Chosen IV Cryptanalysis on Reduced Round ChaCha and Salsa

Subhamoy Maitra

Applied Statistics Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700 108, India
subho@isical.ac.in

Abstract. Recently, ChaCha20 (the stream cipher ChaCha with 20 rounds) is in the process of being a standard and thus it attracts serious interest in cryptanalysis. The most significant effort to analyse Salsa and ChaCha had been explained by Aumasson et al long back (FSE 2008) and further, only minor improvements could be achieved. In this paper, first we revisit the work of Aumasson et al to provide a clearer insight of the existing attack (2^{248} complexity for ChaCha7, i.e., 7 rounds) and showing certain improvements (complexity around 2^{243}) by exploiting additional Probabilistic Neutral Bits. More importantly, we describe a novel idea that explores proper choice of IVs corresponding to the keys, for which the complexity can be improved further (2^{239}). The choice of IVs corresponding to the keys is the prime observation of this work. We systematically show how a single difference propagates after one round and how the differences can be reduced with proper choices of IVs. For Salsa too (Salsa20/8, i.e., 8 rounds), we get improvement in complexity, reducing it to $2^{245.5}$ from $2^{247.2}$ reported by Aumasson et al.

Keywords: Stream Cipher, ChaCha, Salsa, Non-Randomness, Probabilistic Neutral Bit (PNB), ARX Cipher.

1 Introduction

The Salsa20 [2] stream cipher has been designed by Bernstein in 2005 as a candidate for eStream [12] and Salsa20/12 has been accepted in the eStream software portfolio. There are several works that studied the cryptanalysis of Salsa [4, 5, 10, 1, 6, 9, 8, 7] and these works show weaknesses of this cipher in reduced rounds. The central idea in this field of cryptanalysis is as follows.

– Apply some input difference at the initial state and then investigate for biases at some output.
– Once one can proceed a few rounds forward as above, it may be possible to get back a few rounds from a final state to obtain further non-randomness.

The ChaCha [3] stream cipher has been proposed in early 2008 to conjecturally provide better diffusion and cryptanalytic resistance than Salsa. Though ChaCha has been designed long back, the cipher got renewed attention in recent time due to its deployment in several applications of Google as evident from the following news report [11]: “Given recent attacks against older, commonly-used encryption modes RC4 and CBC, the Google team began implementing new algorithms - ChaCha 20 for symmetric encryption and Poly1305 for authentication in OpenSSL and NSS in March 2013.” The only significant cryptanalysis of reduced round Salsa and ChaCha has been presented long back in [1] that introduced Probabilistic Neutral Bits (PNBs) and the works [9, 7] achieved only some incremental advancements over [1]. The idea of Column Chaining Distinguisher (CCD) [9] could only provide minor advantage over the complexities described in [1] for both Salsa and ChaCha. In [7], an interesting observation regarding round reversal of Salsa has been studied, but no significant cryptanalytic improvement could be obtained using this method. However, the work of [7] revisits the attack of [1] in detail, providing some corrections to the values of the parameters that constitute the attack complexity.

Contribution In this paper, we attempt to obtain significant improvement in the cryptanalysis of ChaCha over the work of [1]. First, in Section 1.2, we put a disciplined effort to study the same method of [1] in greater details. This is in line of [7], where the work of [1] has been revisited too in the context of Salsa. We perform detailed experiments and exploit more PNBs than in [1] to obtain better results. In [1], the time complexity of the attack against ChaCha7 (7 round ChaCha) was estimated as 2^{248}, though a careful analysis shows that it is actually slightly better (less), which is $2^{246.71}$. The effort of [9] with the idea of CCD required $2^{246.5}$ complexity and thus one may note that the improvement is indeed minor. In Section 2, with additional PNBs and following the similar method as in [1], we show that the complexity can be reduced to $2^{242.82}$.

Next we show how our idea of exploiting specific IVs corresponding to the secret key helps in improving the attack of [1]. While the cryptanalysis of reduced round Salsa and ChaCha has been presented long back [1], this observation of choosing selected IVs could not be discovered earlier. The cryptanalytic technique presented in [1] depends on some biases related to certain output differences in the forward direction given the input difference(s). Our idea improves such biases significantly. Using this, in Section 3, we show that for proper choices of IVs, the time complexity of ChaCha7 cryptanalysis reduces to $2^{238.94}$.

To explain further, we consider the scenario when the number of differences after the quarterround is the same as in the case when the modulo addition $+$ (nonlinear) is replaced by the linear operation $\oplus$. The differential thus passes with probability one. This happens for proper choices of IV words given the key words. One may also refer to these differences as conditional differences.

We apply the similar technique for Salsa in Section 4. For Salsa20/8, the complexity described in [1] was 2^{251}, but a detailed study in [7] shows that it is actually better, i.e., $2^{247.2}$. The idea of [9] using CCD required 2^{250}, and naturally the improvement was not significant. Our technique, considering the properly chosen IVs, improves the complexity to $2^{245.5}$.

Before proceeding further, let us explain the structure of ChaCha stream cipher as provided in the draft towards standardization [13]. We will come back to the description related to Salsa in Section 4.

1.1 Description of ChaCha

The cipher state is of 16 words, each 32-bit and it can be written in 4×4 matrix format as follows:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & v_0 & v_1 & v_2 \end{pmatrix}.$$

The rightmost matrix shows the initial state, that takes four predefined constants $c_0, \ldots, c_3$ as $c_0 =$ `0x61707865`, $c_1 =$ `0x3320646e`, $c_2 =$ `0x79622d32`, $c_3 =$ `0x6b206574`, 256-bit key $k_0, \ldots, k_7$, 32-bit block counter t_0 and 96-bit nonce v_0, v_1, v_2.

Primitive nonlinear operation here is the quarterround function. Each quarterround(a, b, c, d) consists of four ARX rounds, each of which comprises of addition (A), cyclic left rotation (R) and XOR (X) operation (one each) as given below.

$$\left.\begin{array}{l} a = a + b;\ d = d \oplus a;\ d = d \lll 16; \\ c = c + d;\ b = b \oplus c;\ \ b = b \lll 12; \\ a = a + b;\ d = d \oplus a;\ d = d \lll 8; \\ c = c + d;\ b = b \oplus c;\ \ b = b \lll 7; \end{array}\right\} \qquad (1)$$

Each columnround works as four quarterrounds on each of the four columns of the state matrix and each diagonalround consists of four quarterrounds on each of the four diagonals. In ChaCha20, ten times the rowround and ten times the diagonalround are applied alternatively on the initial state (total 20 times).

In each of the odd rounds, we first apply quarterround on all the four columns in the following order. This is a complete columnround.

$$\mathsf{quarterround}(x_0, x_4, x_8, x_{12}),\ \mathsf{quarterround}(x_1, x_5, x_9, x_{13}),$$
$$\mathsf{quarterround}(x_2, x_6, x_{10}, x_{14}),\ \text{and}\ \mathsf{quarterround}(x_3, x_7, x_{11}, x_{15}).$$

In each of the even rounds, we consider the order

$$\mathsf{quarterround}(x_0, x_5, x_{10}, x_{15}),\ \mathsf{quarterround}(x_1, x_6, x_{11}, x_{12}),$$
$$\mathsf{quarterround}(x_2, x_7, x_8, x_{13}),\ \text{and}\ \mathsf{quarterround}(x_3, x_4, x_9, x_{14}).$$

This describes a complete diagonalround.

By $X^{(r)}$, we mean that r such rounds have been applied (in total, alternatively the columnround in odd rounds and diagonalround in even rounds, the initial round applied is considered as the round 1) on the initial state X. Here $X^{(0)}$ is the initial state X. Finally, after R rounds we have $X^{(R)}$. Then a keystream block of 16 words or 512 bits is obtained as

$$Z = X + X^{(R)}.$$

For ChaCha20, there are 20 rounds, i.e., $R = 20$. It is quite natural that less rounds achieve higher speed and conjecturally, more rounds provide higher security.

Each round of ChaCha is reversible as the state-transition operations are reversible. If $X^{(r+1)} = \mathsf{round}(X^{(r)})$, then $X^{(r)} = \mathsf{reverseround}(X^{(r+1)})$, where reverseround is the inverse of round. The reverse of each quarterround is defined as below.

$$\left.\begin{array}{l} b = b \ggg 7;\quad b = b \oplus c;\ \ c = c - d; \\ d = d \ggg 8;\quad d = d \oplus a;\ a = a - b; \\ b = b \ggg 12;\ b = b \oplus c;\ \ c = c - d; \\ d = d \ggg 16;\ d = d \oplus a;\ a = a - b; \end{array}\right\} \tag{2}$$

Consider that one obtains a state $X^{(1)}$ after one round of ChaCha. Now to know whether it is a valid state after one round, one needs to come back by one reverse round and then check whether the constants in the first row are indeed the specified ones.

Here x_i is the i-th word of the matrix X. Further, by $x_{i,j}$, we mean the j-th bit (0-th bit is the least significant bit) of x_i. Given two states $X^{(r)}, X'^{(r)}$, we denote $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$. By $\Delta_{i,j}^{(r)} = x_{i,j}^{(r)} \oplus x_{i,j}'^{(r)}$, we mean the difference between two states at the j-th bit of the i-th word after r many rounds. For example, '$\Delta_{13,5}^{(0)} = 1$' means that we have two initial states $X^{(0)}, X'^{(0)}$ that differ at the 5-th bit of the 13-th word. Towards cryptanalysis, we are interested to input a difference at the initial state (call it Input Differential or $\mathcal{ID}$) and then try to obtain certain biases corresponding to combinations of some output bits (call it Output Differential or $\mathcal{OD}$). In this direction, one can compute

$$\Pr(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_d),$$

where the probability is estimated for a fixed key and all the possible choices of nonces and counter words v_0, v_1, v_2 and t_0 respectively, other than the constraints imposed due to the input differences on the nonces or counters. For ChaCha and Salsa, most of the biases in literature are discovered through experimentation, and those are estimated by a large number of experiments.

1.2 Revisiting PNBs [1]

The idea that we discuss in this section applies for both ChaCha and Salsa. This is a brief explanation of what has been done in [1] and our discussion is in line of [7] where the presentation was in the context of Salsa.

The method described here is a known plain text only attack. The 512-bit key stream of ChaCha$R$ or Salsa20/R is $X + X^{(R)}$. In this model $X + X^{(R)}$ is completely available to the attacker. However, the 256 key bits are not available, thus only rest 256 bits of X are known. The motivation is to guess the 256 key bits of ChaCha$R$ or Salsa20/R with a key search complexity less than 2^{256}, the complexity for exhaustive search.

The concept of Probabilistic Neutral Bits (PNBs) in [1] was exploited in this framework. Before describing our parameters for an improved attack complexity, we first describe the generic attack using PNBs on ChaCha (similar in case of Salsa). For detailed understanding and formal definitions in this area, one may refer to [1, Section 3].

Bias in forward direction (ϵ_d). Let X, X' be two valid initial states with a given $\mathcal{ID}$ $\Delta_{i,j}^{(0)} = 1$, for which we observe a high bias ϵ_d in an $\mathcal{OD}$ $\Delta_{p,q}^{(r)}$ after $r < R$ ChaCha rounds. Thus, $\Pr(\Delta_{p,q}^{(r)} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_d)$, where $\Delta^{(r)} = X^{(r)} \oplus X'^{(r)}$. The two keystream blocks after R rounds are given by $Z = X + X^{(R)}$ and $Z' = X' + X'^{(R)}$.

Probabilistic Neutral Bits (PNBs) and related bias (γ). Consider that in both X and X', we complement a particular key bit position κ to yield the states $\overline{X}$ and $\overline{X'}$ respectively. Next, we reverse the states $Z - \overline{X}$ and $Z' - \overline{X'}$ by $R - r$ rounds to yield the states Y and Y' respectively. Let $\Gamma_{p,q} = Y_{p,q} \oplus Y'_{p,q}$. Given the $\mathcal{ID}$, if the bias in the event $(\Delta_{p,q}^{(r)} \oplus \Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)} = 1)$ is high, i.e., $\Delta_{p,q}^{(r)} = \Gamma_{p,q}$ with high probability, then we call the key bit κ a *Probabilistic Neutral Bit* (PNB). If $\Pr(\Delta_{p,q}^{(r)} \oplus \Gamma_{p,q} = 0 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \gamma_\kappa)$, then γ_κ is called the *neutrality measure* of the key bit. One should run this experiment for each key bit several times over randomly chosen nonces and counter. We experiment this for 2^{26} samples corresponding to each key bit, which is more than enough to identify the biases. Repeating this for all the 256 key bits, a subset of the key bits can identified, which are called the PNBs. Typically, a threshold probability $\frac{1}{2}(1 + \gamma)$ is chosen to filter the PNBs. If $\gamma_\kappa \geq \gamma$, then the key bit κ is included in the set of the PNBs. Suppose the size of this subset is n and therefore the number of non-PNB bits are $m = 256 - n$. The main idea behind the key recovery is to search these two sets separately.

Biases in reverse direction (ϵ, ϵ_a). After the set of PNBs is determined, the actual attack considers search over the key bits which are not PNBs. By considering a distinguisher, it is possible to identify when the correct keys have appeared. For practical simulations, it is not possible to complete this attack as the complexity is very high. Nevertheless, one can obtain certain biases to estimate the complexity of the attack and that can be achieved by trying out the following.

While studying the PNBs, in X and X', we complemented a particular key bit position κ to yield the states $\overline{X}$ and $\overline{X'}$ respectively. However, in this case, we assign random values to all the PNBs and keep the correct values of the keys in other places. That is, we assign correct key values to the m non-PNB key bits and assign random binary values to the n PNB key bits in both X and X' to yield the states $\hat{X}$ and $\hat{X}'$ respectively. Then we reverse the states $Z - \hat{X}$ and $Z' - \hat{X}'$ by $R - r$ rounds to yield the states $\hat{Y}$ and $\hat{Y}'$ respectively. Let $\hat{\Gamma}_{p,q} = \hat{Y}_{p,q} \oplus \hat{Y}'_{p,q}$ and $\Pr(\hat{\Gamma}_{p,q} = 1 | \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \hat{\epsilon})$. A higher absolute value of $\hat{\epsilon}$ identifies that the PNBs have been chosen properly and even without knowing the n PNBs it is possible to guess the rest of the key bits which are not PNBs.

Next, we assign random binary values to all the 256 key bits in both X and X' to yield the states $\widetilde{X}$ and $\widetilde{X}'$ respectively. Then we reverse the states $Z - \widetilde{X}$ and $Z' - \widetilde{X}'$ by $R - r$ rounds to yield the states $\widetilde{Y}$ and $\widetilde{Y}'$ respectively. Let $\widetilde{\Gamma}_{p,q} = \widetilde{Y}_{p,q} \oplus \widetilde{Y}'_{p,q}$ and $\Pr(\widetilde{\Gamma}_{p,q} = 1 | \Delta^{(0)}_{i,j} = 1) = \frac{1}{2}(1+\widetilde{\epsilon})$.

In actual key recovery attack, if the biases $\hat{\epsilon}$ and $\widetilde{\epsilon}$ can be efficiently distinguished, i.e., if the gap between the biases is significant with $\widetilde{\epsilon} \approx 0$ (as it corresponds to a random event and should not have any bias), then we can conclude that the assignment $\hat{X}$ yields the correct values for the non-PNB bits. This is what that needs to be experimented while choosing the set of PNBs. This also provides the estimate of $\hat{\epsilon}$ that will be involved in the obtaining the complexity. To follow the same notation as in [1], we denote $\epsilon = \hat{\epsilon}$ in following discussion.

In [1], the bias in the event $(\hat{\Gamma}_{p,q} = \Delta^{(r)}_{p,q})$ is denoted by ϵ_a.

Estimation of biases using median. In [1], the estimation of this bias is as follows. The key is fixed and one can vary the nonces and the counters to calculate one ϵ_a. Then we consider many randomly chosen keys to obtain a set of ϵ_a's and finally compute the median ϵ^*_a's from this set. Similarly, the median ϵ^*_d is estimated from the values of several ϵ_d's corresponding to different keys. Finally one can estimate ϵ^* as the median value of ϵ's. The idea of using median is that, one can guarantee that the estimated probabilities will work for at least half of the keys.

It was noted in [1] that one can approximate ϵ^* as $\epsilon^*_d \cdot \epsilon^*_a$. We have also observed this trend. Thus, it is very clear that if one can come up with a set of IVs corresponding to a specific key for which ϵ_d can be increased substantially, then ϵ should also increase. Consequently, the complexity of the attack will decrease. This is used for the improved cryptanalysis of ChaCha7 and Salsa20/8 as described in Sections 3, 4 respectively.

Estimating the complexity of the cryptanalysis. Following [1], if the number of samples used is N and if the probability of false alarm is $P_{fa} = 2^{-\alpha}$, the complexity of the attack is then given by

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}, \tag{3}$$

where the required number of samples is

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

for probability of non-detection $P_{nd} = 1.3 \times 10^{-3}$.

2 Cryptanalysis of ChaCha7 as in [1] and exploiting more PNBs

Let us start with the parameters as described in [1]. As in [1], we consider $R = 7$ and we move $r = 3$ forward rounds, and come back $R - r = 7 - 3 = 4$ rounds from the 7-th round. Throughout this paper, we consider the $\mathcal{ID}$-$\mathcal{OD}$ pair $(\Delta^{(0)}_{13,13}, \Delta^{(3)}_{11,0})$ that has been exploited in [1]. Let us now provide the estimation that has been detailed in [1] for the attack complexity.

Estimate 1 *[1] In this case, the experiments show $\frac{1}{2}(1+\epsilon^*_d) = 0.513$, i.e., $\epsilon^*_d = 2\times 0.013 = 0.026$. Taking $\gamma = 0.5$, $n = 35$ PNBs (see Table 1 for exact key bits) the estimation in [1] had been $\epsilon^* = 0.00059, \epsilon^*_a = 0.023$ as the median bias over all keys. In this case, $m = 256 - 35 = 221$. Taking $\alpha = 11$, one obtains $N \approx 2^{27.03} \approx 2^{27}$ (the data complexity) and thus, the total complexity becomes $2^{221+27} + 2^{256-11}$, which is approximately 2^{248} as described in [1].*

3	6	15	16	31	35	67	68	71	91	92	93
0.58	0.53	0.71	0.56	0.59	0.59	0.74	0.52	0.56	0.95	0.92	0.88
94	95	96	97	98	99	100	103	104	127	136	191
0.82	0.76	0.97	0.94	0.89	0.78	0.59	0.70	0.58	1.00	0.66	0.66
223	224	225	248	249	250	251	252	253	253	255	
0.88	0.68	0.54	1.0	0.1.0	1.0	1.0	1.0	1.0	1.0	1.0	

Table 1. Key-bits κ and the corresponding biases $\gamma_\kappa \geq 0.5$ for the 35 PNBs of the 7-round attack on ChaCha [1]. The index of the key bit κ and the corresponding γ_κ are presented one below the other in each cell.

However, we have got improved results than [1] with our experiments in a completely similar set-up as in [1]. Note that the exact experimental figures related to the number of samples are not available in the paper [1]. We take 2^8 randomly chosen keys and for each of them we take 2^{30} randomly chosen counters and nonces. For each of these keys, we take the average data ϵ and then we consider the median of those ϵ's which comes to be $\epsilon^* = 0.001116$. We believe that the multiplication by two while estimating ϵ^* was missed in [1]. This happens for the estimation of ϵ_a^* too. We obtain $\epsilon_a^* = 0.043572$ which is almost twice than the estimate of [1]. However, we obtain $\epsilon_d^* = 0.0262$, that almost exactly matches with that of [1].

Estimate 2 *With this estimate, keeping all the other parameters same, the complexity of [1] can be revised as follows. In this case, for* $\epsilon^* = 0.001116$, *we obtain* $N \approx 2^{25.19}$ *and thus, the total complexity becomes* $2^{221+25.19} + 2^{256-11} \approx 2^{246.71}$.

In fact, this is almost the same as the the minor improvement proposed in [9] using Column Chaining Distinguisher (CCD) that could achieve the complexity of $2^{246.5}$.

2.1 Adding more PNBs

In [1], the authors commented that "we note that the described complexities may be improved by choosing a smaller γ". However, for a long time (since 2008), it has never been studied how much improvement can exactly be achieved. Given that ChaCha is being considered to be a standard [13] and may have wide deployment, it is necessary to obtain the exact figures. For our study, we look at the PNBs more carefully. We use the same $\mathcal{ID}$-$\mathcal{OD}$ pair $(\Delta^{(0)}_{13,13}, \Delta^{(3)}_{11,0})$. Lowering γ one can find more PNBs and after a few rounds of trial and error we take $\gamma = 0.3$. With this, we obtain additional 10 PNBs, i.e., in total $35 + 10 = 45$ PNBs (see Table 2 for details).

7	17	36	38	72	105	137	156	159	194
0.33	0.35	0.31	0.35	0.40	0.43	0.49	0.47	0.43	0.38

Table 2. Key-bits and the corresponding biases $\gamma_\kappa \geq 0.3$ for the 10 additional PNBs of the 7-round improved attack on ChaCha.

Estimate 3 *Here we have* $m = 256 - 45 = 211$. *We estimate* $\epsilon^* = 0.000132$ *as the median bias over all keys. We also note that* $\epsilon_d^* = 0.025750$ *and* $\epsilon_a^* = 0.005082$ *in this case. Taking* $\alpha = 18$, *we obtain* $N \approx 2^{31.77}$ *and thus, the total complexity becomes* $2^{211+31.77} + 2^{256-18} \approx 2^{242.82}$. *Note that* ϵ^* *is reduced from 0.001116 (as in Estimate 2) to 0.000132 (by a factor of more than 8) here. This increases* N, *but due to the ten additional PNBs, the overall complexity reduces.*

Estimate 4 *We will consider another case with 41 PNBs. In this case, we do not consider 4 PNBs that correspond to the keywords k_1 (i.e., key bits 35, 36, 38) and k_5 (i.e., key bit 191). This case has worse complexity than the case with 45 PNBs, but this is done consciously so that we can explain our improvements over this result in Section 3. Here, $m = 256 - 41 = 215$ and we obtain $\epsilon^* = 0.000404$. Further, $\epsilon_d^* = 0.027418$ and $\epsilon_a^* = 0.014498$ in this case. Taking $\alpha = 17$, we obtain $N \approx 2^{28.49}$ and thus, the total complexity becomes $2^{215+28.49} + 2^{256-17} \approx 2^{243.56}$. One may refer to Figure 1 to see the graphical plot corresponding to this data.*

We present the summary of the comparative results in Table 3 in the concluding section. So far we have studied the scenario for any random key and taking the medians for the calculations guarantee that the attack works for at least half of the keys. However, we discover that if one can properly choose some words of keys and IVs, then ϵ_d can be increased, and thus ϵ will also increase. Following the exact calculations of ϵ_a is quite complicated as it relates to several PNBs in the reverse rounds. However, studying the improvement in the bias ϵ_d during forward rounds is somewhat possible as we start with a single bit input difference. We describe this in the next section.

3 Improved attack on ChaCha7: Choosing proper IVs

We start our explanation following the Figure 1 and Example 1. In the figure we show how the choice of two words of the secret key and one word of the IVs increase ϵ_d significantly than the cases where this choice is not made. This, in turn, increases ϵ in each of the runs and finally ϵ^* to provide an improved attack complexity.

Example 1. We consider $x_5 = x_9 = 0$ and $x_{13} =$ `0xaaaaaaaa`, that is $k_1 = k_5 = 0$, and $v_0 =$ `0xaaaaaaaa`. With this, we observe that $\epsilon^* = 0.002012, \epsilon_a^* = 0.014406$ and $\epsilon_d^* = 0.140344$ as the median bias over all the keys other than the fixed ones. Taking $\alpha = 21$, we obtain $N \approx 2^{24.05}$ and thus, the total complexity becomes $2^{215+24.05} + 2^{256-21} \approx 2^{239.14}$.

3.1 Reducing number of differences after one quarterround

The main issue here is that ϵ_d increases quite significantly with the specific choice of some keywords and the nonces. We are considering the $\mathcal{ID}$ as $\Delta_{13,13}^{(0)}$. The initial state is as follows:

$$\begin{pmatrix} x_0 & \mathbf{x_1} & x_2 & x_3 \\ x_4 & \mathbf{x_5} & x_6 & x_7 \\ x_8 & \mathbf{x_9} & x_{10} & x_{11} \\ x_{12} & \mathbf{x_{13}} & x_{14} & x_{15} \end{pmatrix}$$

and the column marked in bold font is the column that will be affected by the $\mathcal{ID}$ $\Delta_{13,13}^{(0)}$ after one columnround. The words in other columns will remain the same after one round and there will be no difference. To understand how the differences propagate in the words of the marked column depends on the design of quarterround as in (1). Naturally, if one can minimize the differences after the first round, then one may obtain better biases in form of ϵ_d. We try to identify how this happens.

One may note that if the $+$'s (modulo additions over 2^{32}) in (1) are replaced by $\oplus$, then the $\mathcal{ID}$ $\Delta_{13,13}^{(0)}$ generates output differences at 10 places after quarterround is applied over the said column. We try to achieve this scenario in the case of (1) (i.e., with modulo addition) and this depends on the values of and interactions among x_1, x_5, x_9, x_{13}. For ChaCha, we have $x_1 = c_1 =$

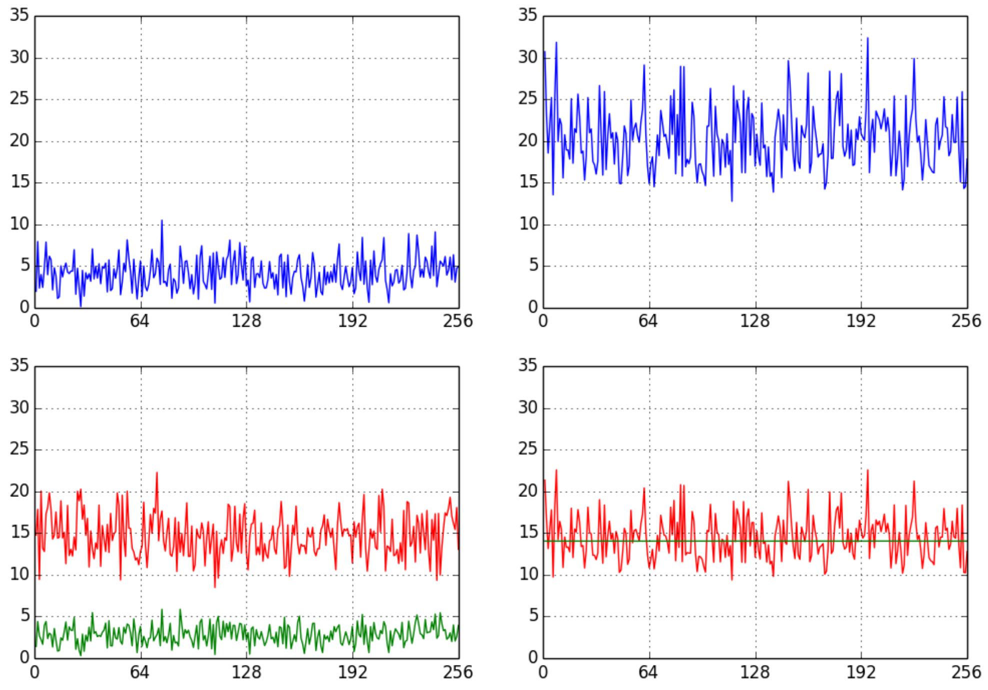

Fig. 1. We have the data corresponding to 256 runs, and the data corresponding to each run is an average of 2^{30} samples. The X axis in each of the four graphs are indices of different runs. Y axis: (i) for random Key/IV (a) Top Left: ϵ (blue), (b) Bottom Left: ϵ_d (green) and ϵ_a (red); (ii) for random Key/IV with the constraints $k_1 = k_5 = 0$ and $v_0 =$ `0xaaaaaaaa` (a) Top Right: ϵ (blue), (ii) Bottom Right: ϵ_d (green) and ϵ_a (red). For better representation, in the plots, ϵ is multiplied by 10^4, ϵ_a by 10^3 and ϵ_d by 10^2. This is for the $\mathcal{ID}$-$\mathcal{OD}$ pair ($\Delta^{(0)}_{13,13}$, $\Delta^{(3)}_{11,0}$). One may note how the values of ϵ_d (green) and ϵ (blue) increase in the right graph than the left graph and ϵ_a remains at the same level.

`0x3320646e`, a constant. As explained above, we consider $x_5 = x_9 = 0$ and $x_{13} =$ `0xaaaaaaaa`, that provides only 10 differences after one quarterround.

Taking $x_1 =$ `0x3320646e`, and experimenting with random choices for x_5, x_9, x_{13}, we have noted that the number of differences after one quarterround is 20.5 on an average (the experimental results are similar in case we modify the constant at x_1 too. We run the experiments 16 times, where the average is taken over 2^{32} data, for all the distinct values of the IV word x_{13}, in each run.

We have noted that for random case (considering all the values of x_{13}), with $\mathcal{ID}$-$\mathcal{OD}$ pair ($\Delta^{(0)}_{13,13}$, $\Delta^{(3)}_{11,0}$), $\epsilon_d \approx 0.026$. However, for the special choices, such that there are only 10 differences after one quarterround, we obtain a much better bias (more than 5 times), $\epsilon_d \approx 0.14$. As one can approximate $\epsilon^* = \epsilon^*_a \cdot \epsilon^*_d$, considering ϵ^*_a remains same given the fixed set of PNBs, we obtain significant improvement in ϵ^*, that in turn reduces the time complexity of the cryptanalysis.

3.2 Choosing IVs corresponding to the keys

To have the number of differences exactly 10 after one quarterround, we studied a scenario as follows. First the constant at x_1 is fixed as in the description of ChaCha. Next we fix $x_5 = k_1$ and $x_9 = k_5$ at a particular value. Then we find out what are the IVs for which we obtain exactly 10 differences after one quarterround given the $\mathcal{ID}$ $\Delta^{(0)}_{13,13}$. We experimented with 2^{11} random choices of k_1, k_5 and in each case, we have studied how many IVs (out of total 2^{32} choices of $x_{13} = v_0$) are there for which we obtain only 10 differences after one quarterround. We note that there are 373 cases (out of 2048), where for some choices of k_1, k_5, there is not a single IV for which the condition is satisfied. However, the average, median and the maximum are respectively $2^{27.08}, 2^{26.73}$ and $2^{28.98}$ out of 2^{32} possible cases. One may note that if we consider more than 10 differences, then we will get valid IV's x_{13} for the fixed keywords x_5, x_9 where we were not getting any IVs for the difference 10. Subsequently, the value of ϵ_d will be decreased as the number of differences grow.

Now let us relate it with the the parameter N as calculated in (4). This is dependent on the value of ϵ^* and given a value of the order of $\epsilon^* = 0.002012$, we get $N \approx 2^{24.05}$. Thus, it is very clear that even if the other IVs are fixed, then also it is possible to obtain enough samples in at least half of the choices of keys given the median value of $2^{26.73}$, which is larger than N.

Moreover, it is possible to obtain high bias for ϵ^*_d even when the number of differences is little more than 10 after one quarterround. In such a case we can proceed with the attack for almost all the keys. A mathematical study of the characterization of differences after a single quarterround would be an interesting combinatorial problem, though it may not be easy due to the complicated quarterround structure in ChaCha.

3.3 How to choose the PNBs to mount the attacks

We need to obtain the exact values of k_1, k_5 along with the constant c_1 to decide v_0. We know that there are n PNBs that we do not search initially, but search for the other $256 - n$ key bits exhaustively. If the PNBs do not fall inside the bits of k_1, k_5 then we need to search these 64 bits as a part of the exhaustive search. We may preprocess this part. That is, for all the 2^{64} choices of k_1, k_5, we need to search and store the IV's that are suitable for the attack so that we obtain a higher value of ϵ^*_d, i.e., that provide a low value of number of differences after a quarterround. The total time and space complexity of this is 2^{96}, which is much less than the complete attack complexity.

Note that we have first made experiments with the $n = 35$ PNBs [1] and presented it in Estimate 2. Then, to get an improved (reduced) attack complexity (in Estimate 3), we added 10 more PNBs, i.e., total $35 + 10 = 45$ PNBs. Further, out of those PNBs, we removed 4 (in Estimate 4) so that there is no PNB from the keywords k_1, k_5. Thus, we provide results with $n = 41$ PNBs in two cases, (i) when the IV's are chosen without any constraint (Estimate 4) and (ii) when the IVs are properly chosen with the constraint that after one quarterround the number of differences become small, i.e., 10 only (Estimate 5).

As we are following in all the cases for ChaCha, we take the $\mathcal{ID}$-$\mathcal{OD}$ pair $(\Delta^{(0)}_{13,13}, \Delta^{(3)}_{11,0})$. Now we choose random secret keys in each run. Thus, x_5, x_9 is fixed in each run. Corresponding to that we try out all the possible IVs x_{13} (out of total 2^{32}) such that we obtain 10 differences in the said column after one quarterround. We try with 320 such runs and in each case we choose a key randomly and fixed that. Indeed there are a few cases (63 out of 320) where we do not obtain any such IV to mount the attack.

Estimate 5 *Considering all the 320 cases, we take the median. With this, we observe that* $\epsilon^* = 0.002158, \epsilon^*_a = 0.015862$ *and* $\epsilon^*_d = 0.136778$ *as the median bias over all the keys. Taking* $\alpha = 22$, *we obtain* $N \approx 2^{23.89}$ *and thus, the total complexity becomes* $2^{215+23.89} + 2^{256-22} \approx 2^{238.94}$. *As we consider the median bias over the complete data, this complexity works for more than half of the keys. For the 63 keys, where we do not obtain 10 differences with any IV, the attack can be mounted considering more than 10 differences in the said column after one* quarterround.

However, to run the experiment with each possible key (without considering the PNBs), we have to consider each possible pair of keywords k_1, k_5. The sets of IVs that should be properly chosen with such a pair of keywords vary too. However, as the number of IVs corresponding to $x_{13} = v_0$ is 2^{32}, it is enough to have these many samples of the keystream. This is a nominal increase in the data complexity that is insignificant to the time complexity of the attack. The scenario is the same for Salsa as we will discuss in the next section.

4 The impact on Salsa

We have the following initial structure in Salsa.

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

where the matrix at right shows the initial state, that takes four predefined constants $c_0, \dots, c_3$ (same values as in ChaCha), 256-bit key $k_0, \dots, k_7$, 64-bit nonce v_0, v_1 and 64-bit counter t_0, t_1. We will refer to the nonce and counter words as IV words in the following discussion. For Salsa, the quarterround is as follows, which is much simpler than ChaCha.

$$\left.\begin{aligned} b &= b \oplus ((a + d) \lll 7), \\ c &= c \oplus ((b + a) \lll 9), \\ d &= d \oplus ((c + b) \lll 13), \\ a &= a \oplus ((d + c) \lll 18). \end{aligned}\right\} \tag{5}$$

In each round, one applies quarterround on all the four columns in the following order:

quarterround(x_0, x_4, x_8, x_{12}), quarterround(x_5, x_9, x_{13}, x_1), quarterround$(x_{10}, x_{14}, x_2, x_6)$, and quarterround$(x_{15}, x_3, x_7, x_{11})$, and then a transpose$(X)$ as follows:

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \rightarrow X^T = \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}.$$

As described earlier, by $X^{(r)}$, we mean that r such rounds have been applied on the initial state X. Here $X^{(0)}$ is the initial state and after R rounds we have $X^{(R)}$. Then a keystream block of 16 words or 512 bits is obtained as $Z = X + X^{(R)}$. While for Salsa20, $R = 20$, the one accepted in eStream [12] software portfolio is Salsa20/12, where $R = 12$.

In [1], the attack has been studied on Salsa20/8, i.e., on reduced round Salsa where $R = 8$, and 4 forward rounds and 4 reverse rounds have been considered. In this case, the $\mathcal{ID}$-$\mathcal{OD}$ pair had been taken as $(\Delta^{(0)}_{7,31}, \Delta^{(4)}_{1,14})$ and the experimental results showed $\epsilon^*_d = 0.1314$.

If the $+$'s (modulo addition over 2^{32}) in (5) is replaced by $\oplus$, then the $\mathcal{ID}$ $\Delta^{(0)}_{7,31}$ generates output differences at 4 places after quarterround is applied over the final column. In fact, with the actual quarterround as in (5), one obtains the value 5 on an average. Thus here the minimum value is close to the average case and the difference is less significant than the case of ChaCha.

Here the final column contains $x_3 = k_2$, $x_7 = v_1$, $x_{11} = k_4$ and $x_{15} = c_3 =$ `0x6b206574`. If we consider a special case $x_3 = k_2 = 0$, $x_{15} = k_7 =$ `0xaaaaaaaa` and $x_{11} = v_1 = 0$, then we always get the minimum number of difference 4 after one round.

Example 2. With this we observe that the value of $\epsilon_d^* = 0.224494$, which is more than what observed in [1]. Further, we need to discard the PNBs corresponding to the keywords k_2, k_4. This reduces 3 PNBs (two from k_2 and one from k_4) from the list of 36 PNBs described in [1, 7]. Thus we have $n = 33$ PNBs here. Finally we obtain, $\epsilon_a^* = 0.014030$ and $\epsilon^* = 0.003150$. Taking $\alpha = 15$, we get $N = 22.46$. Thus, we get the total complexity as $2^{223+22.46} + 2^{256-15} = 2^{245.52}$.

While we got different kinds of results related to the number of IVs for the selected keys that provide less (for example 10) number of differences after one quarterroundfor ChaCha, the situation is much more exact in case of Salsa. In this case, for a specific pair of keywords, we obtain that out of 2^{32} IVs, there are exactly/approximately $2^{31-(\delta-4)}$ IVs corresponding to the number of differences $\delta = 4$ to 21 after one quarterround. For the cases $\delta = 4$ to 12 we find that this is exact for all the experiments. For $\delta = 13$ to 19, it is approximately equal. For $\delta = 20$ (respectively 21), it deviates slightly from the formula and we obtain the number of IVs producing the number of differences δ after one quarterround is approximately $2^{14.6}$ (respectively 2^{13}) instead of 2^{15} (respectively 2^{14}). We believe that answering this combinatorially may be easier in the case of Salsa (as the quarterround structure is less complicated) than ChaCha. In this direction, we provide some combinatorial insight to this in the next section.

4.1 How to choose an input difference in Salsa

We concentrate further on a quarterround of Salsa. In a columnround, the quarterrounds are as follows: quarterround(x_0, x_4, x_8, x_{12}), quarterround(x_5, x_9, x_{13}, x_1), quarterround$(x_{10}, x_{14}, x_2, x_6)$, and quarterround$(x_{15}, x_3, x_7, x_{11})$. The IVs words x_7, x_8 come in the third position in the respective quarterrounds, whereas the IV word x_9 comes in the second position and the IV word x_6 comes in the last position. If one looks at a quarterround(see (5)), it is easy to note that the third word is involved in the second step of a quarterround, all the other words are involved in the first step itself. Thus, to have less number of differences after one quarterround, it is natural to put a difference in the third word of the quarterround, i.e., c rather than the other words, i.e., a, b, d. In general, to present an attack scenario, the differences are put in the IV words where the attacker can have access. Thus, it is better to consider the IV words that come in the third position of quarterround. This happens for x_7, x_8 but not for x_6, x_9. This is why the $\mathcal{ID}$s in the words x_7, x_8 provide substantially high biases after 4 rounds than the $\mathcal{ID}$s in the words x_6, x_9.

Next we refer why it is logical to choose the most significant bit (i.e., the 31st or the leftmost bit) to put the input difference. We refer to the case corresponding to any one of quarterround(x_0, x_4, x_8, x_{12}), quarterround$(x_{15}, x_3, x_7, x_{11})$. In this case the word where we put the difference comes at c (5). There is no difference in a, b, d that are involved in the first step of quarterround. Thus no difference will be generated. In the second step too, there is no difference in a, b. The difference is only in c, which is XORed. Thus, only one bit difference will be remaining in the MSB of c. In the third step c is added to b. Since the difference is at the most significant bit, even if there is a carry, that will be lost and not be propagated. Thus, one more difference will be produced at the 12-th bit of d. In the last step c, d are added. The differential in the MSB of c will not be propagated, but the one at the 12-th bit of d may be propagated.

Adding the bits 0 to 11 of c and d one may or may not get a carry. If one does not get a carry, then keeping the 12-th bit of c as zero will guarantee that the difference will not propagate further. If one does get a carry, then keeping the 12-th bit of c as 1 will guarantee that the carry will be propagated in both the cases and thus no difference will be be generated further. Thus, there will be two differences in $(d+c)$, one at the 12-th bit and one at the 31-st bit. That will be left rotated by 18 places and thus the differences in a will be at 30-th and 17-th bits. In summary, we will have only four differences in such cases after one quarterround, the 17-th and

30-th bits of a, 31-st bit of c and 12-th bit of d. There will be no difference in the word b. Thus, it is very clear that given a pair of keywords in a specific column, one can get exactly half of the IVs (corresponding to "no carry, 12th bit of c is 0" and "carry, 12th bit of c is 1") for which the number of differences after one quarterround is exactly 4 and that will provide a better result than choosing any IV. If such a condition is not satisfied, there will be more differences, but that will occur only at the last step of quarterround. In such a case more than two differences may appear in a, but naturally the probability of obtaining more differences will decrease.

Estimate 6 *As already discussed, we work with the $\mathcal{ID}$-$\mathcal{OD}$ pair ($\Delta^{(0)}_{7,31}$, $\Delta^{(4)}_{1,14}$). Now we choose random secret keys in each run. Thus, x_3, x_{11} are fixed in each run. Corresponding to that we try out all the properly chosen IVs x_7 such that we obtain 4 differences in the said column after one* quarterround*. As explained above, in each case we get 2^{31} such suitable IVs out of total 2^{32}. We experiment with 256 such runs, randomly choosing the secret keys in each case. With this, we observe that $\epsilon^* = 0.003154, \epsilon^*_a = 0.013778$ and $\epsilon^*_d = 0.228538$ as the median bias. Taking $\alpha = 15$, we obtain $N \approx 2^{22.45}$ and thus, the total complexity becomes $2^{223+22.45} + 2^{256-15} \approx 2^{245.52}$. That is the special case in Example 2 provides a good representation of the general case. Our result provides improved cryptanalysis of Salsa20/8 over the existing results [1, 9, 7].*

5 Conclusion

The most significant cryptanalysis of reduced round Salsa and ChaCha had been described in [1]. After that there are minor tweaks over that technique, but no significant improvement in reducing the overall complexity of the attack could be achieved. In this paper we note that the forward bias can be improved significantly by properly choosing the IVs during the search process over the secret keys. This helps in obtaining notable improvement the time complexity of the attack as summarized in Table 3. The data complexity N is much lower in general, and to have chosen IVs corresponding to differnt keys, it is always upper bounded by 2^{32}, given the 32-bit IV word on which we make the choices.

ChaCha7				
Reference	# PNBs (n)	ϵ^*	α	Total Complexity
[1]	35	0.000590	11	2^{248}
[1] (corrected, our experiments)	35	0.001116	11	$2^{246.71}$
CCD[9]	See Section 4.4 and Table 8 in [9]			$2^{246.5}$
Our (Section 2.1)	45	0.000132	18	$2^{242.82}$
Our (Section 2.1)	41	0.000404	17	$2^{243.56}$
Our (Section 3, specific IVs)	41	0.002012	21	$2^{238.94}$
Salsa20/8				
[1]	36	0.00015	8	2^{251}
[7] (correction of [1])	36	0.00060	12.82	$2^{247.2}$
CCD[9]	See Section 4.4 of [9]			2^{250}
Our (Section 4.1, specific IVs)	33	0.003154	15	$2^{245.52}$

Table 3. Summary of the attack complexity against ChaCha7 (top) and Salsa20/8 (bottom).

We show how one should choose the PNBs so that they are not present in the keywords in the same column with the IV word where the input difference is considered. The basic idea of our choice of IVs are from the fact that one should have minimal number of differences after the application of the quarterround function. This helps in obtaining higher biases in the forward direction. We provide several justifications in this regard to explain the interactions between the key and IV words in the same column. Our results provide best attack on reduced round ChaCha (ChaCha7, with 7 rounds) and Salsa (Salsa20/8, with 8 rounds). However, it should be noted that the ChaCha standardization (ChaCha20) considers 20 rounds and the version of Salsa that is in eStream (Salsa20/12) employs 12 rounds. Thus, our findings, as in the case of all the state-of-the-art results in this area, do not provide any cryptanalytic threat towards the ciphers with full rounds.

Acknowledgments: The author likes to thank Prof. Willi Meier for valuable discussions related to ChaCha and Salsa that help in better understanding of these ciphers.

References

1. J. -P. Aumasson, S. Fischer, S. Khazaei, W. Meier and C. Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. FSE 2008, LNCS 5086, pp. 470-488, Springer. See also `http://eprint.iacr.org/2007/472`
2. D. J. Bernstein. Snuffle 2005: the Salsa20 encryption function. `http://cr.yp.to/snuffle.html`
3. D. J. Bernstein. ChaCha, a variant of Salsa20. `http://cr.yp.to/chacha.html`, January 2008.
4. P. Crowley. Truncated differential cryptanalysis of five rounds of Salsa20. SASC 2006. `http://www.ciphergoth.org/crypto/salsa20/`
5. S. Fischer, W. Meier, C. Berbain, J. -F. Biasse and M. J. B. Robshaw. Non-randomness in eSTREAM candidates Salsa20 and TSC-4. INDOCRYPT 2006. LNCS 4329, pp. 2-16, Springer.
6. T. Ishiguro, S. Kiyomoto and Y. Miyake. Latin Dances Revisited: New Analytic Results of Salsa20 and ChaCha. ICICS 2011, LNCS 7043, pp. 255-266, Springer. See corrections in `http://eprint.iacr.org/2012/065`
7. S. Maitra, G. Paul and W. Meier. Salsa20 Cryptanalysis: New Moves and Revisiting Old Styles. WCC 2015, the Ninth International Workshop on Coding and Cryptography, April 13-17, 2015, Paris, France. See also `http://eprint.iacr.org/2015/217`
8. N. Mouha and B. Preneel. Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive: Report 2013/328, `http://eprint.iacr.org/2013/328`
9. Z. Shi, B. Zhang, D. Feng and W. Wu. Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha. ICISC 2012, LNCS 7839, pp. 337-351, Springer.
10. Y. Tsunoo, T. Saito, H. Kubo, T. Suzaki and Hiroki Nakashima. Differential Cryptanalysis of Salsa20/8. SASC 2007. `http://www.ecrypt.eu.org/stream/papersdir/2007/010.pdf`
11. `http://www.infosecurity-magazine.com/news/google-swaps-out-crypto-ciphers-in-openssl/`
12. The ECRYPT Stream Cipher Project. eSTREAM Portfolio of Stream Ciphers. `http://www.ecrypt.eu.org/stream/`
13. `https://tools.ietf.org/html/draft-irtf-cfrg-chacha20-poly1305-10`