Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha[⋆]

Zhenqing Shi[1], Bin Zhang[2], Dengguo Feng[1], and Wenling Wu[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China
[2] Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, 100195, China
zhenqingshi@gmail.com, {zhangbin,feng,wwl}@is.iscas.ac.cn

Abstract. Salsa20 is a stream cipher designed by Bernstein in 2005 and Salsa20/12 has been selected into the final portfolio of the eSTREAM Project. ChaCha is a variant of Salsa20 with faster diffusion for similar performance. The previous best results on Salsa20 and ChaCha proposed by Aumasson et al. exploits the differential properties combined with the probabilistic neutral bits (PNB). In this paper, we extend their approach by considering a new type of distinguishers, named (column and row) chaining distinguishers. Besides, we exhibit new high probability second-order differential trails not covered by the previous methods, generalize the notion of PNB to probabilistic neutral vectors (PNV) and show that the set of PNV is no smaller than that of PNB. Based on these findings, we present improved key recovery attacks on reduced-round Salsa20 and ChaCha. Both time and data complexities of our attacks are smaller than those of the best former results.

Keywords: Stream ciphers, Salsa20, ChaCha, Neutral bits, Distinguisher.

1 Introduction

Salsa20 [1] is a stream cipher designed by Bernstein in 2005 for the eSTREAM project [2]. The 8- and 12-round variants, Salsa20/8 and Salsa20/12 [3], were also published. ChaCha [4] is a variant of Salsa20 with faster diffusion in the core function for similar performance. After three evaluation phases, Salsa20/12 was selected into the final portfolio of the eSTREAM Project in 2008.

Since their publication, Salsa20 and ChaCha have undergone significant cryptographic analysis. In 2006, Crowley presented a truncated differential cryptanalysis on Salsa20/5 [5]. The result revealed that the diffusion of Salsa20 was not ideal, though it did not threaten the full round security of Salsa20. For the same variant, V. Velichkov et al. mounted a key recovery attack using UNAF [6]

[⋆] This work was supported by the programs of the National Natural Science Foundation of China (Grant No. 60833008, 60603018, 61173134, 91118006, 61272476), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701) and the National Grand Fundamental Research 973 Program of China(Grant No. 2013CB338002).

Table 1. Comparisons of our attacks with other well known attacks on Salsa20 and ChaCha

Cipher	Round/Key length	Time	Data	Reference
Salsa20	5/256	2^{165}	2^{6}	[5]
		2^{167}	2^{7}	[6]
		2^{55}	2^{10}	This work
	6/256	2^{177}	2^{16}	[7]
		2^{73}	2^{16}	This work
	7/256	2^{151}	2^{26}	[9]
		2^{148}	2^{24}	This work
	8/256	2^{251}	2^{31}	[9]
		2^{250}	2^{27}	This work
	7/128	2^{111}	2^{21}	[9]
		2^{109}	2^{19}	This work
ChaCha	6/256	2^{139}	2^{30}	[9]
		2^{136}	2^{28}	This work
	7/256	2^{248}	2^{27}	[9]
		$2^{246.5}$	2^{27}	This work
	6/128	2^{107}	2^{30}	[9]
		2^{105}	2^{28}	This work

at FSE 2012. At Indocrypt 2006, Fischer et al. described some non-randomness properties of Salsa20/5 and used this observation to construct a key-recovery attack on Salsa20/6 [7]. At SASC 2007, Tsunoo et al. exploited a bias of Salsa20/4 to construct an attack on Salsa20/7 [8]. So far, the best key recovery attacks on variants of Salsa20 and ChaCha were proposed by Aumasson et al [9] at FSE 2008, exploiting first-order differential properties combined with the probabilistic neutral bits (PNB) technique.

In this paper, we extend the approach of Aumasson et al. by considering a new type of distinguishers, named (column and row) chaining distinguishers, which can efficiently make use of the biases of multiple differential trails and the matrix structure of the cipher. Besides, we find new high probability second-order differential trails that are not covered by the previous results, some of which are employed in our attack. The notion of PNB is generalized to that of probabilistic neutral vectors (PNV), which investigate the properties of the underlying function when more than one input bit are flipped simultaneously and include the PNB as a special case. It is shown that the set of PNV is no smaller than that of PNB. Based on these findings, we construct improved key recovery attacks on reduced-round Salsa20 and ChaCha, repectively. Both time and data complexities of our new attacks are smaller than those of the best former results. Table 1 presents our results together with comparisons with other well known attacks.

The rest of the paper is organized as follows. A brief description of Salsa20 and ChaCha is presented in Section 2. The attacks of Aumasson et al. are recalled in Section 3. In Section 4, our new attacks are described in details with the introduction of the (column and row) chaining distinguishers, high probability differential trails and the notion of PNV. Finally, we conclude the paper in Section 5.

2 Description of Salsa20 and ChaCha

2.1 Salsa20

Salsa20 operates on 32-bit words and supports keys of 128 bits and 256 bits. During its operation, the key, a 64-bit nonce (unique message number), a 64-bit counter and four 32-bit constants are used to construct the 512-bit initial state.

Denote the 256-bit key by $k = (k_0, k_1, ..., k_7)$, the 64-bit nonce by $v = (v_0, v_1)$ and the 64-bit counter corresponding to the integer i by $t = (t_0, t_1)$. The Salsa20 function acts on the following 4×4 matrix of 32-bit words.

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

The c_i's are predefined constants. For the 128-bit key k', just fill the 256 key bits in the matrix as $k = k'||k'$. If not mentioned otherwise, we focus on the 256-bit version. A keystream block Z is then defined as $Z = X + X^{20}$, where $+$ is the wordwise integer addition and $X^r = \mathsf{Round}^r(X)$ with the round function Round of Salsa20. This round function is based on the following nonlinear operation (also called the quarterround function), which transforms a vector (x_0, x_1, x_2, x_3) to (z_0, z_1, z_2, z_3) by sequentially computing

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

In odd numbers of rounds (which are called columnrounds in the original specification of Salsa20), the nonlinear operation is applied to the columns (x_0, x_4, x_8, x_{12}), (x_5, x_9, x_{13}, x_1), $(x_{10}, x_{14}, x_2, x_6)$, $(x_{15}, x_3, x_7, x_{11})$. In even numbers of rounds (which are also called the rowrounds), the nonlinear operation is applied to the rows (x_0, x_1, x_2, x_3), (x_5, x_6, x_7, x_4), $(x_{10}, x_{11}, x_8, x_9)$, $(x_{15}, x_{12}, x_{13}, x_{14})$. We write Salsa20/$R$ for R-round variants, i.e., with $Z = X + X^R$. After R iterations of the Salsa20/R round function, the updated state is used as a 512-bit keystream output. Each such output block is an independent combination of the key, nonce, and counter.

2.2 ChaCha

ChaCha is similar to Salsa20 except the following three differences: the composition of the initial matrix, the composition of the quarterround function and the round function. See [4] for details.

3 Aumasson et al's Attacks on Salsa20 and ChaCha

In this section, we briefly recall the first-order differential attacks of Aumasson et al. These attacks are based on the concept of probabilistic neutral bits (PNB). Let us first list some notations used hereafter.

- Let x_i be the i-th word of the initial matrix X and the j-th least significant bit of x_i is denoted by $[x_i]_j$.
- Let x'_i be an associated word with the difference $\Delta_i^0 = x_i \oplus x'_i$.
- The first-order differential for the input X, with a single-bit input-difference $[\Delta_i^0]_j = 1$ and a single-bit output-difference $[\Delta_p^r]_q$ after r rounds, is denoted by $([\Delta_p^r]_q|[\Delta_i^0]_j)$.

3.1 First-Order Differential Analysis of Salsa20 and ChaCha

First note that the round function of Salsa20 and ChaCha is invertible. Define the r-round inverse $X^{-r} = \mathsf{Round}^{-r}(X)$, which is different depending on whether it inverts after an odd or an even number of rounds.

For a fixed differential $([\Delta_p^r]_q|[\Delta_i^0]_j)$, the corresponding output Z and Z' can be observed for nonce v, counter t and key k. Based on $Z = X + X^R$, if we have v, t and k, we can get $X^r = (Z - X)^{r-R}$ and $(X')^r = (Z' - X')^{r-R} (r < R)$. Then $[\Delta_p^r]_q = X^r \oplus (X')^r = (Z - X)^{r-R} \oplus (Z' - X')^{r-R}$ can be observed. Define the Boolean function $f(k, v, t, Z, Z') = [\Delta_p^r]_q$. Then, for a fixed key, the bias ε_d of f is defined by $Pr\{f(k, v, t, Z, Z') = 1\} = Pr\{[\Delta_p^r]_q = 1|[\Delta_i^0]_j\} = \frac{1}{2}(1 + \varepsilon_d)$, where the probability holds over all nonces and counters.

Given enough output block pairs with the presumed differential, we have $Pr\{f(\bar{k}, v, t, Z, Z') = 1\} = \frac{1}{2}(1 + \varepsilon_d)$ conditioned on $\bar{k} = k$, whereas for (almost all) $\bar{k} \neq k$ we expect f be unbiased, i.e., $Pr\{f(\bar{k}, v, t, Z, Z') = 1\} = \frac{1}{2}$. Note that the complexity of the above way is 2^{256}. Instead, Aumasson et al. proposed to find some approximation g of f which effectively depends on s key bits ($s < 256$). Let f be correlated to g with the bias ε_a, i.e., $Pr\{g(k', v, t, Z, Z') = f(k, v, t, Z, Z')\} = \frac{1}{2}(1+\varepsilon_a)$, where k' is the s bits subkey of k and the probability holds over all nonces and counters. Assume $Pr\{g(k', v, t, Z, Z') = 1\} = \frac{1}{2}(1+\varepsilon)$, then under some reasonable independency assumptions, we have $\varepsilon = \varepsilon_d \cdot \varepsilon_a$. Hence, given enough output block pairs with the presumed differential, we can verify the correctness of a guessed candidate subkey $\bar{k}'$ for the subkey k' by evaluating the bias of the function g. More precisely, we have $Pr\{g(\bar{k}', v, t, Z, Z') = 1\} = \frac{1}{2}(1+\varepsilon)$ if $\bar{k}' = k'$, whereas for (almost all) $\bar{k}' \neq k'$ we expect g be unbiased, i.e. $Pr\{g(\bar{k}', v, t, Z, Z') = 1\} = \frac{1}{2}$. In this way, the complexity is reduced from 2^{256} to 2^s. Next, we will give a concise description of how the approximation g of f is found.

3.2 Key Recovery Attacks on Salsa20 and ChaCha

In [9], Aumasson et al. gave an efficient way of finding suitable approximations $g(k', v, t, Z, Z')$ of the function $f(k, v, t, Z, Z')$ using the concept of PNB.

Definition 1. *The neutrality measure of the key bit k_i with respect to the function $f(k, v, t, Z, Z')$ is defined as γ_i, where $Pr = \frac{1}{2}(1+\gamma_i)$ is the probability (over all k, v and t) that complementing the key bit k_i does not change the output of $f(k, v, t, Z, Z')$.*

When the threshold γ is set, all the key bits can be divided into two sets: significant key bits with neutrality measure $\gamma_i < \gamma$ (size of s) and non-significant key bits with neutrality measure $\gamma_i \geq \gamma$ (size of ns). Then we can define $g(k', v, t, Z, Z')$ as an approximation of $f(k, v, t, Z, Z')$ just simply making k' be the significant key bits and the non-significant key bits be set to fixed values (e.g. all zero). More details of the construction of the distinguisher g can be found in [9]. Then, the whole attack can be carried out as follows:

Attack Online

```
Parameters: $N$, $s$, $\varepsilon$
1: Collect $N$ pairs of keystream blocks where each pair is produced by states
   with a random nonce and counter (satisfying the relevant input-difference).
2: For each choice of the subkey (i.e. the $s$ significant key bits) do:
   (2.1) Compute the bias of $g$ using the $N$ keystream block pairs.
   (2.2) If the optimal distinguisher legitimates the subkeys candidate as a
         (possibly) correct one, perform an additional exhaustive search over
         the $ns$ non-significant key bits in order to check the correctness of this
         filtered subkey and to find the non-significant key bits.
   (2.3) Stop if the right key is found and output the recovered key.
```

In this attack, there is a set of 2^s sequences of random variables by guessing s significant key bits. Actually, $2^s - 1$ of them should verify the null hypothesis H_0, and a single one verify the alternative hypothesis H_1. For a realization a of the corresponding random variable A, the decision rule $D(a) = i$ to accept H_i can lead to two types of errors:

1. Non-detection: $D(a) = 0$ and $A \in H_1$. The probability of this event is p_{nd}.
2. False alarm: $D(a) = 1$ and $A \in H_0$. The probability of this event is p_{fa}.

The Neyman-Pearson decision theory gives results to estimate the number of samples N required to get some bounds on the probabilities. It can be shown that

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

samples suffices to achieve $p_{nd} = 1.3 \times 10^{-3}$ and $p_{fa} = 2^{-\alpha}$. Calculus details and the construction of the optimal distinguisher can be found in [10].

The time complexity of this attack can be estimated as follows. Step 2 is repeated 2^s times. For each subkey, step (2.1) is always executed which has complexity of N. The search part of step (2.2) is performed only with probability $p_{fa} = 2^{-\alpha}$ which brings an additional cost of $2^{ns}(= 2^{256-s})$ in case a subkey passes the optimal distinguisher's filter. Therefore, the complexity of step (2.2) is $2^{256-s} \cdot p_{fa}$, showing a total complexity of $2^s(N + 2^{256-s} \cdot p_{fa}) = 2^s \cdot N + 2^{256-\alpha}$.

4 Our Attacks

In the above attack, α (and hence N) is chosen such that it minimizes $2^s \cdot N + 2^{256-\alpha}$ based on the single distinguisher g. In this section, our new attacks are described in details with the introduction of the (column and row) chaining distinguishers, high probability differential trails and the notion of PNV.

4.1 Chaining Distinguishers

For a subkey K', define the set $S(K') = \{k_i | k_i \text{ is involved in the subkey } K'\}$. Note that, all the single distinguishers we mentioned below are constructed with the method in Section 3.

Definition 2. *Column Chaining Distinguishers(CCD for short): For a collection of subkey $\{K'_i\}_{i\in A}$ with $S(K'_i) \subset S(K'_j)$ ($\forall i, j \in A$ and $i < j$), if there exists a collection of distinguishers $\{D_i\}_{i\in A}$, and $\forall i \in A$ the distinguisher D_i effectively depends on the subkey K'_i, we call $\{D_i\}_{i\in A}$ the Column Chaining Distinguishers of $\{K'_i\}_{i\in A}$.*

What's the advantage of CCD? Here is an example: Suppose $A = \{1, 2, 3\}$, and $\{D_i\}_{i\in A}$ are CCD of $\{K'_i\}_{i\in A}$ with $S(K'_1) \subset S(K'_2) \subset S(K'_3)$. For each $i \in A$, there is a relation between the data N_i and $(p_{fa})_i = 2^{-\alpha_i}$ with the $(p_{nd})_i$ fixed (see Eq. 1). Let $s_i = |S(K'_i)|$, so we have $s_1 < s_2 < s_3$. Then the execution of CCD is described as follows:

```
Execution of CCD
Parameters: $\{N_i\}$, $\{s_i\}$
1: For each subkey candidate $\bar{K'_1}$ by guessing the $s_1$ key bits of set $S(K'_1)$,
   verify $\bar{K'_1}$ with the distinguisher $D_1$ of $N_1$ pairs of keystream blocks, if this
   step succeeds, do Step 2;
2: Guess the $s_2 - s_1$ key bits of set $S(K'_2) - S(K'_1)$, then for the subkey candidate
   $\bar{K'_2}$ ($\bar{K'_1}$ plus $s_2 - s_1$ guessed key bits), verify $\bar{K'_2}$ with the distinguisher $D_2$
   of $N_2$ pairs of keystream blocks, if this step succeeds, do Step 3;
3: Guess the $s_3 - s_2$ key bits of set $S(K'_3) - S(K'_2)$, then for the subkey candidate
   $\bar{K'_3}$ ($\bar{K'_2}$ plus $s_3 - s_2$ guessed key bits), verify $\bar{K'_3}$ with the distinguisher $D_3$
   of $N_3$ pairs of keystream blocks, if this step succeeds, do Step 4;
4: Perform an additional exhaustive search over the $256 - s_3$ key bits (i.e. not
   in the set $S(K'_3)$) in order to check the correctness of this filtered subkey
   and to find the remaining key bits.
```

Let us discuss the time complexity of such an attack. Step 1 is repeated for all 2^{s_1} subkey candidates, and each incorrect subkey candidate passes the test of distinguisher D_1 with probability $(p_{fa})_1 = 2^{-\alpha_1}$ (according to N_1 pairs of keystream blocks). Step 2 needs to search over $s_2 - s_1$ key bits of set $S(K'_2) - S(K'_1)$, and each incorrect subkey candidate passes the test of distinguisher D_2 with probability $(p_{fa})_2 = 2^{-\alpha_2}$ (according to N_2 pairs of keystream blocks). Step 3 needs to search over $s_3 - s_2$ key bits of set $S(K'_3) - S(K'_2)$, and each incorrect subkey candidate passes the test of distinguisher D_3 with probability $(p_{fa})_3 = 2^{-\alpha_3}$ (according to N_3 pairs of keystream blocks). And step 4 needs to search over the remaining $256 - s_3$ key bits. So the total complexity is

$$\begin{aligned} & 2^{s_1} \cdot N_1 + 2^{s_1} \cdot (p_{fa})_1 \cdot 2^{s_2-s_1} \cdot N_2 + \\ & 2^{s_1} \cdot (p_{fa})_1 \cdot 2^{s_2-s_1} \cdot (p_{fa})_2 \cdot 2^{s_3-s_2} \cdot N_3 + \\ & 2^{s_1} \cdot (p_{fa})_1 \cdot 2^{s_2-s_1} \cdot (p_{fa})_2 \cdot 2^{s_3-s_2} \cdot (p_{fa})_3 \cdot 2^{256-s_3} \\ = \ & 2^{s_1} \cdot N_1 + 2^{s_2-\alpha_1} \cdot N_2 + 2^{s_3-\alpha_1-\alpha_2} \cdot N_3 + 2^{256-\alpha_1-\alpha_2-\alpha_3} \end{aligned}$$

If we use single distinguisher D_1 to recover the key, the time complexity is $2^{s_1} \cdot N_1 + 2^{256-\alpha_1}$. Furthermore, we can easily get:

$$\min_{N_1}\{2^{l_1}\cdot N_1+2^{256-\alpha_1}\} \geq \min_{N_1,N_2,N_3}\{2^{l_1}\cdot N_1+2^{l_2-\alpha_1}\cdot N_2+2^{l_3-\alpha_1-\alpha_2}\cdot N_3+2^{256-\alpha_1-\alpha_2-\alpha_3}\}.$$

Two ordinary methods of constructing CCD are as follows:

First method of constructing CCD

```
1: (a) Find a highly biased differential and set a threshold $\gamma^{(1)}$;
   (b) Estimate the measure $\gamma_i$ of all the key bits and put all those key bits
       with $\gamma_i \geq \gamma^{(1)}$ into the set $S_1$;
   (c) Construct a single distinguisher $D_1$ with the key bits in $S_1$ being set to
       a fixed value, if the bias of $D_1$ is non-negligible, do Step 2;
2: (a) Find another highly biased differential and set a threshold $\gamma^{(2)}$;
   (b) Estimate the measure $\gamma_i$ of all the key bits in set $S_1$ and put all those
       key bits in set $S_1$ with $\gamma_i \geq \gamma^{(2)}$ into the set $S_2$;
   (c) Construct a single distinguisher $D_2$ with the key bits in $S_2$ being set to
       a fixed value(the same as in Step1), if the bias of $D_2$ is non-negligible,
       do Step 3;
3: (a) Find another highly biased differential and set a threshold $\gamma^{(3)}$;
   (b) Estimate the measure $\gamma_i$ of all the key bits in set $S_2$ and put all those
       key bits in set $S_2$ with $\gamma_i \geq \gamma^{(3)}$ into the set $S_3$;
   (c) Construct a single distinguisher $D_3$ with the key bits in $S_3$ being set to
       a fixed value(the same as in Step1), if the bias of $D_3$ is non-negligible,
       do Step 4;
4: Continue the work until only a few key bits are left to be guessed.
```

Second method of constructing CCD

```
1: Find a highly biased differential and set a threshold $\gamma$;
2: Construct a distinguisher $g_\gamma(K'_\gamma, v, t, Z, Z')$ based on the subkey $K'_\gamma =$
   $\{k_i | \gamma_i < \gamma\}$;
3: If the bias of $g_\gamma(K'_\gamma, v, t, Z, Z')$ is non-negligible, set a series of thresholds
   $\gamma^{(1)} < \gamma^{(2)} < ... < \gamma^{(e)}$ with $\gamma^{(i)} \geq \gamma$, and for each $\gamma^{(i)}$, construct the dis-
   tinguisher $g_{\gamma^{(i)}}(K'_{\gamma^{(i)}}, v, t, Z, Z')$ effectively depending on the subkey $K'_{\gamma^{(i)}}$.
```

The first method is feasible for lower rounds of Salsa20 and ChaCha because of sufficient numbers of PNB's (usually more than half of the key bits with high γ_i). And the second method of constructing CCD is based on a single distinguisher, which is more feasible when the numbers of PNB's are insufficient.

Definition 3. *Row Chaining Distinguishers(RCD for short): For a fixed subkey K', if there exists a collection of distinguishers $\{D_i\}_{i\in A}$ which effectively depend on the subkey K', we call the $\{D_i\}_{i\in A}$ the Row Chaining Distinguishers for K'.*

The advantage of RCD is obvious: firstly, some incorrect subkey candidate $\bar{K}'$ may verify the alternative hypothesis of distinguisher D_i, while the probability that it verifies all the alternative hypothesis of distinguishers $\{D_i\}_{i\in A}$ is much lower; secondly, RCD can be used as a CCD[1] . We will show how to construct RCD based on a second-order differential in the next subsection.

4.2 Second-Order Differential Analysis on Salsa20 and ChaCha

First, we recall the second-order differential: let X be the initial matrix, X_1, X_2 and X_3 be associated initial matrices with a single-bit input-difference $[\Delta_i^0]_j = 1$, a single-bit input-difference $[\Delta_m^0]_n = 1$ and the double-bit input-differences $[\Delta_i^0]_j = 1$ and $[\Delta_m^0]_n = 1$ respectively. Note that $(i-m)^2 + (j-n)^2 = 0$ should not hold. We consider a single-bit output-difference $[\Delta_p^r]_q = [X_p^r]_q \oplus [(X_1)_p]_q^r \oplus [(X_2)_p^r]_q \oplus [(X_3)_p^r]_q$ after r rounds. Then the second-order differential for the input X is denoted by $([\Delta_p^r]_q|[\Delta_i^0]_j, [\Delta_m^0]_n)$. The bias ε_d of the output-difference is defined by $Pr\{([\Delta_p^r]_q = 1|[\Delta_i^0]_j, [\Delta_m^0]_n)\} = \frac{1}{2}(1+\varepsilon_d)$, where the probability holds over all keys, nonces and counters. We found many highly biased differentials for Salsa20 and ChaCha (see Table 2).

For a fixed differential $([\Delta_p^r]_q|[\Delta_i^0]_j, [\Delta_m^0]_n)$ with bias ε_d, let $Z = X+X^R$, $Z_1 = X_1+(X_1)^R$, $Z_2 = X_2+(X_2)^R$, and $Z_3 = X_3+(X_3)^R$, so Z, Z_1, Z_2, and Z_3 can be observed for nonce v, counter t and key k. As mentioned in section 3, the round functions of Salsa20 and ChaCha is invertible, i.e. $X^r = (Z-X)^{r-R} (r < R)$, so $[\Delta_p^r]_q = [((Z-X)^{r-R} \oplus (Z_1-X_1)^{r-R} \oplus (Z_2-X_2)^{r-R} \oplus (Z_3-X_3)^{r-R})_p]_q$. Define $F_{p,q,i,j,m,n}(k, v, t, Z, Z_1, Z_2, Z_3) = [\Delta_p^r]_q$. For short, we define $F_{p,q,i,j,m,n}(k, W) = [\Delta_p^r]_q$ where $W = (v, t, Z, Z_1, Z_2, Z_3)$. The next work is finding suitable approximations $G_{p,q,i,j,m,n}(k', W)$ of the function $F_{p,q,i,j,m,n}(k, W)$. Here, we also use the PNB's mentioned in Section 3.

After all the neutrality measure γ_l's of each key bit k_l be estimated, we set a threshold γ and put all the key bits with $\gamma_l < \gamma$ into a set denoted by $K_{p,q,i,j,m,n}(\gamma) = \{k_l|\gamma_l < \gamma\}$. Having found the set $K_{p,q,i,j,m,n}(\gamma)$, we simply let k' be subkey with the key bits in the set $K_{p,q,i,j,m,n}(\gamma)$ and define $G_{\gamma,p,q,i,j,m,n}(k', W)$ as $F_{p,q,i,j,m,n}(k, W)$ with the remaining key bits (i.e. not in the set $K_{p,q,i,j,m,n}(\gamma)$) with a fixed value. The bias ε_a of the correlation between F and G is defined by $Pr\{G_{\gamma,p,q,i,j,m,n}(k', W) = F_{p,q,i,j,m,n}(k, W)\} = \frac{1}{2}(1+\varepsilon_a)$, where the probability holds over all keys, nonces and counters. Denote the bias

[1] Actually, RCD are special CCD with the subkey unchanged.

Table 2. Some highly biased differentials for Salsa20/4 and ChaCha3

Type	$[\Delta_i^0]_j, [\Delta_m^0]_n$	$[\Delta_p^r]_q$	ε_d
Salsa20/4	$[\Delta_7^0]_{24}, [\Delta_8^0]_{17}$	$[\Delta_1^4]_7$	0.67
	$[\Delta_7^0]_{24}, [\Delta_8^0]_{18}$	$[\Delta_1^4]_7$	0.64
	$[\Delta_7^0]_{24}, [\Delta_8^0]_{19}$	$[\Delta_1^4]_7$	0.62
	$[\Delta_7^0]_{24}, [\Delta_8^0]_{20}$	$[\Delta_1^4]_7$	0.58
	$[\Delta_7^0]_{25}, [\Delta_8^0]_{17}$	$[\Delta_1^4]_7$	0.59
ChaCha3	$[\Delta_{12}^0]_{15}, [\Delta_{13}^0]_{20}$	$[\Delta_3^3]_0$	0.43
	$[\Delta_{12}^0]_{20}, [\Delta_{15}^0]_{15}$	$[\Delta_2^3]_0$	0.43
	$[\Delta_{13}^0]_{15}, [\Delta_{14}^0]_{20}$	$[\Delta_0^3]_0$	0.43
	$[\Delta_{14}^0]_{15}, [\Delta_{15}^0]_{20}$	$[\Delta_1^3]_0$	0.43
	$[\Delta_{13}^0]_{16}, [\Delta_{14}^0]_{20}$	$[\Delta_0^3]_0$	0.41

of G by ε, i.e. $Pr\{G_{\gamma,p,q,i,j,m,n}(k', W) = 1\} = \frac{1}{2}(1+\varepsilon)$. Under some reasonable independency assumptions, the equality $\varepsilon = \varepsilon_d \cdot \varepsilon_a$ holds. Hence, given enough output block pairs with the presumed differential, we can verify the correctness of a guessed candidate subkey $\bar{k}'$ for the subkey k' by evaluating the bias of the function G. More precisely, we have $Pr\{G_{\gamma,p,q,i,j,m,n}(k', W) = 1\} = \frac{1}{2}(1+\varepsilon)$ conditioned on $\bar{k}' = k'$, whereas for (almost all) $\bar{k}' \neq k'$ we expect G be unbiased, i.e. $Pr\{G_{\gamma,p,q,i,j,m,n}(k', W) = 1\} = \frac{1}{2}$. The way for searching such a distinguisher is similar to that of the first-order differentials.

Now, we show how to use the second-order differentials to construct RCD. For a second-order differential $([\Delta_p^r]_q | [\Delta_i^0]_j, [\Delta_m^0]_n)$, we choose a threshold γ empirically and construct a single distinguisher $G_{\gamma,p,q,i,j,m,n}(k', W)$ using the method above, where k' is the subkey of all key bits in the set $K_{p,q,i,j,m,n}(\gamma)$. In order to construct RCD, the subkey should stay the same, i.e. the set $K_{p,q,i,j,m,n}(\gamma)$ should stay the same. Now, we consider the factors of the set $K_{p,q,i,j,m,n}(\gamma)$: p, q, i, j, m, n, γ. By tests, we find: if the value of p changes, the set $K_{p,q,i,j,m,n}(\gamma)$ will change with a high probability, so do the factors q, i, m, γ; while, if only the factor j changes, $K_{p,q,i,j,m,n}(\gamma)$ will stay the same with a high probability, so does the factor n. Hence, for the distinguisher $G_{\gamma,p,q,i,j,m,n}(k', W)$, we search over all j's only or all m's only to construct a new distinguisher with subkey unchanged. Here we give an example of the RCD on 256-bit ChaCha7. We construct 4-Step RCD $\{G_{\gamma,9,0,14,j,15,12}(k, W)\}_{j\in\{0,1,2,28\}}$. Let $\gamma = 0.3$, and we get $K_{9,0,14,j,15,12}(\gamma)$ ={ 3, 7, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 39, 40, 63, 67, 79, 80, 95, 99, 159, 160, 184, 185, 186, 187, 188, 189, 190, 191, 200, 255} for any $j \in \{0, 1, 2, 28\}$.

Actually, we can easily find RCD for Salsa20 and ChaCha using the method above. However, we did not find enough PNB's to improve our attacks[2] on Salsa20 and ChaCha. And such a reality limits us to display the use of RCD. We believe the concept of RCD can be used in other ciphers.

4.3 Probabilistic Neutral Vectors

Note that, contrary to the mutual interaction between neutral bits, we have directly combined several PNB's without altering their probabilistic quality, so do J. Aumasson et al. In order to justify the reasonableness, we introduce a generalized concept of PNB's called *probabilistic neutral vectors*(PNV's).

Definition 4. *The neutrality measure of the two-dimension key bit vector (k_i, k_j) with respect to the function $f(k, v, t, Z, Z')$ is defined as $\gamma_{(i,j)}$, where $Pr = \frac{1}{2}(1 + \gamma_{(i,j)})$ is the probability (over all k, v and t) that complementing the key bit k_i and k_j does not change the output of $f(k, v, t, Z, Z')$.*

Simulations shows that: for key bit vector (k_i, k_j), we have $\gamma_i \cdot \gamma_j \leq \gamma_{(i,j)} \leq max\{\gamma_i, \gamma_j\}$. Furthermore, for a fixed differential, denote the set $H_1(\gamma) = \{k_i | \gamma_i \geq \gamma\}$ and $H_2(\gamma) = \{k_i | \gamma_{(i,j)} \geq \gamma$ for at least one $k_j\}$. Then we have the following lemma:

Lemma 1. *For a fixed differential of Salsa20 or ChaCha, $H_1(\gamma) \subseteq H_2(\gamma)$, and hence $| H_1(\gamma) | \leq | H_2(\gamma) |$.*

For Salsa 20/7 with the differential $([\Delta_1^4]_{14} | [\Delta_7^0]_{31})$, we have $H_2(0.4) - H_1(0.4) = \{k_1, k_{78}\}$. So it's reasonable to combine several PNB's directly in our attacks. Actually, if we want to construct a distinguisher with s key bits fixed, we should use the concept of s-dimension PNV's. However, when s is too big, the cost of finding the most significant PNV's is too high to search over all C_{256}^{s} cases.

4.4 Experimental Results with CCD

We present attacks on 256-bit Salsa20/5 and Salsa20/6 with the CCD constructed by the first method. And the rest improved attacks are based on the CCD constructed by the second method. In order to compare our method with that in [9], we use the same differentials and the same threshold γ as used in [9]. And we believe there exists other choices that lead better results.

Attack on 256-bit Salsa20/5. The output differential is observed after working two rounds backward from a 5-rounds keystream block. We use five differentials: $([\Delta_3^3]_{15} | [\Delta_6^0]_0)$, $([\Delta_3^3]_9 | [\Delta_6^0]_0)$, $([\Delta_8^3]_{11} | [\Delta_7^0]_2)$, $([\Delta_8^3]_{20} | [\Delta_7^0]_0)$ and $([\Delta_{12}^3]_{23} | [\Delta_7^0]_1)$. We set the threshold $\gamma = 0.9$ and the subkeys for each distinguisher are listed in the Appendix A. The parameters of our attacks are listed in Table 3 (see

[2] We only test the second-order differential with single bit input and single bit(and double bits) output, and for other second-order differential, there maybe exist enough PNB's to improve the attacks.

Appendix B). And the total attack runs in time 2^{55} and data 2^{10}.

Attack on 256-bit Salsa20/6. The output differential is observed after working two rounds backward from a 6-rounds keystream block. We construct a CCD using four differentials: $([\Delta_6^4]_{26}|[\Delta_7^0]_{31})$, $([\Delta_1^4]_3|[\Delta_7^0]_{29})$, $([\Delta_1^4]_{26}|[\Delta_7^0]_{13})$ and $([\Delta_1^4]_{12}|[\Delta_7^0]_{13})$. For each difference, we use the same threshold $\gamma = 0.9$ and the subkeys for each distinguisher are listed in the Appendix A. The parameters of our attacks are listed in Table 4 (see Appendix B). And the total attack runs in time 2^{73} and data 2^{16}.

Attack on 256-bit Salsa20/7. We use the differential $([\Delta_1^4]_{14}|[\Delta_7^0]_{31})$ with $|\varepsilon_d| = 0.131$. The parameters and results of Aumasson's attacks are listed in Table 5 (see Appendix B). We construct 2-step CCD using $\gamma^{(1)} = 0.5$ and $\gamma^{(2)} = 0.6$ with $\varepsilon^{(1)} = 0.0022$ and $\varepsilon^{(2)} = 0.0050$ respectively. Note that, $\varepsilon^{(2)} = 0.0050 < 0.0064$. That because we test and find that such a value leads a result: if the correct key passes the distinguisher of $\gamma^{(1)} = 0.5$ (with success probability 50%[3]), then it can pass the distinguisher of $\gamma^{(2)} = 0.6$ with success probability more than 90% (we define this probability by step success probability). The time complexity is $2^{125} \cdot N_1 + 2^{132-\alpha_1} \cdot N_2 + 2^{256-\alpha_1-\alpha_2}$. We choose $\alpha_1 = 10$ and $\alpha_2 = 104$, then get $N_1 = 2^{23}$ and $N_2 = 2^{23}$ respectively by Eq.1. So the time complexity is $2^{125} \cdot N_1 + 2^{132-\alpha_1} \cdot N_2 + 2^{256-\alpha_1-\alpha_2} \approx 2^{148}$, the data complexity is $2^{23} + 2^{23} = 2^{24}$, and the success probability is $50\% \times 90\% = 45\%$.

Attack on 256-bit Salsa20/8. For the differential $([\Delta_1^4]_{14}|[\Delta_7^0]_{31})$ with $|\varepsilon_d| = 0.131$, we construct 2-step CCD. using $\gamma^{(1)} = 0.15$ [4] and $\gamma^{(2)} = 0.20$ with $\varepsilon^{(1)} = 0.00047$ and $\varepsilon^{(2)} = 0.00102$ respectively. For the threshold $\gamma^{(1)} = 0.15$, we find $ns_1 = 33$ non-significant key bits, and for the threshold $\gamma^{(2)} = 0.20$, we find $ns_2 = 30$ non-significant key bits. Note that, the value $\varepsilon^{(2)} = 0.00102$ is chosen with the step success probability 90%. The time complexity is $2^{223} \cdot N_1 + 2^{226-\alpha_1} \cdot N_2 + 2^{256-\alpha_1-\alpha_2}$. We choose $\alpha_1 = 2$ and $\alpha_2 = 7$, then get $N_1 = 2^{26.5}$ and $N_2 = 2^{25}$ respectively by Eq.1. So the time complexity is 2^{250}, the data complexity is $2^{26.5} + 2^{25} = 2^{27}$, and the success probability is $50\% \times 90\% = 45\%$.

Attack on 128-bit Salsa20/7. For the differential $([\Delta_1^4]_{14}|[\Delta_7^0]_{31})$, we construct 4-step CCD. The parameters of our attacks are listed in Table 6 (see Appendix B). Note that, the value $\varepsilon^{(i)}(i = 2, 3, 4)$ is chosen with the step success probability 95%. The time complexity is $2^{90} \cdot N_1 + 2^{92-\alpha_1} \cdot N_2 + 2^{94-\alpha_1-\alpha_2} \cdot N_3 + 2^{98-\alpha_1-\alpha_2-\alpha_3} \cdot N_4 + 2^{128-\alpha_1-\alpha_2-\alpha_3-\alpha_4} \approx 2^{109}$, the data complexity is $2^{19} + 2^{17.5} + 2^{16.5} + 2^{15.5} \approx 2^{19}$, and the success probability is $50\% \times (95\%)^3 \approx 43\%$.

[3] In [9], they use the median bias in their attack, which leads in a success probability of at least $\frac{1}{2}(1 - p_{nd}) \approx 50\%$.

[4] In [9], the threshold γ is set to 0.12, and they get $\varepsilon_a = 0.0011$. However, $\varepsilon_a = 0.0011$ is not reasonable as they say: we can only measure a bias of about $|\varepsilon_a| > c \cdot 2^{-12}$(where $c \approx 10$ for a reasonable estimation error).

Attack on 256-bit Chacha6. For the differential $([\Delta^3_{11}]_0|[\Delta^0_{13}]_{13})$ with $|\varepsilon_d| = 0.026$, we construct 3-step CCD. The parameters of our attacks are listed in Table 7 (see Appendix B). Note that, the value $\varepsilon^{(i)}(i = 2, 3)$ is chosen with the step success probability 95%. The time complexity is $2^{209} \cdot N_1 + 2^{213-\alpha_1} \cdot N_2 + 2^{256-\alpha_1-\alpha_2} \cdot N_3 + 2^{214-\alpha_1-\alpha_2-\alpha_3} \approx 2^{136}$, the data complexity is $2^{27} + 2^{25.5} + 2^{26.3} \approx 2^{28}$, and the success probability is $50\% \times (95\%)^2 \approx 45\%$.

Attack on 256-bit Chacha7. For the differential $([\Delta^3_{11}]_0|[\Delta^0_{13}]_{13})$, we construct 4-step CCD. The parameters of our attacks are listed in Table 8 (see Appendix B). Note that, $\varepsilon^{(i)}(i = 2, 3, 4)$ is chosen with the step success probability 95%. The time complexity is $2^{221} \cdot N_1 + 2^{222-\alpha_1} \cdot N_2 + 2^{224-\alpha_1-\alpha_2} \cdot N_3 + 2^{228-\alpha_1-\alpha_2-\alpha_3} \cdot N_4 + 2^{256-\alpha_1-\alpha_2-\alpha_3-\alpha_4} \approx 2^{246.5}$, the data complexity is $2^{26.3} + 2^{25.3} + 2^{24.2} + 2^{22.4} \approx 2^{27}$, and the success probability is $50\% \times (95\%)^3 \approx 43\%$.

Attack on 128-bit Chacha6. For the differential $([\Delta^3_{11}]_0|[\Delta^0_{13}]_{13})$ with $|\varepsilon_d| = 0.026$, we construct 3-step CCD. The parameters of our attacks are listed in Table 9 (see Appendix B). Note that, the value $\varepsilon^{(i)}(i = 2, 3)$ is chosen with the step success probability 95%. The time complexity is $2^{77} \cdot N_1 + 2^{81-\alpha_1} \cdot N_2 + 2^{85-\alpha_1-\alpha_2} \cdot N_3 + 2^{128-\alpha_1-\alpha_2-\alpha_3} \approx 2^{105}$, the data complexity is $2^{27.9} + 2^{24.6} + 2^{23.3} \approx 2^{28}$, and the success probability is $50\% \times (95\%)^2 \approx 45\%$.

5 Conclusions

In this paper, we extend the approach of Aumasson et al. by considering a new type of distinguishers, named (column and row) chaining distinguishers, which can efficiently make use of the biases of multiple differential trails and the matrix structure of the cipher. Besides, we find new high probability second-order differential trails that are not covered by the previous results, some of which are employed in our attack. The notion of PNB is generalized to that of probabilistic neutral vectors (PNV), which investigate the properties of the underlying function when more than one input bit are flipped simultaneously and include the PNB as a special case. It is shown that the set of PNV is no smaller than that of PNB. Based on these findings, we construct improved key recovery attacks on reduced-round Salsa20 and ChaCha, repectively. Both time and data complexities of our new attacks are smaller than those of the best former results.

Acknowledgement. The authors gratefully acknowledge the anonymous referees, whose comments helped to improve the presentation.

References

1. Bernstein, D.J.: Salsa20. Technical Report 2005/025, eSTREAM, ECRYPT Stream Cipher Project, `http://cr.yp.to/snuffle.html`

2. The eSTREAM project, `http://www.ecrypt.eu.org/stream/`
3. Bernstein, D.J.: Salsa20/8 and Salsa20/12. Technical Report 2006/007, eSTREAM, ECRYPT Stream Cipher Project, `http://cr.yp.to/snuffle/812.pdf`
4. Bernstein, D.J.: ChaCha, a variant of Salsa20, `http://cr.yp.to/chacha.html`
5. Crowley, P.: Truncated differential cryptanalysis of five rounds of Salsa20. In: Stream Ciphers Revisited - SASC 2006 (2006)
6. Velichkov, V., Mouha, N., De Cannière, C., Preneel, B.: UNAF: A Special Set of Additive Differences with Application to the Differential Analysis of ARX. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 287–305. Springer, Heidelberg (2012)
7. Fischer, S., Meier, W., Berbain, C., Biasse, J.-F., Robshaw, M.J.B.: Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 2–16. Springer, Heidelberg (2006)
8. Tsunoo, Y., Saito, T., Kubo, H., Suzaki, T., Nakashima, H.: Differential cryptanalysis of Salsa20/8. In: The State of the Art of Stream Ciphers - SASC 2007 (2007)
9. Aumasson, J.-P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of Latin dances: analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 470–488. Springer, Heidelberg (2008)
10. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. IEEE Transactions on Computers 34(1), 81–85 (1985)

Appendix A: CCD of Salsa20/5 and Salsa20/6

For Salsa20/5, we get the significant key bits sets:
$A_1 = \{$0, 1, 32, 33, 34, 35, 36, 37, 38, 74, 75, 76, 77, 78, 84, 85, 86, 87, 88, 89, 90, 129, 130 131, 132, 133, 134, 135, 143, 144, 145, 146, 147, 148, 149, 208, 209, 210, 211, 212 244, 245, 246, 247, 248 $\}$,
$A_2 = \{$21, 22, 23, 24, 25, 26, 27, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 68, 69, 70, 71, 72, 80, 81, 82, 83, 128, 139, 140, 141, 142, 200, 201, 202, 203, 204, 205, 206, 236, 237, 238, 239, 240, 241, 242 $\}$,
$A_3 = \{$ 2, 3, 4, 5, 6, 7, 8, 14, 15, 16, 17, 18, 19, 20, 43, 44, 45, 46, 47, 96, 97, 98, 99, 100, 101, 102, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 207, 213, 214, 215, 216, 217, 218, 219$\}$,
$A_4 = \{$ 9, 10, 11, 12, 13, 28, 29, 30, 31, 50, 51, 52, 91, 92, 93, 103, 104, 105, 106, 107, 108, 109, 110, 111, 136, 137, 138, 181, 182, 183, 184, 185, 186, 187, 188, 189, 192, 193, 194, 195, 196, 220, 221, 222, 223 $\}$,
$A_5 = \{$ 39, 40, 41, 48, 49, 112, 113, 114, 115, 116, 117, 151, 152, 153, 154, 155, 156, 157, 160, 161, 162, 190, 191, 197, 198, 199, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 243, 249, 250, 251, 252, 253 $\}$.
For each differential $([\Delta_3^3]_{15}|[\Delta_6^0]_0)$, $([\Delta_3^3]_9|[\Delta_6^0]_0)$, $([\Delta_8^3]_{11}|[\Delta_7^0]_2)$, $([\Delta_8^3]_{20}|[\Delta_7^0]_0)$ and $([\Delta_{12}^3]_{23}|[\Delta_7^0]_1)$, construct the single distinguisher $D_j (j = 1, 2, ..., 5)$ with the non-significant key bits being set to a fixed value. And the D_j effectively depends on subkey $K'_j = \{k_i | i \in \cup_{l=1}^{j} A_l\}$.

For Salsa20/6, we get the significant key bits sets:

A_1 = {0, 1, 2, 3, 31, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 74, 75, 76, 77, 78, 79, 80, 81, 96 108, 109, 110, 111, 112, 113, 114, 122, 123, 124, 125, 126, 127, 185, 186, 187, 188 189, 190, 191, 217, 218, 219, 220, 221, 222, 223, 230, 231, 232, 233, 234, 235, 236 },
A_2 = { 8, 9, 10, 11, 12, 13, 14, 35, 36, 37, 67, 68, 69, 70, 71, 72, 73, 85, 86, 87, 88, 89, 90, 91, 130, 131, 132, 133, 134, 135, 136, 166, 167, 168, 169, 170, 171, 172, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 243, 244, 245, 246, 247, 248, 249, 250},
A_3 = { 4, 5, 21, 22, 23, 24, 25, 32, 33, 34, 58, 59, 60, 61, 62, 63, 64, 82, 92, 93, 94, 95, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 198, 199, 200, 201, 202, 203, 204, 205, 206, 237, 238, 239, 240, 241},
A_4 = { 7, 15, 18, 19, 20, 50, 51, 52, 53, 54, 55, 65, 66, 139, 140, 141, 142, 143, 144, 145, 175, 176, 177, 178, 179, 180, 181, 192, 193, 194, 195, 224, 225, 226, 227, 252, 253, 254, 255}.

For each differential $([\Delta_6^4]_{26}|[\Delta_7^0]_{31})$, $([\Delta_1^4]_3|[\Delta_7^0]_{29})$, $([\Delta_1^4]_{26}|[\Delta_7^0]_{13})$, $([\Delta_1^4]_{12}|[\Delta_7^0]_{13})$, construct the single distinguisher $D_j (j = 1, 2, 3, 4)$ with the non-significant key bits being set to a fixed value. And the D_j effectively depends on subkey $K'_j = \{k_i | i \in \cup_{l=1}^{j} A_l\}$.

Appendix B: Parameters for Our Attacks

Table 3. Different parameters for our attack on 256-bit Salsa 20/5

i	Differential	ns	ε_d	ε_a	ε	α	Data
1	$([\Delta_3^3]_{15}\|[\Delta_6^0]_0)$	211	0.995	0.677	0.674	45	2^8
2	$([\Delta_3^3]_9\|[\Delta_6^0]_0)$	165	0.929	0.670	0.622	46	2^8
3	$([\Delta_8^3]_{11}\|[\Delta_7^0]_2)$	121	0.999	0.737	0.736	44	2^8
4	$([\Delta_8^3]_{20}\|[\Delta_7^0]_0)$	76	0.971	0.947	0.921	45	2^7
5	$([\Delta_{12}^3]_{23}\|[\Delta_7^0]_1)$	32	0.918	0.943	0.866	44	2^7

Table 4. Different parameters for our attack on 256-bit Salsa 20/6

i	Differential	ns	ε_d	ε_a	ε	α	Data
1	$([\Delta_6^4]_{26}\|[\Delta_7^0]_{31})$	196	0.201	0.680	0.137	60	2^{13}
2	$([\Delta_1^4]_3\|[\Delta_7^0]_{29})$	140	0.1113	0.664	0.075	56	2^{15}
3	$([\Delta_1^4]_{26}\|[\Delta_7^0]_{13})$	93	0.110	0.771	0.085	47	2^{14}
4	$([\Delta_1^4]_{12}\|[\Delta_7^0]_{13})$	54	0.183	0.801	0.147	39	2^{13}

Table 5. Different parameters for Aumasson's attack on 256-bit Salsa 20/7

γ	ns	ε_a	ε	α	Time	Data
1.0	39	1.000	0.1310	31	2^{230}	2^{13}
0.9	97	0.655	0.0860	88	2^{174}	2^{15}
0.8	103	0.482	0.0634	93	2^{169}	2^{16}
0.7	113	0.202	0.0265	101	2^{162}	2^{19}
0.6	124	0.049	0.0064	108	2^{155}	2^{23}
0.5	131	0.017	0.0022	112	2^{151}	2^{26}

Table 6. Parameters for our attack on 128-bit Salsa 20/7

i	$\gamma^{(i)}$	ns_i	$\varepsilon^{(i)}$	α_i	N_i
1	0.40	38	0.0059	2	2^{19}
2	0.42	36	0.0105	4	$2^{17.5}$
3	0.45	34	0.0165	6	$2^{16.5}$
4	0.60	30	0.0359	18	$2^{15.5}$

Table 7. Parameters for our attack on 256-bit Chacha6

i	$\gamma^{(i)}$	ns_i	$\varepsilon^{(i)}$	α_i	N_i
1	0.60	147	0.00048	4	2^{27}
2	0.66	143	0.00091	8	$2^{25.5}$
3	0.75	139	0.00171	120	$2^{26.3}$

Table 8. Parameters for our attack on 256-bit Chacha7

i	$\gamma^{(i)}$	ns_i	$\varepsilon^{(i)}$	α_i	N_i
1	0.50	35	0.00059	3.8	$2^{26.3}$
2	0.53	34	0.00080	3.5	$2^{25.3}$
3	0.55	32	0.00127	5	$2^{24.2}$
4	0.58	28	0.00280	9	$2^{22.4}$

Table 9. Parameters for our attack on 128-bit Chacha6

i	$\gamma^{(i)}$	ns_i	$\varepsilon^{(i)}$	α_i	N_i
1	0.50	51	0.00034	4	$2^{27.9}$
2	0.56	47	0.00114	5.5	$2^{24.6}$
3	0.65	43	0.00281	25	$2^{23.3}$