

Cryptanalysis of Reduced Word Variants of Salsa

Sylvain Pelissier

EPFL LACAL, Lausanne, Switzerland

1 Introduction

The Salsa20 stream cipher [1] was proposed by Bernstein as a candidate of the eSTREAM project [2] and was accepted for the final portfolio. The original version of Salsa20 is composed of 20 rounds, works with 32-bit words and uses modular addition, bitwise XOR and rotation operations.

In this paper, we study the best attack on Salsa20 with 256-bit keys due to Aumasson et al. [3] which can break only up to 8 rounds. The attack uses a new concept called probabilistic neutral bits to recover the key faster than exhaustive search in a known keystream differential attack scenario. In other words, the adversary needs to be able to collect the keystream bits for some pairs with a desired input differential. The authors constructed a key recovery attack against Salsa20/8 and Salsa20/7, *i.e.* Salsa20 reduced to 8 and 7 rounds respectively, with respective complexities 2^{251} and 2^{151} .

In the very early publication of Salsa, Bernstein recommended the cryptanalysts [1] to evaluate their attack also on Salsa20 reduced to words of w bits with $w < 32$ and to state the success probability of the attack for every different w . However, to the best of our knowledge, the previous attacks against Salsa were never tested with smaller words length. In addition, since reducing the length of the words means reducing the attack complexity, there is also an opportunity to bypass the outrageous complexity of 32-bit version. Hence we can see if the concept of probabilistic neutral bits works in practice.

That is our motivation to explore how this new method can be applied for word-reduced variants of Salsa20. In particular, we apply this method when Salsa20 is defined on 16-bit and 8-bit words. We can not break the 16-bit version (with a 128-bit key) when it uses 7 rounds. Although, we can attack 6-round and 5-round variants with complexities 2^{117} and 2^{71} respectively, these numbers are still practically hard to reach. For the 8-bit version (with a 64-bit key) we can not break the 6-round variant, however, the 5-rounds version can be broken in time 2^{38} . We focus on the 5-round 8-bit version of Salsa20, implement the attack and extract information which shows that the probabilistic neutral bits method indeed works in practice. However, some cares need to be taken into account if we want to exactly estimate the attack complexity.

2 Specification of the cipher

2.1 Salsa

We use the notation $\text{Salsa}_{(w,R)}$ for the family of stream ciphers which operates on w -bit words and composed of R round. The $8w$ -bit key of this cipher is denoted by the

8-word vector $k = (k_0, k_1, \dots, k_7)$. The $\text{Salsa}_{(w,R)}$ cipher takes as parameters the key and a two-word nonce $v = (v_0, v_1)$. It outputs a sequence of 16-word keystream blocks where the t -th block of this sequence is the output of the *Salsa function* which takes as parameter, a two-word counter (t_0, t_1) corresponding to the integer t in addition to the key and the nonce. This function acts on the 4×4 matrix of words given by :

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

where the c_i 's are known constants.

For a matrix X , the *Salsa function* returns the keystream block Z defined by:

$$Z = X + \text{Round}_R(X)$$

where “+” denotes the matrix modular addition and the Round function is based on a function called *quarterround* function. The *quarterround* function takes as parameter a vector (x_0, x_1, x_2, x_3) and return a vector (y_0, y_1, y_2, y_3) which is defined by:

$$\begin{aligned} y_1 &= x_1 \oplus [(x_3 + x_0) \lll s_0] \\ y_2 &= x_2 \oplus [(x_0 + y_1) \lll s_1] \\ y_3 &= x_3 \oplus [(y_1 + y_2) \lll s_2] \\ y_0 &= x_0 \oplus [(y_2 + y_3) \lll s_3] \end{aligned}$$

where \lll denotes the left rotation on w bits words and the values s_i are some known fixed rotation values. Then Round_R function applies sequentially this *quarterround* function for rounds $i = 1, \dots, R$ on columns (x_0, x_4, x_8, x_{12}) , $(x_5, x_9, x_{13}, x_{17})$, $(x_{10}, x_{14}, x_{18}, x_{22})$ and $(x_{15}, x_{19}, x_{23}, x_{27})$ if i is odd. For even rounds, this function is applied to the rows (x_0, x_1, x_2, x_3) , (x_5, x_6, x_7, x_4) , $(x_{10}, x_{11}, x_8, x_9)$ and $(x_{15}, x_{12}, x_{13}, x_{14})$.

Note that the *Salsa function* is completely specified once we know the constant and rotation values. The version of Salsa submitted to the eSTREAM project, called Salsa20, corresponds to $\text{Salsa}_{(32,20)}$ with the constant values $(c_0, c_1, c_2, c_3) = (61707865, 3320646E, 79622D32, 6B206574)$ and rotation values $(s_0, s_1, s_2, s_3) = (7, 9, 13, 18)$. For $\text{Salsa}_{(16,R)}$ we choose the constant values $(c_0, c_1, c_2, c_3) = (6170, 7865, 3320, 646E)$ and the rotation values $(s_0, s_1, s_2, s_3) = (4, 5, 7, 9)$. We also define $\text{Salsa}_{(8,R)}$ by setting $(c_0, c_1, c_2, c_3) = (61, 70, 78, 65)$ and $(s_0, s_1, s_2, s_3) = (2, 3, 4, 5)$.

The constants have been simply chosen by truncating those of the original Salsa20. The rotation values have been selected to keep almost the same ratio between the rotation value and the word size among the different versions. More precisely we fixed the new value of $s_i(w)$ for w -bit version of Salsa from the value of $s_i(32)$ of the 32-bit version of Salsa by according to the formula $s_i(w) = \left\lceil \frac{s_i(32)}{32} \cdot w \right\rceil$.

3 Experimental results

3.1 Salsa over 32-bit words

We have been able to verify the attack presented in [3]. This version of Salsa is $\text{Salsa}_{(32,7)}$, i.e Salsa with 32-bit word and with 7 rounds. We found the same results

in term of complexity as the ones given in [3] i.e. we have an attack which uses 2^{26} keystream blocks and 2^{151} Salsa encryptions.

3.2 Salsa over 16-bit words

We have made a version of previous attack for $\text{Salsa}_{(16,7)}$. In this case, we did not find a truncated differential after four rounds forward bigger than 10^{-4} . This show that the version of Salsa defined over 16-bit words is more resistant to differential analysis than the version over 32-bit words. We have found some high probability differentials after three rounds of Salsa, we give the two highest of them in Table 1. Refer to [3] for notations.

| ϵ_d | \mathcal{ID} | \mathcal{OD} |
|--------------|---------------------|---------------------|
| 0.977 | $[\Delta_7^0]_{15}$ | $[\Delta_9^3]_{12}$ |
| 0.969 | $[\Delta_7^0]_{14}$ | $[\Delta_9^3]_{11}$ |

Table 1. 3-round differentials for $w = 16$.

| ϵ_d | \mathcal{ID} | \mathcal{OD} |
|--------------|------------------|------------------|
| 0.7607 | $[\Delta_7^0]_7$ | $[\Delta_4^3]_3$ |
| 0.8205 | $[\Delta_7^0]_7$ | $[\Delta_4^3]_4$ |

Table 2. 3-round differentials for $w = 8$

We construct the attack using the same method as previous one against $\text{Salsa}_{(16,6)}$ with the first differential of Table 1, and we obtained a better attack than brute force. It results in an attack with time complexity 2^{117} and a space complexity 2^{16} . We also decided to analysed $\text{Salsa}_{(16,5)}$, i.e. we looked only after two round backward computations. This gives an attack on $\text{Salsa}_{(16,5)}$ with time complexity 2^{71} and space complexity 2^7 .

3.3 Salsa over 8-bit words

We have also implemented the analysis of $\text{Salsa}_{(8,5)}$. As for the 16-bit case, we failed to find a useful truncated differential after four rounds and the number of probabilistic neutral bits were too small after three rounds in backward computations. The Table 2 gives the differentials found after three round foreword. We use the first differential of Table 2 for the attack. This gives an attack on $\text{Salsa}_{(8,5)}$ with time complexity 2^{38} and space complexity 2^8 . The time complexity being low enough makes it tempting to practically simulate the attack.

4 Practical key recovery on $\text{Salsa}_{(8,5)}$

We have implemented a practical attack against $\text{Salsa}_{(8,5)}$. We used the first differential of Table 2. We chose 232 random keys and made the experiment of the first phase of the attack for each key. The first phase of the attack uses an optimal distinguisher to identify 27 subkey bits by making exhaustive search over all possible ones. According to the theory, 2^8 pairs of output keystreams suffice to identify the correct subkey. That is, it is expected that no false positive appears. However, our simulations show that in average 153 subkey candidates show up after the first phase filtering; this is not exactly what is predicted by the theory. This phenomena is due

to what the authors explained in [3] but was not taken into account in the theoretical complexity formula. The Figure 1 shows the distribution of the number of subkey candidates after the first phase.

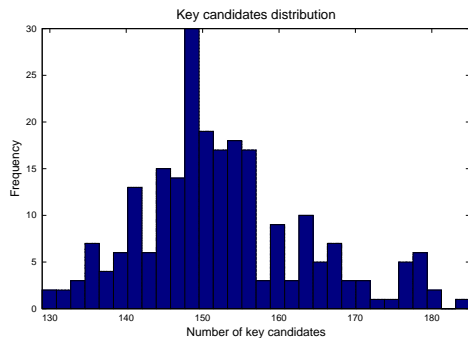


Fig. 1. Subkey candidate distribution

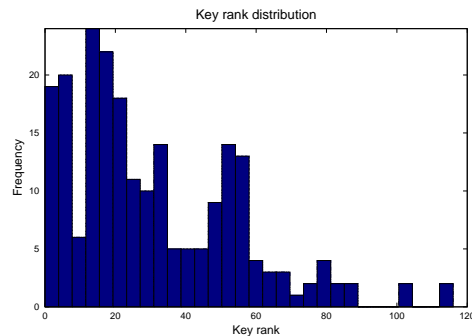


Fig. 2. Right subkey rank

The first phase of the attack last in average 7h15 on our workstation. Then for each subkey candidate which passes the optimal distinguisher filter, the brute-force phase takes about 16 hours which gives an overall attack in about 102 days ($153 \times 16h$) whereas a brute-force attack would take in average 124538 years on the same computer. We did not run the whole second phase on our work station since it takes too long. However, we run it for the right subkey in order to verify that the program outputs the correct key at the end. We also run the brute force phase on some false subkeys in order to measure the time it takes to finish this phase. This practical key recovery attack shows that the concept of probabilistic neutral bits can be applied successfully in practice but the number of false positives are higher than the one predicted by the theory. Therefore the theoretical complexity does not *exactly* describe the running time of the attack and some simulation needs to be performed to refine the complexity estimation.

The distinguisher of [3] is constructed by comparing the correlation measures of the subkeys with a fixed threshold. In other words the ones having a correlation measure greater than the predefined threshold pass the filter. This distinguisher does not need extra memory to store the filtered subkeys. They can be tested for the the second phase right after they pass the filter. One can also consider a distinguisher which works based on the ranking of the subkeys according to their correlation measure which require extra memory to store the candidates which passe the filter but performs better than the earlier. We simulated the rank of the correct subkey, *i.e.* we measured the number of false subkeys which have a bias bigger than the bias of the right subkey. We give the result of this experiment in Figure 2. Our results show that the number of false positive subkeys reduces to 30 in average for the same setup as before. In other words, if after the end of the first phase we sort the subkey candidates according to their bias correlation measure, we need to execute the second phase in average only 30 times, reducing the whole running time to 21 days in average. Despite decreasing the total complexity, it is still higher than what is theoretically expected.

5 Effect on the complexity of the attack on Salsa20/7 and Salsa20/8

We have seen that in the previous attack some non predicted false positive can appear at the end of the first phase and they increase the complexity of the attack. Therefore it is necessary to investigate how the complexity 2^{251} of the attack on Salsa20/8 proposed by Aumasson et al is affected in light of this phenomena. Especially since it is quite competitive to the average complexity of 2^{255} of the brute force attack. Our simulations show that that the time complexity of the attack on Salsa20/7 presented by Aumasson et al. increases by a factor of two, *i.e.* 2^{152} . However, the simulations were too long to be applied for the attack on Salsa20/8 due to huge data complexity of 2^{31} . Nevertheless a less exact analysis of the phenomena can justify that the complexity of this attack should not be affected by this phenomena. Hence we still consider Salsa20/8 as a theoretically broken cipher in time 2^{251} .

6 Conclusion

We gave concrete evidence that the concept of probabilistic neutral bits can be applied successfully in practice and since this concept is quite general we can imagine that it could be applied to the analysis of other ciphers. As explained by Bernstein, it is necessary to validate new attacks on reduced versions to see the non predicted effects of the attack and to see if the predicted complexity agrees with the practical running time. We showed that in this new kind of attack, an unpredicted phenomena could arise. After the first phase of the attack, some false positives which are related to the key happen and they are not predicted by the theory. This phenomena needs to be taken into account to compute the complexity of such attacks and compare the results with other attacks specially when the attack is very competitive with the other known ones. We proposed a new method to limit its impact. In particular, we show that this phenomena makes the best attack on Salsa20/7 two times slower but we expect that the attack on Salsa20/8 remains unchanged.

We also see that Salsa on bigger word size needs more number of rounds to achieve its theoretically expected security level. The minimum number of rounds for 8, 16 and 32-bit versions is respectively 6, 7 and 9 to provide respective security of 64, 128 and 256 bits.

References

1. Bernstein, D.J.: Salsa20. Technical Report 2005/025, eSTREAM, ECRYPT Stream Cipher Project (2005), <http://cr.yp.to/snuffle.html>
2. The eSTREAM Project, <http://www.ecrypt.eu.org/stream>
3. Aumasson, J.P., Fischer, S., Khazaei, Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Goos, G., Hartmanis, J., and Leeuwen, J. (eds.) FSE 2008. . LNCS, vol. 5086, pp. 1148–1158. Springer, Heidelberg (2008)