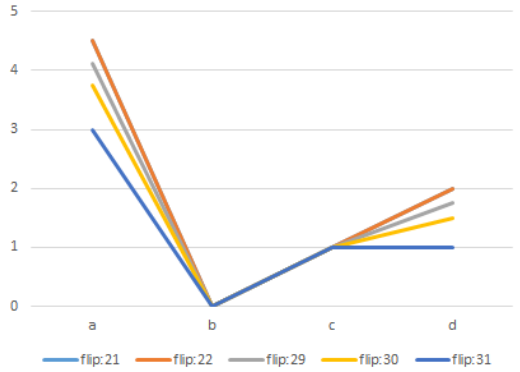

Fig. 5: Diffusion on outputs a, b, c, d for 1-bit-difference input c

TABLE IV: 1/2-round Diffusions in Column B (`Salsa`)

IP\OP	a	b	c	d
a	12.324/9.498 (0.771)	2.018/2.433 (1.206)	4.414/2.865 (0.649)	6.417/6.578 (1.025)
b	7.208/8.727 (1.211)	1/1.956 (1.956)	1.969/2.435 (1.237)	4.422/5.636 (1.275)
c	4.453/0 (0)	0/0 (-)	1/0 (0)	1.982/0 (0)
d	8.937/9.502 (1.063)	1.988/2.433 (1.224)	2.424/2.867 (1.1829)	5.691/6.580 (1.156)

TABLE V: 1/2-round Diffusions in Column C (`Salsa`)

IP\OP	a	b	c	d
a	12.324/10.811 (0.877)	2.018/0 (0)	4.414/4.362 (0.988)	6.417/5.215 (0.8123)
b	7.208/5.302 (0.736)	1/0 (0)	1.969/1.957 (0.994)	4.422/2.436 (0.551)
c	4.453/4.386 (0.985)	0/0 (-)	1/1 (1)	1.982/1.955 (0.987)
d	8.937/6.147 (0.688)	1.988/0 (0)	2.424/2.435 (1.005)	5.691/2.868 (0.504)

TABLE VI: 1/2-round Diffusions in Column D (`Salsa`)

IP\OP	a	b	c	d
a	12.324/14.398 (1.168)	2.018/6.427 (3.185)	4.414/7.047 (1.597)	6.417/10.607 (0.911)
b	7.208/13.004 (1.804)	1/4.384 (4.385)	1.969/5.274 (2.679)	4.422/7.941 (1.796)
c	4.453/8.133 (1.826)	0/1.957 (-)	1/2.433 (2.433)	1.982/5.280 (2.664)
d	8.937/14.856 (1.662)	1.988/5.634 (2.834)	2.424/6.679 (2.756)	5.691/9.942 (1.747)

TABLE VII: 2-round diffusion (`ChaCha`)

IP \OP	a	b	c	d
a	15.814	15.980	16.031	15.907
b	15.996	15.970	15.975	16.004
c	15.60	16.091	15.764	15.356
d	15.578	15.928	15.941	14.935

V. Conclusion

. This study has reanalyzed the diffusion of `Salsa` and `ChaCha` from the viewpoint of bit dependency and round dependency. Our experiments clarified each set of weak bits, namely, $S_W^{\texttt{Salsa}} = \{21, 22, 29, 30, 31\}$ and $S_W^{\texttt{ChaCha}} = \{14, 15, 29, 30, 31\}$ for `Salsa` and `ChaCha`, respectively. For second-round diffusion, we found strong biases in that the second-round diffusion for column B for the first-round one-bit difference c disappeared; and the second-round diffusion in column C on input b for any first-round one-bit difference disappeared. Furthermore, we have theoretically shown that the second-round diffusion in column B for the 1st-round one-bit difference c disappeared.

Acknowledgment

This work is partially supported by JSPS KAKENHI Grant (C)(JP15K00183), Microsoft Research Asia, CREST(JPMJCR1404) at Japan Science and Technology Agency, the Japan-Taiwan Collaborative Research Program at Japan Science and Technology Agency, and Project for Establishing a Nationwide Practical Education Network for IT Human Resources Development, Education Network for Practical Information Technologies.

References

[1] Daniel J. Bernstein. Chacha, a variant of salsa20, 2008.
[2] Daniel J. Bernstein. The salsa20 family of stream ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*, pages 84–97, 2008.
[3] Bursztein. E.: Speeding up and strengthening https connections for chrome on anddroid . tech. rep. (april 2014). https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html.
[4] Arka Rai Choudhuri and Subhamoy Maitra. Differential cryptanalysis of salsa and chacha – an evaluation with a hybrid model. Cryptology ePrint Archive, Report 2016/377, 2016. https://eprint.iacr.org/2016/377.
[5] Subhamoy Maitra. Chosen iv cryptanalysis on reduced round chacha and salsa. In *IACR Cryptology ePrint Archive*, 2015. http://eprint.iacr.org/2015/698.
[6] Rajeev Sobti and Geetha Ganesan. Analysis of quarter rounds of salsa and chacha core and proposal of an alternative design to maximize diffusion. In *Indian Journal of Science and Technology*, volume 9(3), jan 2016.
[7] Rajeev Sobti and G. Geetha. A comparison of diffusion properties of salsa, chacha, and mcc core.