

Linear approximating for the Cipher Salsa20

Li An-Ping

apli0001@sina.com

Abstract. In this paper we will provide two linear approximating for the stream cipher Salsa20, the both are probable to form distinguishing attack with about 2^{64} blocks of keystream.

1 Introduction

The candidate algorithms for ECRYPT's Stream Cipher Project may be roughly divided as two classes, one class are LFSR-based and another are Addition-Rotation-based. The stream cipher Salsa20 is one of the candidates which belongs to the second class. The detail description of the Salsa20 is referred to see the author's paper [1], the following is a simple describe for the Salsa.

As usually, the symbol \oplus stands for the operation XOR and $R(z, t)$ represents rotate the integer z by t bits.

In the cipher Salsa20, there is defined a function $z = \text{quarterround}(y)$, where

$y = (y_0, y_1, y_2, y_3)$ and $z = (z_0, z_1, z_2, z_3)$ are two arrays of four 32-bits integers.

$$\begin{aligned} z_1 &= y_1 \oplus R(y_0 + y_3, 7), \\ z_2 &= y_2 \oplus R(z_1 + y_0, 9), \\ z_3 &= y_3 \oplus R(z_2 + z_1, 13), \\ z_0 &= y_0 \oplus R(z_3 + z_2, 18). \end{aligned} \tag{1.1}$$

Suppose that ζ is a array of 16 integers, which can be viewed a 4×4 matrix, in the Salsa20 the function $\text{rowround}(\zeta)$ and the function $\text{columnround}(\zeta)$ represent the function $\text{quarterround}(y)$ take on the each row and each column respectively, and the function $\text{doubleround}(\zeta) = \text{rowround}(\text{columnround}(\zeta))$, and the keystream function called $\text{salsa}(x)$ is defined as following.

$$\text{salsa}(x) = x + \text{doubleround}^{10}(x). \tag{1.2}$$

Where x is a array of 64-byte, or 16 32-bits integers, $x = (\sigma_0, k_1, \sigma_1, n, \sigma_2, k_2, \sigma_3)$, $\sigma_i, 0 \leq i < 4$, are four constants of integers, k_1 and k_2 are two 16-byte secret keys, and n consists of a 8-byte nonce and a 8-byte index.

From the simple describe, we found that in the cipher Salsa20 the basic operations are addition, XOR and rotations. Because the sizes of input and the output of the cipher $\text{salsa}(x)$ are large of 64 bytes, so that it is impossible to know the behavior of $\text{salsa}(x)$ in linear approximations by tests. In this paper, we will give a statistic discussion, in the section 2, we will at first give some results about linear approximating for the integer addition, as a application, in the section 3, we will provide two linear approximating for the cipher Salsa20.

2 The linear approximating for the addition of integers

In this section, we will give a discussion about the linear approximating for the integer addition, especially, about the difference between the operation addition and the operation XOR in statistic aspect, the results presented here formerly appeared in our paper [2].

For a binary segment z , denoted by $z[i]$ the i -th bit, and let $s_1(z) = \sum_i z[i]$,

$s_0(z) = h - s_1(z)$, $d(z) = s_0(z) - s_1(z)$, that is, $s_1(z)$ and $s_0(z)$ are the numbers of bit '1'

and bit '0' of the integer z respectively, and $d(z)$ is the bias of them. Suppose that x, y are two

integers of length h bits, denote by $L(x, y) = (x + y) \oplus (x \oplus y)$, we define

$$\begin{aligned} D_i &= (2^{2h} - 2 \sum_{x,y} L(x, y)[i]) / 2^{2h}, \\ D &= \sum_{x,y} d(L(x, y)) / (h \cdot 2^{2h}), \end{aligned} \quad (2.1)$$

Namely, D and D_i are the bias of the occurrence frequencies of bit '0' and '1' in the integers

$L(x, y)$ and in the position i -th bit of $L(x, y)$ respectively as x and y take over all integers of length h . We have following results

Proposition 1

$$D_i = 1/2^i, \quad 0 \leq i < h, \quad (2.2)$$

$$D = \frac{2}{h} \cdot \left(1 - \frac{1}{2^h} \right). \quad (2.3)$$

Proof. It is easy directly to check out $D_0 = 1$, so we assume that $i > 0$. For an integer z ,

denoted by z_i the integer formed by the segment of z from the bit 0 to the bit i . Let

$w = L(x, y)$, denoted by N_i the number of $L(x, y)$ with $w[i] = 1$. It is easy to know that

$w[i] = 0$ if and only if $x_{i-1} + y_{i-1} < 2^i$, then

$$N_i = \left(\sum_{1 \leq k < 2^i} k \right) \times 2^{2(h-i)} = (2^{2h} - 2^{2h-i}) / 2. \quad (2.4)$$

Hence

$$D_i = (2^{2h} - 2 \cdot N_i) / 2^{2h} = 1/2^i.$$

Moreover, it is easy to know that

$$D = \left(\sum_{0 \leq i < h} \Delta[i] \right) / h. \quad (2.5)$$

So,

$$D = \left(\sum_{0 \leq i < h} 1/2^i \right) / h = \frac{2}{h} \cdot \left(1 - \frac{1}{2^h} \right).$$

□

From the Proposition 1 we have seen that, in statistic, $x + y$ is still some like $x \oplus y$, especially, in the first bits, though there are undecided carry operations in the additions. In other words, the probability $(x + y)[i] = (x \oplus y)[i]$ has notable advantage when i is small, e.g. $i = 0, 1, 2, \dots etc.$

Suppose z is an integer variable over the domain Ω , denoted by $\delta(z) = \bigoplus_i z[i]$, we define

$$\Delta_z = \left(|\Omega| - 2 \sum_{z \in \Omega} \delta(z) \right) / |\Omega|. \quad (2.6)$$

Namely, Δ_z is the bias between the frequencies of the values of variable z with even number of bit '1' and with odd number of bit '1'.

Proposition 2

$$\Delta_{L(x,y)} = 1/2^{16}. \quad (2.7)$$

Proof. Dividing a 32-bit integer as two 16-bit integers, then the equation (2.6) can be directly verified through the computer. □

Suppose that $\{w_i\}_1^s$ are a set of Boolean variables in the domain Ω , and let $w = \bigoplus_i w_i$. If

$\{w_i\}_1^s$ are independent of each other, we know that

$$\Delta_w = \prod_i \Delta_{w_i}. \quad (2.8)$$

Although there are no similar results for the generous case that the variables $\{w_i\}_1^s$ are not independent, we give some consideration in statistic.

Let us consider a special case that each w_i is a Boolean function on the domain Ω , that is,

$w_i = w_i(t), t \in \Omega$. If $\{(w_1(t), w_2(t), \dots, w_s(t)) \mid t \in \Omega\}$ are taken as a point set randomly from the domain Ω^s , then for the variable $w^{(1)} = \bigoplus_i w_i(t)$, the bias $\Delta_{w^{(1)}}$ will be about equal to the expectation of $\Delta_{w^{(1)}}$, i.e.

$$\Delta_{w^{(1)}} \approx (\Delta_w)^{1/s}. \quad (2.9)$$

Similarly, if each w_i is a Boolean function on the domain Ω^k , that is, $w_i = w_i(t), t \in \Omega^k$, then the bias of the variable $w^{(k)} = \bigoplus_i w_i(t)$ will be about

$$\Delta_{w^{(k)}} \approx (\Delta_w)^{k/s}. \quad (2.10)$$

3 Two linear approximating for the cipher Salsa20

Linear approximating 1

Let $y = (y_0, y_1, y_2, y_3)$ and $z = (z_0, z_1, z_2, z_3)$ as defined in (1.1), it has that

$$\begin{aligned} z_1 &= y_1 \oplus R(y_0 \oplus y_3, 7) \oplus R(L(y_0, y_3), 7), \\ z_2 &= y_2 \oplus R(z_1 \oplus y_0, 9) \oplus R(L(z_1, y_0), 9), \\ z_3 &= y_3 \oplus R(z_2 \oplus z_1, 13) \oplus R(L(z_2, z_1), 13), \\ z_0 &= y_0 \oplus R(z_3 \oplus z_2, 18) \oplus R(L(z_3, z_2), 18). \end{aligned} \quad (3.1)$$

Denote by

$$\varepsilon = \delta(L(y_0, y_3)) \oplus \delta(L(z_1, y_0)) \oplus \delta(L(z_2, z_1)) \oplus \delta(L(z_3, z_2)). \quad (3.2)$$

Then XOR the four equations in (3.1), it follows that

$$\delta(z_1 \oplus z_2 \oplus z_0) = \delta(y_1 \oplus y_2 \oplus y_0) \oplus \varepsilon. \quad (3.3)$$

Denoted by $\varepsilon_r^{(i)}$, $\varepsilon_c^{(i)}$ and $\varepsilon_d^{(i)}$ the ε 's which are in correspondence to the functions *rowround*(ζ), *columnround*(ζ) and *doubleround*(ζ) in the round i respectively. Let

$z = \text{salsa20}(x)$ and $z' = z_0 \oplus z_1 \oplus z_2$, by the equation (3.3) we have

$$\delta(z') = \bigoplus_{0 \leq i \leq 10} \varepsilon_d^{(i)}, \quad (3.4)$$

where $\varepsilon_d^{(10)}$ correspond to the last round of operation *doubleround*¹⁰(x) + x .

We can also write the equation (3.3) as the following

$$\delta(z') = \bigoplus_{u,v} \delta(L(u, v)), \quad (3.5)$$

where u, v take over some intermediate variables involved the additions.

Let

$$\Delta = \prod_{u,v} \Delta_{L(u,v)}, \quad \bar{\Delta} = \Delta^{1/336}. \quad (3.6)$$

The variables u, v and $L(u, v)$ can be viewed as the functions of the nonce and index when the secret key is fixed. From Proposition 2 and the discussion in the section 2 we have the estimation

$$\Delta_{z'} \approx \bar{\Delta}^2 \approx 1/2^{32}. \quad (3.7)$$

Hence, the linear approximating described above will be form a distinguishing attack in even one nonce of 2^{64} blocks keystream under the assumption that all or most of the intermediate variables u, v are nearly uniform distributed.

Linear approximating 2

For an integer $i, 0 \leq i < 32$, from (3.1) we have that

$$\begin{aligned} z_1[7+i] &= y_1[7+i] \oplus (y_0 \oplus y_3)[i] \oplus L(y_0, y_3)[i], \\ z_2[9+i] &= y_2[9+i] \oplus (z_1 \oplus y_0)[i] \oplus L(z_1, y_0)[i], \\ z_3[13+i] &= y_3[13+i] \oplus (z_2 \oplus z_1)[i] \oplus L(z_2, z_1)[i], \\ z_0[18+i] &= y_0[18+i] \oplus (z_3 \oplus z_2)[i] \oplus L(z_3, z_2)[i]. \end{aligned} \quad (3.8)$$

where the index in brackets will be taken modulo 32.

Summarizing the four equations in (3.8), it follows

$$\begin{aligned} z_1[7+i] \oplus z_2[9+i] \oplus z_3[13+i] \oplus z_0[18+i] \oplus z_3[i] \\ = y_1[7+i] \oplus y_2[9+i] \oplus y_3[13+i] \oplus y_0[18+i] \oplus y_3[i] \oplus \sigma, \end{aligned} \quad (3.9)$$

where

$$\sigma = L(y_0, y_3)[i] \oplus L(z_1, y_0)[i] \oplus L(z_2, z_1)[i] \oplus L(z_3, z_2)[i].$$

For a binary segment α , denoted by $\alpha[i_1, i_2, \dots, i_s] = \bigoplus_{1 \leq k \leq s} \alpha[i_k]$. Let $y = \text{doubleround}^{10}(x)$

and write variable x and y as 4×4 matrices $x = (x_{i,j})_{4 \times 4}$, $y = (y_{i,j})_{4 \times 4}$. Suppose that

$\zeta = (\zeta_{i,j})_{4 \times 4}$ is a 4×4 matrix with entries of integers and let

$$\zeta^{(i)} = \begin{pmatrix} \zeta_{0,0}[29+i] & \zeta_{0,1}[18+i] & \zeta_{0,2}[20+i] & \zeta_{0,3}[i+11, 24+i] \\ \zeta_{1,0}[18+i] & \zeta_{1,1}[7+i] & \zeta_{1,2}[9+i] & \zeta_{1,3}[i, 13+i] \\ \zeta_{2,0}[20+i] & \zeta_{2,1}[9+i] & \zeta_{2,2}[11+i] & \zeta_{2,3}[i+2, 15+i] \\ \zeta_{3,0}[i+11, 24+i] & \zeta_{3,1}[i, 13+i] & \zeta_{3,2}[i+2, 15+i] & \zeta_{3,3}[25+i, 19+i] \end{pmatrix}. \quad (3.10)$$

Then from (3.9) we have

$$\delta(y^{(i)}) = \delta(x^{(i)}) \oplus \left(\bigoplus_{u,v} L(u, v)[i, i+2, i+6, i+11, i+25] \right). \quad (3.11)$$

where u, v take over some intermediate variables involved the additions. Hence

$$\delta(z^{(i)}) = (L(x, z) \oplus \bigoplus_{u,v} L(u, v))[i, i+2, i+6, i+11, i+25]. \quad (3.12)$$

Especially, take $i = 0$, similar to the discussion in linear approximating 1 and by Proposition 1 we have

$$\Delta_{z^{(0)}} \approx 1/2^{2 \times (2+6+11+25)/4} = 1/2^{22}. \quad (3.13)$$

Hence this linear approximating will be able to form a distinguishing attack for Salsa20 with a nonce of 2^{64} keystream enough if the intermediate variables u, v are nearly uniform distributed.

If the secret key is viewed as variable, in other words, if the space checked is assumed in a larger range, then the estimations about $\Delta_{z'}$ and $\Delta_{z^{(0)}}$ will become

$$\Delta_{z'} \approx \bar{\Delta}^6 \approx 1/2^{96}, \quad \Delta_{z^{(0)}} \approx 1/2^{66}, \quad (3.14)$$

and so the correspond distinguishing attacks will require about 2^{192} and 2^{132} blocks of keystream respectively.

Clearly, the distinguishing attacks from the later estimations will have higher reliability than the previous ones.

We have made some tests for the two linear approximating, the following is the test data with 2^{30} blocks of keystream index from 0 to $2^{30} - 1$ and $nonce = 0$.

The Bias in the Linear Approximating 1, 2

Key	$-\log_2(\Delta_{z'})$	$-\log_2(\Delta_{z^{(0)}})$
1	15.13	21.39
2	15.67	15.01
3	15.77	15.85
4	15.67	15.7
5	14.6	14.45
6	15.5	14.47
7	15	15.19
8	15.66	17.88

Table 1

4 Conclusion

The analyses above are statistic, so the accuracy of the estimations rely on that all or most of the

intermediate variables u, v involved in the additions as the functions of the nonce and index and/or are nearly uniform distributed. Though the real cases will not be fully as the treatment above, however this condition should be almost tenable for a good cipher.

References

- [1] Daniel Bernstein, Synchronous Stream Cipher Salsa20, Available at <http://www.ecrypt.eu.org/stream/salsa20.html>
- [2] Li An-Ping, A note on the difference between the operations Addition and XOR, (unpublished).