Truncated differential cryptanalysis of five rounds of Salsa20

Paul Crowley*

17th October 2005

Abstract

We present an attack on Salsa20 reduced to five of its twenty rounds. This attack uses many clusters of truncated differentials and requires 2^{165} work and 2^6 plaintexts.

1 Definition of Salsa20

Salsa20 [1] is a candidate in the eSTREAM project to identify new stream ciphers that might be suitable for widespread adoption. For convenience, we recap here the parameterized family of variants Salsa20-w/r, with w the word size and r the number of rounds; Salsa20 itself is Salsa20-32/20. A **word** is an element of $\mathbb{Z}/2^w\mathbb{Z}$. We omit the precise definitions of word-oriented operations here for brevity; addition ($+$), XOR ($\oplus$) and rotation ($\lll$) are defined in the usual way, and where words are mapped to bytes, a little-endian mapping is used. We define a bijective map S on four-element column vectors of words:

$$S_a((\ y_0 \quad y_1 \quad y_2 \quad y_3\ )^T) = (\ y_1 \oplus ((y_0 + y_3) \lll a) \quad y_2 \quad y_3 \quad y_0\ )^T$$

and compose it four times to build this bijective map on the same:

$$Q = S_{18} \circ S_{13} \circ S_9 \circ S_7$$

(note that the constants given in the subscripts are appropriate for $w = 32$; different constants might be used for a different w) and compose it with a row and column rotate to get this bijective map on matrices:

$$Q'(m) = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & q_1 \\ m_{2,1} & m_{2,2} & m_{2,3} & q_2 \\ m_{3,1} & m_{3,2} & m_{3,3} & q_3 \\ m_{0,1} & m_{0,2} & m_{0,3} & q_0 \end{pmatrix}$$

*paul@ciphergoth.org. Work sponsored by LShift Ltd, www.lshift.net

$$\text{where} \quad q = Q \begin{pmatrix} m_{0,0} \\ m_{1,0} \\ m_{2,0} \\ m_{3,0} \end{pmatrix}, \quad m = \begin{pmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,3} \\ m_{1,0} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,0} & m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,0} & m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix}$$

from which we build this bijective map on four-by-four square matrices of words:

$$R(m) = (Q'^4(m))^T$$

and from this, we define the Salsa20 "hash function":

$$H(m) = m + R^r(m)$$

Salsa20 maps an eight-word key $k_{0...7}$, a two-word nonce $v_{0...1}$ and a two-word stream position $i_{0...1}$ onto a 16-word output matrix as follows:

$$\text{Salsa20}_k(v, i) = H \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

where $c_{0...3}$ are constants dependent on the key length and omitted here for brevity. We also omit the (straightforward) definition of the row-wise deserialization of this output matrix, the resulting counter-mode-like stream cipher, and the Salsa20 variants defined for shorter keys. Salsa20's security goal is that the function above be indistinguishable from a random function to a suitably-bounded attacker; from this its security as a stream cipher may be inferred.

2 Cryptanalysis of $r = 5$

We here attack the Salsa20 PRF directly; the resulting attack on the Salsa20 stream cipher follows straightforwardly. Though many techniques of block cipher cryptanalysis are applicable to Salsa20, it has several features to defeat these techniqes. First, the large block size allows for rapid diffusion without penalty of speed. Second, the attacker can control only four words of the sixteen-word input to the block cipher stage. Nevertheless, we can construct an attack based on multiple truncated differentials which breaks five rounds of the cipher.

Where $r = 5$, the output of the PRF is $m + R^5(m)$. Eight of the sixteen cells in m are known to us; the other eight cells contain the key. We can thus straightforwardly infer eight of the sixteen cells in $R^5(m)$. If we correctly guess k_3, this will give us a complete row in $R^5(m)$, to which we can apply Q^{-1} to infer a complete row of $R^4(m)$.

To go further back, we observe that if every input word but the first to Q^{-1} is known, the final output word may be inferred, and if every input but the second is

known, the first may be inferred. If we can guess the key words $k_{3...7}$, this allows us to infer these entries of $R^5(m)$ given $H(m)$:

$$\begin{pmatrix} \bullet & ? & ? & ? \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

Applying Q^{-1} to each row except the first allows us to infer these entries in $R^4(m)$:

$$\begin{pmatrix} ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \end{pmatrix}$$

from which we can infer these entries in $R^3(m)$:

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ \bullet & ? & ? & \bullet \end{pmatrix}$$

Given a sufficiently powerful distinguisher for the function family $f_k(v,i) = (\text{Salsa20}_k(v,i)_{3,0}, \text{Salsa20}_k(v,i)_{3,3})$ we can therefore test our guesses at $k_{3...7}$.

Consider this example of a low-weight (ie high-probability) truncated differential trail suitable for our purposes, identifed using the techniques of [2]. The limitations on the bits under the attacker's control make it difficult to identify trails that start with useful combinations of bits; each word we control is combined with three we do not before the results are combined with each other. Thus, our input difference is simply a single bit in the high word of the stream position, chosen to minimize the nonlinear avalanche. Before round 1:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0x80000000 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

After round 1 (with probability $\frac{1}{2}$):

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0x00201000 & ? & 0x80000000 & 0x00000100 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

After round 2 (with probability 2^{-9}):

$$\begin{pmatrix} ? & 0x00201000 & 0x40200000 & 0x02000800 \\ ? & ? & ? & ? \\ ? & ? & ? & 0x00000040 \\ 0 & 0x00001000 & 0x00200000 & 0x04000080 \end{pmatrix}$$

And after round 3 (with probability 2^{-12}):

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0x02002802 & ? & ? & ? \end{pmatrix}$$

This trail has sufficiently high probability to act as a suitable distinguisher from which an attack can be built. However, we can do much better. The probability of this difference appearing in the output is much higher than this trail would suggest—in fact, it is closer to 2^{-9}. This is because there are many other low-weight differential trails that result in this difference in $R^3(m)_{3,0}$. Furthermore, there are many high-probability differentials in this word. By experiment, we have even determined a few differential trails whose probability appears to be twice as high as their weight would suggest —this is presumably because of problems with the independence assumption, and suggests that there may be trails which are less probable than their weight would suggest.

By considering many trails, we can build a far more effective attack. Many tradeoffs are possible; we give one example here. We have experimentally determined a set of 1024 possible differences in $R^3(m)_{3,0}$ from this one input difference such that the probability of one of them being right appears to be roughly 30%. With 32 output pairs, the probability that 5 or more of these pairs show a difference in the set is greater than $1 - 2^{-3}$, while the probability of this threshold being met or exceeded by chance is less than 2^{-99}. We try all 2^{160} possible values of $k_{3...7}$; for each that meets the threshold, we try to determine $k_{0...2}$ by simple brute-force search. The true key will be among these values with probability $1 - 2^{-3}$ as noted, and we can expect $2^{160-99} = 2^{61}$ false positives; the cost of the brute-force search stage will thus be roughly $2^{96+61} = 2^{157}$, much less than the cost of determining our candidates for $k_{3...7}$.

3 Conclusions and open questions

It is clear that a naive attack of this type cannot be extended to more than a handful of rounds; this has no negative implications for the security of the full Salsa20-32/20 presented to eSTREAM.

Nonetheless, the degree of clustering exhibited by these differential characteristics is surprising; it is more usual for a single differential trail to dominate. It is also striking to find differential trails whose overall probability is so greatly mispredicted by the products of the probabilities of its components, marking a violation of the independence assumption usual in differential cryptanalysis. In both instances, it would bear investigation whether other ciphers that rely heavily on addition mod 2^n to introduce nonlinearity in $GF(2)$ would also show these properties in differential cryptanalysis, or related properties in other forms of cryptanalysis.

References

[1] Daniel J. Bernstein. Salsa20 specification. http://cr.yp.to/snuffle/spec.pdf, 2005.

[2] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *Fast Software Encryption 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001.

A Anomalous differential trails

We give here examples of differential trails whose observed frequency is markedly different from that predicted by the simplifying assumptions of differential cryptanalysis. The trails below should appear with frequency 2^{-9}, but in 2^{26} trials appeared not the expected 131072 times, but 262018 and 262412 times respectively. Both trails start

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0x80000000 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0x00601000 & ? & 0x80000000 & 0x00000100 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

One goes on thus:

$$\begin{pmatrix} ? & 0x00601000 & 0x40200000 & 0x02000800 \\ ? & ? & ? & ? \\ ? & ? & ? & 0x00000040 \\ 0 & 0x00000100 & ? & ? \end{pmatrix}$$

and the other thus:

$$\begin{pmatrix} ? & 0x00601000 & 0x40200000 & 0x02001800 \\ ? & ? & ? & ? \\ ? & ? & ? & 0x00000040 \\ 0 & 0x00000100 & ? & ? \end{pmatrix}$$