

Distinguishing Attacks on Stream Ciphers Based on Arrays of Pseudo-Random Words

Nathan Keller^{a,1}, Stephen D. Miller^{b,2}

^a*Department of Mathematics, Hebrew University, Givat Ram, Jerusalem, 91904, Israel*

^b*Department of Mathematics, Rutgers University, Piscataway, NJ 08854*

Abstract

In numerous modern stream ciphers, the internal state consists of a large array of pseudo-random words, while the output key-stream is a relatively simple function of the state. It has been heuristically shown in several situations ([3, 8, 9, 10, 11, 14]) that this structure may lead to distinguishing attacks on the cipher. In this note we present a more rigorous treatment of this structural attack. First, we present a rigorous proof of the main probabilistic claim behind it in the basic cases. We then apply it concretely to the cipher SN3 [12], and demonstrate that the heuristic assumptions of the attack are remarkably precise in more complicated cases.

Key words: Stream ciphers, distinguishing attacks, MV3, SN3.

1. Introduction

Stream ciphers are widely used in practical cryptography, especially in environments where the high speed of encryption is crucial. While in general there exist many different types of stream ciphers, there are several structures which are shared by numerous modern ciphers.

One of these structures is basing the internal state of the cipher on a very large array, where the output key-stream is a relatively simple function of the

Email addresses: `nkeller@math.huji.ac.il` (Nathan Keller), `miller@math.rutgers.edu` (Stephen D. Miller)

¹Partially supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

²Partially supported by NSF grant DMS-0601009. The authors wish to thank Orr Dunkelman, Ilya Mironov, and Ramarathnam Venkatesan for their fruitful discussions, Alex Sherman for his assistance conducting the experiments on SN3, and Souradyuti Paul for his discussions on the constant c_0 in footnote 6.

array elements. The first cipher having this structure was alleged RC4 (designed in 1987); modern ciphers of this class include HC-256 [14], the Py family [2], SN3 [12], NGG [7], GGHN [4], MV3 [5], and others (see [11]). The main advantage of ciphers of this class is their high speed: due to the simple update and output rules, the encryption is very fast, while the security follows from pseudo-random accesses to a very large array, unknown to the attacker.

In alleged RC4, the array consists of a permutation of the values $0, 1, \dots, 255$. In the modern ciphers, which are based on longer (usually, 32-bit) words, keeping a permutation of all the possible words in memory is infeasible, and hence the permutation is replaced by an array of pseudo-random words.

Several papers attacking particular stream ciphers (e.g., [9, 10, 14]) demonstrate that in various cases, the *randomness* of the array elements can be used to mount a distinguishing attack on the cipher.³ In [11], Paul and Preneel unified this class of attacks into a single framework. Their method is based on the following phenomenon:

Proposition 1. *Let T be an array of n independently and uniformly distributed random values in $\{0,1\}$, and let i, j be chosen independently and uniformly at random from $\{1, \dots, n\}$. Then*

$$Pr[T[i] = T[j]] = \frac{1}{2} + \frac{1}{2n}.$$

This calculation comes from the fact that if $i = j$, then $T[i] = T[j]$ holds for sure, while if $i \neq j$, then $Pr[T[i] = T[j]] = 1/2$ by randomness. Put another way, the XOR of randomly chosen bits from a fixed, random array is *not* uniformly distributed, but has a slight bias towards zero. This bias potentially applies to any cryptographic operation summing random entries from random arrays. Note that the attack does not work if the values of the array are not random (like in alleged RC4, where at any stage of the encryption, exactly half of the LSBs of the array elements are zero). This is because in the case $i \neq j$, there is a small *counter bias*, which makes the overall probability unbiased. The main

³We note that the practical significance of distinguishing attacks of this class (i.e., attacks that are unlikely to be leveraged to a key-recovery attack), can be questioned (see [13]). This issue is outside of the scope of the current paper.

idea of the attack in [11] (originally presented in [3]) is to find a condition E on the elements of the array, such that if E is satisfied, then the least significant bit (LSB) of a combination of the output words is biased; this bias is used to distinguish the output of the cipher from a random string.

While the framework introduced in [11] is quite general, it is based on heuristic randomness assumptions. In this note we present a more rigorous treatment of this framework. First, we present a rigorous generalization of Proposition 1 to computing the bias of the sum of k random elements, for all $k \geq 1$. The case $k = 4$ is used in [6] to attack a variant of the MV3 stream cipher [5]. Then we demonstrate the precision of the heuristic assumption (i.e., the assumption that if the special event E does not occur, then there is no counter-bias) in more complicated cases by considering a concrete example. We devise a distinguishing attack on the stream cipher SN3 [12] designed by Maltchev in 2002. Our heuristic computation shows that the bias of the distinguisher is $2^{-15.40}$, and a series of experiments with a key-stream of length 2^{40} shows that the bias is in the range between $2^{-15.37}$ and $2^{-15.42}$.⁴ Hence, the heuristic assumption is remarkably precise in this case.

2. Rigorous Proof of the Heuristic Assumption in Basic Cases

In this section we present a rigorous generalization of Proposition 1 to the sum of k random 0/1 elements, for all $k \geq 1$. We give both an exact formula and a simpler asymptotic for large arrays.

Proposition 2. *Let T be an array of N independently distributed uniform random values in $\{0, 1\}$, and let $k \geq 1$. Let S_k denote the sum of k elements of the array which are chosen uniformly and independently at random (allowing possible repetitions). We have that*

$$p_k := \Pr[S_k \equiv 0 \pmod{2}] = \frac{1}{2} + \frac{1}{2} \sum_{r=0}^N \left[2^{-N} \frac{N!}{(N-r)!r!} \right] \left(\frac{2r}{N} - 1 \right)^k ;$$

⁴We note that another distinguishing attack on SN3 was presented in [8]. The attack uses part of the observations used in our attack, along with different techniques. The authors of [8] claim that the attack requires $2^{28.2}$ words of key-stream. However, a careful examination of their attack shows that it requires about 2^{37} words of key-stream, while our attack requires less than 2^{30} words. The attack presented in [9] suffers from similar miscalculations. See [6] for more details.

in particular, $p_k = \frac{1}{2}$ for k odd, $p_2 = \frac{1}{2} + \frac{1}{2N}$, and $p_4 = \frac{1}{2} + \frac{3}{2N^2} - \frac{1}{N^3}$. For N large, p_{2k} is approximately $\frac{1}{2} + \frac{1}{2}(\frac{2}{N})^k \pi^{-1/2} \Gamma(k + 1/2)$, where Γ denotes the Gamma function.

PROOF. Let t_1, \dots, t_N denote the elements of the array, and consider the random variable

$$X := \sum_{i=1}^N (-1)^{t_i} = 2Y - N,$$

where $Y = \sum_{i \leq N} t_i$ is the *Binomial*($N, 1/2$) random variable whose probability distribution is given by the bracketed expression above. Expanding out this sum for k -th powers of X , we see

$$\begin{aligned} X^k &= \sum_{i_1, \dots, i_k=1}^N (-1)^{t_{i_1} + \dots + t_{i_k}} \\ &= 2 \#\{1 \leq i_1, \dots, i_k \leq N \mid t_{i_1} + \dots + t_{i_k} \equiv 0 \pmod{2}\} - N^k. \end{aligned}$$

Combining these formulas shows that p_k is determined by the expected value of moments of Y :

$$p_k = \frac{\#\{1 \leq i_1, \dots, i_k \leq N \mid t_{i_1} + \dots + t_{i_k} \equiv 0\}}{N^k} = E \left[\frac{1}{2} \left(\frac{2Y}{N} - 1 \right)^k \right] + \frac{1}{2}.$$

This establishes the formula asserted above, because this expected value is its second term. The special cases of p_k mentioned can be seen by direct calculation.

As far as the limit of p_{2k} for N large, we recall the Central Limit Theorem: that the *Binomial*($N, 1/2$) random variable is approximated by the Normal distribution with mean $N/2$ and variance $N/4$. This allows us to approximate the sum above with the integral

$$\int_{\mathbb{R}} \left[\sqrt{\frac{2}{\pi N}} e^{-\frac{(2x-N)^2}{2N}} \right] \left(\frac{2x}{N} - 1 \right)^{2k} dx,$$

which — after changing variables to $y = \sqrt{2x - N}$ — may be computed as $(\frac{2}{N})^k \pi^{-1/2} \Gamma(k + 1/2)$ via Euler's integral formula $\Gamma(s) = \int_0^\infty e^{-u} u^{s-1} du$. \square

3. Practical Verification of the Heuristic Assumptions in a Special Case: the SN3 Stream Cipher

In this section we demonstrate the precision of the heuristic assumptions in cases where the assumptions cannot be proved rigorously. We do it by devising

a practical distinguishing attack on the stream cipher SN3 [12]. We are able to compute its bias using the heuristic assumption, and verify it experimentally to striking precision. Of course, it does not prove that the assumption is valid for any stream cipher, but it does demonstrate that the assumption is reasonable.

3.1. Description of SN3

The stream cipher SN3 [12] was designed by Simeon Maltchev in 2002. SN3 is a software-efficient stream cipher, optimized for execution on 32-bit microprocessors. It is based on a large array of pseudo-random words, and uses only simple word operations, like XOR and cyclical rotations.

The structure of SN3 is the following: The internal state of the cipher consists of three arrays $V1$, $V2$, and $V3$, of sixty four 32-bit words each, and three 6-bit indices, i , j , and m . Index i addresses only the $V1$ array, index j addresses only the $V2$ array, and index m addresses only the $V3$ array.

The key-stream generation is composed of 64-step cycles, where in each cycle, i takes the values from 0 to 63 sequentially, and j and m perform a (pseudo)-random walk, determined by the elements of the $V1$ array. In each step, one word from each array is selected according to i , j , and m , the XOR of these three words is output as the key-stream word, and the three words are used to update themselves by a relatively simple update rule.

The structure of a step is outlined in the pseudo-code below. In the code, \oplus denotes bitwise XOR, \lll denotes cyclical left rotation, and \ggg denotes noncyclical right shift (which discards the rightmost bits). The key-stream word which is outputted at the end of the step is denoted K_i . The indices i and j are set to zero at the beginning of the key-stream generation process.

- | | | | |
|----|---------------------------------|-----|--|
| 1. | $T1 = V1[i]$ | 6. | $V2[j] = (T2 \lll 5) \oplus T3 \oplus 0x8c591ca1$ |
| 2. | $T2 = V2[j]$ | 7. | $V3[m] = (T3 \lll 17) \oplus T1 \oplus 0xab8ec254$ |
| 3. | $m = T1 \pmod{64}$ | 8. | $i = i + 1 \pmod{64}$ |
| 4. | $T3 = V3[m]$ | 9. | $j = T1 \ggg 8 \pmod{64}$ |
| 5. | $V1[i] = (T1 \lll 1) \oplus T2$ | 10. | $K_i = T1 \oplus T2 \oplus T3$ |

After each 64-step cycle, the arrays are rotated cyclically to the left: the contents of $V1$ are placed in $V3$; those of $V3$, in $V2$; and those of $V2$, in $V1$. We omit

the key expansion algorithm, and instead, we assume that the initial values in the arrays $V1$, $V2$, and $V3$ are independently uniformly distributed.

3.2. Distinguishing Attack Based on the Heuristic Assumption

The basic observation we use in the attack is a weakness in the update rule of $V1$, $V2$, and $V3$. We observe that regardless of the values of the words $T1$, $T2$, and $T3$ before the update operation, the values $V1[i]$, $V2[j]$, and $V3[m]$ after the update satisfy a simple linear relation involving the Hamming weight (the number of 1's in a word's binary representation). If these words are next updated simultaneously, this will result in a bias that is described below.

Proposition 3. *Consider the values $V1[i]$, $V2[j]$, and $V3[m]$ right after step 7 above. The parity of the Hamming weight of $V1[i] \oplus V2[j] \oplus V3[m]$ is zero, that is, its binary representation has an even number of 1's.*

PROOF. Let $g(x)$ denote the parity of the Hamming weight of a 32-bit word x , i.e., $g(x) \equiv HW(x) \pmod{2}$. It is easy to see that for any two words x and y , we have $g(x \oplus y) = g(x) \oplus g(y)$, $g(x) \oplus g(x) = 0$, and $g(x \lll k) = g(x)$ for any $0 \leq k \leq 31$. Using these rules and the formulas in steps 5-7 above, one verifies

$$\begin{aligned} g(V1[i] \oplus V2[j] \oplus V3[m]) &= g(V1[i]) \oplus g(V2[j]) \oplus g(V3[m]) = \\ &= g(T1) \oplus g(T2) \oplus g(T2) \oplus g(T3) \oplus g(0x8c591ca1) \oplus g(T3) \oplus g(T1) \oplus g(0xab8ec254) \\ &= g(0x8c591ca1) \oplus g(0xab8ec254) = 0. \quad \square \end{aligned}$$

In order to exploit Proposition 3, we first give a criterion that describes such a simultaneous update of the three words above in terms of the indices (i, j, m) in the current step, and the indices (i', j', m') in a certain step in the previous cycle of 64 steps. We denote the k -th step in the r -th cycle by S_k^r , and examine the triples of indices (i, j, m) corresponding to each step.

Definition 1. *We say Event E_k^r occurs if the following relations among indices are met during step S_k^r . Let $(T1, T2, T3) = (V1[a_1], V2[a_2], V3[a_3])$ be the words producing the output of this step (i.e., for this step, $(i, j, m) = (a_1, a_2, a_3)$); in particular, $a_1 = k$). The event occurs when these three conditions are satisfied:*

1. In step $S_{a_3}^{r-1}$, the indices (i, j, m) are equal to (a_3, a_1, a_2) .

2. In the steps S_i^{r-1} for all $i > a_3$, the indices satisfy $j \neq a_1$ and $m \neq a_2$.
3. In the steps S_i^r for all $i < a_1$, the indices satisfy $j \neq a_2$ and $m \neq a_3$.

Note that if Event E_k^r occurs, then the values $T1$, $T2$, and $T3$ used for producing the key-stream word in step S_k^r are exactly of the form described in Proposition 3. Indeed, Condition 1 assures that $T1$, $T2$, and $T3$ were updated together in step $S_{a_3}^{r-1}$, and Conditions 2 and 3 assure that none of them was updated since the simultaneous update. The next proposition calculates the probability of the events E_k^r , based on our randomness assumption mentioned at the end of Section 1. The proofs of the following two Propositions can be found in [6].

Proposition 4. *The probability of the event E_k^r is $Pr[E_k^r] = 2^{-13.20}(1-1/64)^{2k}$.*⁵

Using Proposition 4 we can show that the parity of the Hamming weight of the key-stream words in SN3 is biased:

Proposition 5. *Consider a random key-stream word x output by the SN3 algorithm (at any stage of the key-stream generation), and its Hamming weight parity $g(x)$. Then $Pr[g(x) = 0] = \frac{1}{2} + 2^{-15.40}$.*

Because the Hamming weight parity of a random stream is unbiased, we obtain:

Corollary 1. *The key-stream of SN3 can be distinguished from a random stream using less than 2^{30} key-stream words.*⁶

3.3. Experimental Results

We modified Maltchev's C code [12] in order to verify the bias predicted by Proposition 5. It took approximately 30 hours to generate each of 3 separate samples of 2^{40} output words on a Dell PowerEdge 2650 computer equipped with an Intel Xeon 2.6GHZ processor and 2GB RAM. The empirical bias from our $3 \cdot 2^{40}$ trials was $0.0000231 \approx 2^{-15.402}$, almost identical to our prediction.

⁵Throughout this paper numbers are rounded to the precision shown.

⁶We calculate the amount of key-stream required for a distinguishing attack using Theorem 1 of [11], which states that a key-stream length at least $c_0 q^{-2}$ is sufficient for getting an advantage of .5 over a random stream. Here q denotes the bias of the linear approximation, and $c_0 \approx .454936$ is the positive constant such that $\frac{1}{\sqrt{2\pi}} \int_{-\sqrt{c_0}}^{\sqrt{c_0}} \exp(-u^2/2) du = \frac{1}{2}$. (Please note that the value of c_0 was slightly misstated as $\approx .4624$ in [11].) For a detailed discussion, see [1].

References

- [1] C. Baigneres, P. Junod, and S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, Advances in Cryptology - proceedings of Asiacrypt 2004, Lecture Notes in Computer Science **3329**, pp. 432–450, Springer-Verlag, 2004.
- [2] Eli Biham and Jennifer Seberry, *Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays*, eStream, ECRYPT Stream Cipher Project, Report 2005/023, 2005.
- [3] S. Fluhrer and D. McGrew, *Statistical Analysis of the Alleged RC4 Keystream Generator*, proceedings of FSE 2000, Lecture Notes in Computer Science **1978**, pp. 19–30, Springer-Verlag, 2001.
- [4] C. Gong, K.C. Gupta, M. Hell, and Y. Nawaz, *Towards a General RC4-like Keystream Generator*, proceedings of CISC 2005, Lecture Notes in Computer Science **3822**, pp. 162–174, Springer-Verlag, 2005.
- [5] N. Keller, S. D. Miller, I. Mironov, and R. Venkatesan, *MV3: A New Word Based Stream Cipher Using Rapid Mixing and Revolving Buffers*, Topics in Cryptology - proceedings of CT-RSA 2007, Lecture Notes in Computer Science **4377**, pp. 1–19, Springer-Verlag, 2006.
- [6] N. Keller and S. D. Miller, *Distinguishing Attacks on Stream Ciphers Based on Arrays of Pseudo-Random Words*, full version.
- [7] Y. Nawaz, K.C. Gupta, and C. Gong, *A 32-bit RC4-like Keystream Generator*, Cryptology ePrint Archive, 2005/175.
- [8] M. A. Orumiehchi and S. F. Mohebbipoor, *Distinguishing Attack on SN3 Stream Cipher*, proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1392–1395, 2008.
- [9] M. A. Orumiehchi, S. F. Mohebbipoor, and H. Ghodosi, *Cryptanalysis of MV3 Stream Cipher*, proceedings of CANS 2008, Lecture Notes in Computer Science **5339**, pp. 240–251, Springer-Verlag, 2008.

- [10] S. Paul, B. Preneel, and G. Sekar, *Distinguishing Attacks on the Stream Cipher Py*, proceedings of Fast Software Encryption 2006, Lecture Notes in Computer Science **4047**, pp. 405–421, Springer-Verlag, 2006.
- [11] Souradyuti Paul and Bart Preneel, *On the (In)security of Stream Ciphers Based on Arrays and Modular Addition*, Advances in Cryptology - proceedings of ASIACRYPT 2006, Lecture Notes in Computer Science **4284**, pp. 69–83, Springer-Verlag, 2006.
- [12] Simeon V. Maltchev, *The SN3 Stream Cipher*, Available on-line at <http://www.geocities.com/smaltchev/sn3.html>.
- [13] G. Rose and P. Hawkes, *On the Applicability of Distinguishing Attacks Against Stream Ciphers*, preproceedings of the 3rd NESSIE workshop, available online at <http://eprint.iacr.org/2002/142.pdf>.
- [14] Hongjun Wu, *A New Stream Cipher HC-256*, proceedings of Fast Software Encryption 2004, Lecture Notes in Computer Science **3017**, pp. 226–244, Springer-Verlag, 2004.