Modified version of "Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha"

Tsukasa Ishiguro

KDDI R&D Laboratories Inc.
2-1-15 Ohara, Fujimino, Saitama 356-8502, Japan
tsukasa@kddilabs.jp

1 Contents of modification

This paper is a modified version of own paper "Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha" presented in ICICS2011. In the original paper, there are incorrect data because of software bug in the experimental program. Therefore, we conducted with a correct program. Additionally, we modified the algorithm with a view to improvement of analysis precision.

Table1 shows the maximum values per round of Salsa20 and ChaCha, and output of Mersenne Twister as PRF. This table show there are no double bit biases of 5 or more round Salsa20 and ChaCha (see Section 4). Therefore, these ciphers are not presently under threat.

Table 1. Double bit differentials of Salsa20

Type	Round	Key length	$[\Delta_i^0]_j$	$[\Delta_p^r]_q$	$[\Delta_s^r]_t$	$\lvert\widetilde{\varepsilon_d}\rvert$
Salsa20	4	256	$[\Delta_8^0]_{17}$	$[\Delta_6^4]_{23}$	$[\Delta_7^4]_0$	0.177887
Salsa20	5	256	-	-	-	Not Found
Salsa20	6	256	-	-	-	Not Found
Salsa20	7	256	-	-	-	Not Found
Salsa20	8	256	-	-	-	Not Found
Salsa20	9	256	-	-	-	Not Found
ChaCha	4	256	-	-	-	Not Found
ChaCha	5	256	-	-	-	Not Found
ChaCha	6	256	-	-	-	Not Found
ChaCha	7	256	-	-	-	Not Found
ChaCha	8	256	-	-	-	Not Found

Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha

Tsukasa Ishiguro, Shinsaku Kiyomoto, and Yutaka Miyake

KDDI R&D Laboratories Inc.
2-1-15 Ohara, Fujimino, Saitama 356-8502, Japan
{tsukasa,kiyomoto,miyake}@kddilabs.jp

Abstract. In this paper, we improve an analysis algorithm and apply it to cryptanalysis of Salsa and ChaCha. We constructed a distinguisher of double-bit differentials to improve Aumasson's single-bit differential cryptanalysis. This method has potential to apply to a wide range of stream ciphers; a double-bit correlation would be found in case that no single-bit correlation is found. However, there are no double bit biases of 5 or more round Salsa20 and ChaCha.

Keywords: Stream cipher, Salsa20, ChaCha, eSTREAM

1 Introduction

Efficient implementations of stream ciphers are useful in any application which requires high-speed encryption, such as SSL[13] and WEP[21]. The stream cipher project of ECRYPT(eSTREAM)[11] was launched to identify new stream ciphers that realizes secure and high-speed encryption. This project ended with a proposal of a list of new eight algorithms in 2008, and one was removed from the list in 2009[2] due to a new vulnerability of the cipher. Four algorithms are assumed to apply to software implementations, and remaining three are for lightweight hardware implementations.

Salsa20, one of algorithms for software implementations, was proposed by Bernstein[5] in 2005, and the cipher is the finalist of the eSTREAM. Salsa20 offers a simple, clean, and scalable design and is suitable for software implementations. Bernstein advocated use of 8, 12 and 20 round versions of Salsa20. However, in eSTREAM, the 12-round version was adopted due to the balance, combining a very nice performance profile with what appears to be a comfortable margin for security.

More recently, he has proposed the ChaCha[3], a variant of the Salsa20 family. ChaCha follows the same design principles as Salsa20, and a difference between Salsa20 family and ChaCha is the core function; the core function of ChaCha realizes faster diffusion than that of Salsa20 family. ChaCha achieves faster software speed than Salsa20 in some platforms.

Related work. There are many ciphers proposed in eSTREAM, and some have been broken by distinguishing attacks. NLS proposed by Hawkes et al[14], is an extended version of SOBER[20]. NLS is a software-oriented cipher based on simple 32-bit operations (such as 32-bit XOR and addition modulo 2^{32}), and is related to small fixed arrays. This stream cipher was broken by a distinguishing attack[8] and a Crossword Puzzle Attack[7] which is a variant of the distinguishing attack. LEX[6] has a simple design and based on AES. A variant of the distinguishing attack[10] was found on LEX. Yamb[17] is a synchronous encryption algorithm that allows keys of any length in the range 80-256 bits and allows initial vectors IV of any length in the range 32-128 bits. Yamb was broken by a distinguishing attack proposed by Wu et al.[24]. Some other stream ciphers have been broken by distinguishing attacks[16, 22].

Some independent cryptanalyses on Salsa20 have been published, to report key-recovery attacks for its reduced versions with up to 8 rounds, while Salsa20 has a total of 20 rounds. Previous attacks on Salsa20 used a distinguishing attack exploiting a truncated differential over 3 or 4 rounds. The first attack was presented by Crowley[9], and it was claimed that an adversary could break the 5-round version of Salsa20 within 3^{165} trials using a 256-bit key. Later, a four round differential was exploited by Fischer et al.[12] to break 6 rounds in 2^{177} trials and by Tsnunoo et al.[23] to break 7 rounds in about 2^{190} trials.

The best attack is proposed by Aumasson et al.[15] so far, and it covers the 8-round version of Salsa20 with an estimated complexity of 2^{251}. Regarding the 128-bit key, Aumasson proposed key-recovery attacks for reduced versions with up to 7 rounds[15]. Priemuth-Schmid proposed a distinguishing attack using slid pairs[19], but Bernstein showed that time complexity of the attack is higher than brute force attack[4].

For ChaCha, Aumasson attacked the 6-round version with an estimated complexity of 2^{139} and the 7-round version with an estimated complexity of 2^{248} using a 256-bit key. Regarding the 128-bit key, Aumasson proposed key-recovery attacks for reduced versions with up to 7 rounds with an estimated complexity of 2^{107}[15].

These attacks are single-bit differential attacks, a type of correlation attacks. In this method, an adversary chooses the input pair X, X' and observes the output pair Z, Z', where there is a differential in one bit between X and X'. Then, the adversary collects many output pairs by changing input pair and observes the one bit differential from the output pair. If the position of the input differential correlates strongly with the position of output differential, the adversary could distinguish real keystream from a random bit stream. Additionally, it was indicated a strong correlation from his experimental results.

Contribution. In this paper, we improve an analysis algorithm and apply it to cryptanalysis of Salsa and ChaCha. We construct a distinguisher using double-bit differentials to improve Aumasson's method, called single-bit differential cryptanalysis[1]. In our attack, the adversary chooses the input pair X, X' with a one-bit differential in the same way for a single-bit differential. Then, the adversary collects many output pairs by changing the input pair and observing the double-bit difference from the output pair. Finally, the adversary observes a correlation of the double-bit of the output pair and distinguishes keystream from the random bits. We searched correlations to compute 2-3 days using a PC, and can not find strong correlations 5 or more round Salsa20 and ChaCha.

2 Latin Dances

In this section, we describe the specifications of Salsa20[5] and ChaCha[3].

2.1 Salsa20

Algorithm 1 shows Salsa20 algorithm. The stream cipher Salsa20 operates on 32-bit words, takes as input a 256-bit key $k = (k_0, k_1, ..., k_7)$ or 128-bit key $k = (k_0, k_1, ..., k_3)$ and a 64-bit nonce $v = (v_0, v_1)$, and produces a sequence of 512-bit keystream blocks. The i-th block is the output of the Salsa20 function that takes as input the key, the nonce, and a 64-bit counter $t = (t_0, t_1)$ corresponding to the integer i. This function acts on the 4×4 matrix of 32-bit words written as:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} \tau_0 & k_0 & k_1 & k_2 \\ k_3 & \tau_1 & v_0 & v_1 \\ i_0 & i_1 & \tau_2 & k_4 \\ k_5 & k_6 & k_7 & \tau_3 \end{pmatrix} \text{or,}$$

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} \sigma_0 & k'_0 & k'_1 & k'_2 \\ k'_3 & \sigma_1 & v_0 & v_1 \\ i_0 & i_1 & \sigma_2 & k'_0 \\ k'_1 & k'_2 & k'_3 & \sigma_3 \end{pmatrix},$$

where σ and τ are constants dependent on the key length.

Then a keystream block Z is defined as:

$$Z = X + X^{20},$$

where $X^r = Round^r(X)$ with the round function of Salsa20 and $+$ is word-wise addition modulo 2^{32}. If $Z = X + X^r$, it is called "r-round Salsa20". A round function is called a doubleround function, and it consists of a columnround function followed by a rowround function. The doubleround function of Salsa20 is repeated 10 times. A vector (x_0, x_1, x_2, x_3) of four words is transformed into (z_0, z_1, z_2, z_3) by calculating as:

$$\begin{aligned} z_1 &= x_1 \oplus ((x_0 + x_3) \lll 7) \\ z_2 &= x_2 \oplus ((z_1 + x_0) \lll 9) \\ z_3 &= x_3 \oplus ((z_2 + z_1) \lll 13) \\ z_0 &= x_0 \oplus ((z_3 + z_2) \lll 18) \end{aligned}$$

This nonlinear operation is called a quarterround function and it is a basic part of the columnround function where it is applied to columns (x_0, x_4, x_8, x_{12}), (x_5, x_9, x_{13}, x_1), $(x_{10}, y_{14}, y_2, y_6)$ and $(y_{15}, y_3, y_7, y_{11})$, and then rowround function transforms rows (x_0, x_1, x_2, x_3), (x_4, x_5, x_6, x_7), $(x_8, x_9, x_{10}, x_{11})$, and $(x_{12}, x_{13}, x_{14}, x_{15})$.

Algorithm 1 Algorithm of Salsa20

```
INPUT: Initial matrix $X$, $r \in \mathbb{N}$
OUTPUT: $Z = X + X^r$
 1: $X' \leftarrow X$
 2: for $l = 0$ up to $\frac{r}{2}$ do
 3:     $(x'_0, x'_1, x'_2, x'_3) \leftarrow quarterround(x'_0, x'_1, x'_2, x'_3)$  /* 3-6:Columnround */
 4:     $(x'_5, x'_6, x'_7, x'_4) \leftarrow quarterround(x'_5, x'_6, x'_7, x'_4)$
 5:     $(x'_{10}, x'_{11}, x'_8, x'_9) \leftarrow quarterround(x'_{10}, x'_{11}, x'_8, x'_9)$
 6:     $(x'_{15}, x'_{12}, x'_{13}, x'_{14}) \leftarrow quarterround(x'_{15}, x'_{12}, x'_{13}, x'_{14})$
 7:     $(x'_0, x'_4, x'_8, x'_{12}) \leftarrow quarterround(x'_0, x'_4, x'_8, x'_{12})$  /* 7-10:Rowround */
 8:     $(x'_5, x'_9, x'_{13}, x'_1) \leftarrow quarterround(x'_5, x'_9, x'_{13}, x'_1)$
 9:     $(x'_{10}, x'_{14}, x'_2, x'_6) \leftarrow quarterround(x'_{10}, x'_{14}, x'_2, x'_6)$
10:     $(x'_{15}, x'_{[3}, x'_7, x'_{11}) \leftarrow quarterround(x'_{15}, x'_{[3}, x'_7, x'_{11})$
11: end for
12: return $X + X'$
```

2.2 ChaCha

Algorithm 2 shows ChaCha algorithm. ChaCha is similar to Salsa20 except the following two points. First, the composition of the quarterround function is defined as below.

$$\begin{aligned}
z_0 &= z_0 + z_1, \quad z_3 = z_3 \oplus z_0, \quad z_3 = z_3 \lll 16,\\
z_2 &= z_2 + z_3, \quad z_1 = z_1 \oplus z_2, \quad z_1 = z_1 \lll 12,\\
z_0 &= z_0 + z_1, \quad z_3 = z_3 \oplus z_0, \quad z_3 = z_3 \lll 8,\\
z_2 &= z_2 + z_3, \quad z_1 = z_1 \oplus z_2, \quad z_1 = z_1 \lll 7
\end{aligned}$$

Second, the composition of the initial matrix defined as below.

$$X = \begin{pmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \sigma_3 \\ k'_0 & k'_1 & k'_2 & k'_3 \\ k'_0 & k'_1 & k'_2 & k'_3 \\ v_0 & v_1 & i_0 & i_1 \end{pmatrix}, \text{or} \begin{pmatrix} \tau_0 & \tau_1 & \tau_2 & \tau_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ v_0 & v_1 & i_0 & i_1 \end{pmatrix}$$

3 Attack on Latin Dances

In this section, we discuss a distinguishing attack on Salsa20 and ChaCha. First, we define the *semi-regular distinguisher* and explain construction of the distinguisher. Next, we propose a distinguishing attack using double-bit differentials. Finally, we analyze the attack based on experimental results using a PC and estimate the number of keystream bits required for the attack and time complexity of the attack.

3.1 Types of distinguisher

Three types of a distinguisher are known[18] as below.

Algorithm 2 Algorithm of ChaCha

```
INPUT: Initial matrix $X$, $r \in \mathbb{N}$
OUTPUT: $Z = X + X^r$
 1: $X' \leftarrow X$
 2: for $l = 0$ up to $\frac{r}{2}$ do
 3:     $(x'_0, x'_4, x'_8, x'_{12}) \leftarrow quarterround(x'_0, x'_4, x'_8, x'_{12})$   /* 3-6:Columnround */
 4:     $(x'_1, x'_5, x'_9, x'_{13}) \leftarrow quarterround(x'_1, x'_5, x'_9, x'_{13})$
 5:     $(x'_2, x'_6, x'_{10}, x'_{14}) \leftarrow quarterround(x'_2, x'_6, x'_{10}, x'_{14})$
 6:     $(x'_3, x'_7, x'_{11}, x'_{15}) \leftarrow quarterround(x'_3, x'_7, x'_{11}, x'_{15})$
 7:     $(x'_0, x'_5, x'_{10}, x'_{15}) \leftarrow quarterround(x'_0, x'_5, x'_{10}, x'_{15})$   /* 7-10:Rowround */
 8:     $(x'_1, x'_6, x'_{11}, x'_{12}) \leftarrow quarterround(x'_1, x'_6, x'_{11}, x'_{12})$
 9:     $(x'_2, x'_7, x'_8, x'_{13}) \leftarrow quarterround(x'_2, x'_7, x'_8, x'_{13})$
10:     $(x'_3, x'_4, x'_9, x'_{14}) \leftarrow quarterround(x'_3, x'_4, x'_9, x'_{14})$
11: end for
12: return $X + X'$
```

1. *Regular Distinguisher.*
   The adversary selects a single key/IV randomly and produces keystream bits, seeded by the chosen key/IV, which is long enough to distinguish it from a random bit stream with a high probability.

2. *Prefix Distinguisher.*
   The adversary uses many randomly chosen key/IV's rather than a single key and a few specified bytes from each of the keystream bits generated by those key/IV's.

3. *Hybrid Distinguisher.*
   The adversary uses many key/IV's and for each key/IV the adversary collects long keystream bits.

In this paper, we define the *Semi-regular Distinguisher* as follows;

Semi-regular Distinguisher. An adversary uses a single random key and enough randomly chosen IVs to distinguish keystream from random bits with a high probability. The adversary's ability is intermediate between a regular distinguisher and prefix distinguisher.

3.2 Construction of Distinguisher

The adversary chooses a key at random. Then the adversary randomly generates IV and inputs matrix X, X' that has a difference of i-th bit. The number of inputs is m. Output sequences are $\{z_0, \cdots, z_{m-1}\}, \{z'_0, \cdots, z'_{m-1}\}$, where $z_i, z'_i \in \{0, 1\}$. After that, the adversary observes $t_i = z_i \oplus z'_i, (0 \leq i < m)$, where $\oplus$ is exclusive-or.

If $\{z_0, \cdots, z_{m-1}\}, \{z'_0, \cdots, z'_{m-1}\}$ were random bit sequences, the probabilities:

$$\begin{aligned}\Pr[t_i = 1] &= \Pr[t_i = 0]\\ &= \frac{1}{2}, (0 \leq i < m)\end{aligned}$$

are hold.

If $\{z_0, \cdots, z_{m-1}\}$ and $\{z'_0, \cdots, z'_{m-1}\}$ were keystream bits from a stream cipher, we obtain the following equations:

$$\Pr[t_i = 1] = \frac{1}{2}(1 + \varepsilon_d)$$
$$\Pr[t_i = 0] = \frac{1}{2}(1 - \varepsilon_d), (0 \le i < m)$$

In this instance, the number of keystream bits required for a distinguishing attack is $O(\varepsilon_d^{-2})$, where ε_d is the differential bias explained in Section 3.3. If ε_d is large enough, an adversary can distinguish keystream bits from random bit sequences. For example ε_d is sufficiently large for 7-round Salsa20 to distinguish keystream bits[15]. We propose a double-bit distinguisher for 9-round Salsa20 and 8-round ChaCha in the later section.

3.3 Distinguishing attack using double-bit differentials

In this section, we propose a distinguishing attack using double-bit differentials, which extends the single-bit distinguishing attack in[15]. Let x_i, x'_i be the i-th word of the initial matrix X, X', and j-th bit of x_i is denoted $[x_i]_j$. Then, let $[\varDelta_i^r]_j$ be a differential of j-th bit of i-th word after r rounds, where $[\varDelta_i^0]_j = [x_i]_j \oplus [x'_i]_j$. In[15], the differential of r rounds output under $[\varDelta_i^0]_j = 1$ is denoted $([\varDelta_p^r]_q | [\varDelta_i^0]_j)$ [1], and a single-bit differential is defined by

$$\Pr([\varDelta_p^r]_q | [\varDelta_i^0]_j) = \frac{1}{2}(1 + |\varepsilon_s|).$$

The bias ε_s represents the strength of the correlations between one bit in input and one bit in output. If a keystream bit is pseudorandom, ε_s must come close to 0. Aumasson indicated significant differentials between keystream bits and random bit sequences in 8 rounds of Salsa20 and 7 rounds of ChaCha. However, he could not find a significant differential, where there were more than rounds and 7 rounds.

In a distinguishing attack using double-bit differentials, the bias ε_d of the output differential is defined by

$$\Pr(([\varDelta_p^r]_q \oplus [\varDelta_s^r]_t = 1) | [\varDelta_i^0]_j) = \frac{1}{2}(1 + |\varepsilon_d|).$$

When ε_s is zero, pairs of $(p, q), (s, t)$ have no significant single-bit differentials. That means zero and one appear with a probability of $\frac{1}{2}$. In other words, a single-bit differential only indicates a frequency of $[\varDelta_p^r]_q = 1$. There is a possibility that a correlation exists between cases of $[\varDelta_p^r]_q = 1$ and $[\varDelta_s^r]_t = 1$. If the bias $\varepsilon_d \neq 0$, a double-bit differential indicates such correlations.

In concrete terms, an adversary chooses $[\varDelta_i^0]_j$ from a nonce v or a counter i; therefore, i and j for Salsa20 are chosen within the ranges $6 \le i < 10, 0 \le j < 32$. In

[1] This notation is different from [15] in a precise sense. We defined the reduced version as $X + X^r$ where r is the number of rounds.

Algorithm 3 Search for double-bit differentials

```
INPUT: $r, \alpha, \beta \in \mathbb{N}$
OUTPUT: Average of double-bit differential of $r$ round
 1: Initialize all $count$ by zero
 2: for $l = 0$ up to $\alpha$ do
 3:     for all $[\varDelta_i^0]_j$ such that $i, j$ in controllable value do
 4:         Choose key $K$ at random
 5:         Choose $X, X'$ at random where $X \oplus X' = [\varDelta_i^0]_j = 1$
 6:         $Z \leftarrow X + X^r$
 7:         $Z' \leftarrow X' + X'^r$
 8:         for all $[\varDelta_p^r]_q$ such that $0 \le p < 16, 0 \le q < 32$ do
 9:             for all $[\varDelta_s^r]_t$ such that $0 \le s < 16, 0 \le t < 32$ do
10:                 $count_{p,q,r,s}[i, j] \leftarrow count_{p,q,r,s}[i, j] + ([\varDelta_s^r]_t \oplus [\varDelta_p^r]_q)$
11:             end for
12:         end for
13:     end for
14: end for
15: $average_{p,q,r,s} \leftarrow$ average of $count_{p,q,r,s}[i]$ for all $i, j, (p, q, r, s)$
16: return $(average_{p,q,r,s}, (i, j, p, q, r, s))$
```

ChaCha, i and j are chosen within the ranges $12 \le i < 16, 0 \le j < 32$. The bias ε_d is dependent on keys k, and it is difficult to calculate all values of ε_d due to huge time complexity. The value ε_d can be guessed as a average value $\widetilde{\varepsilon_d}$ as follows;

$$\Pr_k(([\varDelta_p^r]_q \oplus [\varDelta_s^r]_t = 1)|[\varDelta_i^0]_j) = \frac{1}{2}(1 + |\widetilde{\varepsilon_d}|).$$

4 Experimental Results

In this section, we discuss the experimental results for distinguishing attacks using double-bit differentials. In Section 4.1, we present an algorithm searching for the maximum double-bit differential. Then, we demonstrate efficacy of our method using experimental results.

4.1 Algorithm

In a distinguishing attack using double-bit differentials, the adversary previously has obtained the positions of the maximum double-bit differential in order to distinguish keystream bits from random bits. First, the adversary chooses a key K at random and fixes it. Then, the adversary generates many input pairs which have a one-bit differential each other. After the calculation of the output pair corresponding to each input, the adversary searches all combinations of output positions for double-bit differentials. Finally, the adversary calculates the averages value with randomly changing keys.

Algorithm 3 shows details of the search algorithm. This algorithm requires $r, \alpha \in \mathbb{N}$, where r is a number of round, α is the number of trials required to calculate the average. The balance between the precision of outputs and the time complexity depends

Algorithm 4 Sieving a list of candidates

```
INPUT: Set of $\{i, j, p, q, r, s\}$
OUTPUT: Subset of $\{i, j, p, q, r, s\}$, which have high bias
 1: Initialize all count by zero
 2: $T \leftarrow$ A list of outputs of over threshold value $\gamma$ of Algorithm 3
 3: for $l = 0$ up to $\beta - 1$ do
 4:     $U \leftarrow$ A list of outputs of over threshold value $\gamma$ of Algorithm 3
 5:     $T \leftarrow T \cap U$
 6:     if $T = \phi$ then
 7:         return "Not Found"
 8:     end if
 9: end for
10: return $T$
```

on these parameters. We discuss the balance and our adoptions in section 4.2. After the choice of K at step 3, the chosen key is used for the next loop (from step 4 to step 15). In the loop, we calculate the average values of the double-bit differential for fixed key K are calculated. Values $[\Delta_i^0]_j$ for all i, j of *controllable value* have to be chosen at step 5, where *controllable values* are *nonce* or *counter* in the initial matrix (see Section 2, Section 3.2). Hence, in the case of Salsa20, we choose i and j within the ranges $7 \leq i < 11, 0 \leq j < 32$, or in ChaCha, we choose them within the ranges $12 \leq i < 16, 0 \leq j < 32$. From step 6 to step 13, we calculate the double-bit differential using XOR operation; the computational cost of these steps is dominant in the whole algorithm. The time complexity of the step is $(2^9)^2/2 = 2^{17}$. Remaining computational costs of the algorithm is calculated as follows; the number of iterations of the loop from step 5 is 2^7, and the number of iteration of the loop from step 2 is α. Thus, the total cost of the algorithm is $\alpha \cdot 2^{24}$.

However, If the outputs have large biases, these values can appeared at all times. Therefore, we constructs Algorithm to check reliability of the outputs of above algorithm. **Algorithm 4** shows details of the sieving algorithm.

4.2 Results

In the distinguishing attack using double-bit differentials, we need to find the maximum values of ε_d. Accordingly, we conducted an experiment shown in **Algorithm 3** to find the maximum values for Salsa20 and ChaCha. The total time complexity of the experiment is 2^{48}: the space of IV is 128 bits(=2^7), the combination of output is $2^{18}/2 = 2^{17}$, and the number of trials is 2^{24}. A Intel Core i7 3.3GHz PC requires 2 days computation for the experiment.

We sampled 2^{24} output pairs for each per one input pair. Let σ be the variance of samples, N be the average and N' is the population mean of $[\Delta_s^r]_t \oplus [\Delta_p^r]_q$, where $\sigma \approx \sqrt{N}$. The confidence interval is $[N' - \theta, N' + \theta]$ and $\theta \approx 2^{-12}$, where the confidence coefficient is 95%. In our experiment, $\widetilde{\varepsilon_d}$ is larger than 2^{-12}; thus, we set $\gamma = 2 \cdot 1/2^{-12}$. The results are shown in table 1 and table 2.

Table 1. Double bit differentials of Salsa20

Type	Round	Key length	$[\varDelta_i^0]_j$	$[\varDelta_p^r]_q$	$[\varDelta_s^r]_t$	$\lvert\widetilde{\varepsilon_d}\rvert$
Salsa20	4	256	$[\varDelta_8^0]_{17}$	$[\varDelta_6^4]_{23}$	$[\varDelta_7^4]_0$	0.177887
Salsa20	5	256	-	-	-	Not Found
Salsa20	6	256	-	-	-	Not Found
Salsa20	7	256	-	-	-	Not Found
Salsa20	8	256	-	-	-	Not Found
Salsa20	9	256	-	-	-	Not Found
ChaCha	4	256	-	-	-	Not Found
ChaCha	5	256	-	-	-	Not Found
ChaCha	6	256	-	-	-	Not Found
ChaCha	7	256	-	-	-	Not Found
ChaCha	8	256	-	-	-	Not Found

5 Concluding Remarks

We proposed new distinguishing attacks on Salsa20 and ChaCha, which uses double-bit differentials. In addition, we proposed shieving algorithm to find a proper double bit biese. However, there are no double bit biases of 5 or more round Salsa20 and ChaCha. Therefore, these ciphers are not presently under threat. Obviously, the distinguishing attack using double-bit differentials can be extended to distinguishing attacks using a triple-bit differential or more-bit differentials. We will improve the applicability of our method to extend the number of bits for differentials in our future research.

References

1. JP. Aumasson, S. Fischer, S. Khazaei, and W. Meier. New features of Latin dances: analysis of Salsa, ChaCha, and Rumba. In *Fast Software Encryption 2008*, pp. 470–488, 5086.
2. S. Babbage, C. D. Cannière, A. Canteaut, C. Cid, H. Gilbert, T. Johansson, M. Parker, B. Preneel, V. Rijmen, and M. Robshaw. The estream portfolio (rev. 1). eSTREAM, ECRYPT Stream Cipher project, 2008. `http://www.ecrypt.eu.org/stream/portfolio_revision1.pdf`.
3. D. J. Bernstein. ChaCha, a variant of Salsa20. *The State of the Art of Stream Ciphers SASC 2008*, 2008. `http://cr.yp.to/ChaCha.html`.
4. D. J. Bernstein. Response to "Slid pairs in Salsa20 and Trivium", 2008. `http://cr.yp.to/snuffle/reslid-20080925.pdf`.
5. D. J. Bernstein. The Salsa20 family of stream ciphers. In D. Buell, editor, *New Stream Cipher Designs*, No. 4986 in Lecture Notes in Computer Science, pp. 84–97. Springer, 2008. `http://cr.yp.to/salsa20.html`.
6. A. Biryukov. A new 128-bit key stream cipher LEX. eSTREAM, ECRYPT Stream Cipher project, 2005. `http://www.ecrypt.eu.org/stream/nls.html`.
7. J. Y. Cho and J. Pieprzyk. Crossword puzzle attack on NLS. Cryptology ePrint Archive, Report 2006/049, 2006. `http://eprint.iacr.org/`.
8. J. Y. Cho and J. Pieprzyk. Linear distinguishing attack on NLS. In *eSTREAM The ECRYPT Stream Cipher Project*, No. 2006/044, pp. 285–295, 2006.

9. P. Crowley. Truncated differemtial cryptanalysis of five round Salsa20. *The State of the Art of Stream Ciphers SASC2006*, pp. 198–202, 2006.
10. H. Englund, M. Hell, and T. Johansson. A note on distinguishing attacks. *IEEE Trans. on Info. Theory*, pp. 1–4, 2007.
11. eSTREAM. Ecrypt stream cipher project. `http://www.ecrypt.eu.org/stream`.
12. S. Fischer, W. Meier, C. Berbain, J.-F. Biasse, and M. Robshaw. Non-randomness in eSTEAM candidate Salsa20 and TSC-4. In *Indocrypt2006*, No. 4329 in Lecture Notes in Computer Science, pp. 2–16. Springer, 2006.
13. A. O. Freier, P. Kocher, and P. C. Kaltorn. The SSL protocol version 3.0 draft. `http://home.netscape.com/eng/ssl3/draft302.txt`.
14. P. Hawkes, M. Paddon, G. Rose, and M. Wiggers de Vries. Primitive specification for NLS. eSTREAM, ECRYPT Stream Cipher project, 2005. `http://www.ecrypt.eu.org/stream/nls.html`.
15. S. Khazaei. *Neutrality-Based Symmetric Cryptanalysis*. PhD thesis, Lausanne EPFL, 2010.
16. S. Kunzli and W. Meier. Distinguishing attack on MAG. eSTREMA report, Report 2005/053, 2005. `http://www.ecrypt.eu.org/stream/papersdir/053.pdf`.
17. A. N. Lebedev, A. Ivanov, S. Starodubtzev, and A. Kolchkov. Yamb LAN crypto submission to the ecrypt stream cipher project. In *eSTREAM The ECRYPT Stream Cipher Project*, No. 2005/034, 2005.
18. S. Poul, B. Preneel, and G. Sekar. Distinguishing attacks on the stream cipher Py. In *Indocrypt2008*, No. 5365 in Lecture Notes in Computer Science, pp. 1–14. Springer, 2008.
19. D. Priemuth-Schmid and A. Biryukov. Slid pairs in Salsa20 and Trivium. In *Fast Software Encryption FSE2006*, No. 4047 in Lecture Notes in Computer Science, pp. 405–421. Springer, 2006.
20. G. Rose. A stream cipher based on linear feedback over $GF(2^8)$. In *Proc. Australian Conference on Information Security and Privacy*, Vol. 1438/1998, pp. 135–146. Springer, 1998.
21. IEEE Computer Society. Wireless lan medium access control (MAC) and physical layer (PHY) speciffications. IEEE Std802.11, 1999.
22. Y. Tsunoo, T. Saito, H. Kubo, and M. Shigeri. Cryptanalysis of Mir-1, a T-function based stream cipher, 2006.
23. Y. Tsunoo, T. Saito, H. Kubo, T. Suzaki, and H. Nakashima. Differential cryptanalysis of Salsa20/8. *The State of the Art of Stream Ciphers SASC 2007*, 2007.
24. H. Wu and B. Preneel. Distinguishing attack on stream cipher Yamb. In *eSTREAM The ECRYPT Stream Cipher Project*, No. 2005/043, 2005.