Statistical weakness in Spritz against VMPC-R: in search for the RC4 replacement

Bartosz Zoltak

www.vmpcfunction.com
bzoltak@vmpcfunction.com

Abstract. We found a statistical weakness in the Spritz algorithm designed by Ronald L. Rivest and Jacob C. N. Schuldt. For $N = 8$: $Prob(output(x) = output(x + 2)) = 1/N + 0.000498$. The bias becomes statistically significant (for $N = 8$) after observing about $2^{21.9}$ outputs. Analogous bias occurs for $N = 16$. We propose an algorithm (VMPC-R) which for $N = 8$ produced $2^{46.8}$ (31 million times more) outputs which remained undistinguishable from random in the same battery of tests. Supported by a series of additional statistical tests and security analyses we present VMPC-R as an algorithm we hope can be considered a worthwhile replacement for RC4.

Keywords: PRNG; CSPRNG; Spritz; RC4; VMPC-R; stream cipher; distinguishing attack

1 Introduction

RC4 is a popular and one of the simplest symmetric encryption algorithms. Over the years numerous attacks against it have been published. In 2014 Ronald L. Rivest, the author of RC4, along with Jacob C. N. Schuldt published their new proposition of an RC4-like algorithm - Spritz [1]. In this paper we report a statistical weakness in Spritz and propose another algorithm - VMPC-R [3], which appears to produce output of much higher statistical quality.

The authors of Spritz carried out a substantial amount of statistical tests on their algorithm and also on a large number of other candidate designs which they rejected an route to their best choice that became Spritz. This is the exact research approach which we applied in 2010-2013 when we were designing VMPC-R; see [3] for more details.

In this paper we use the distant-equalities statistical test (which we used in the VMPC-R research process and published in 2013 in [3]) to show a statistical weakness in the output generated by Spritz: probability that $output(x) = output(x + 2)$ is biased.

We found the distant-equalities test to be extremely sensitive and extremely hard to pass. Our finalist - VMPC-R - which was the first one among about 250 candidates to pass the test - passed all the further statistical tests we subjected it to. In 2014 we published another paper [4] in which we showed weaknesses in 20 other RC4-like algorithms - all using only the distant-equalities test. Spritz is a significant improvement over RC4 but it still fails this test.

In 2010-2013 we carried out an experiment to find our proposition for the RC4 replacement. VMPC-R is the result of that project. In the approach we took when designing VMPC-R we rejected all candidates which revealed any bias for any word size even in relatively huge (like $2^{46.8}$ for $N = 8$) sample sizes. We accepted only a candidate which performed flawlessly even for very small word sizes with $N \in \{2, 3, 4, 5, 6, 7, 8, 16, ...\}$). On the contrary - according to [1], Spritz was accepted despite showing biases for small word sizes; the biases are not specified, only reported to require 2^{81} outputs for $N = 256$.

VMPC-R was proposed by Bartosz Zoltak in 2013 in [3]. In contrast to a 3-bit ($N = 8$) RC4, which failed the distant-equalities test with only about 100,000 outputs, and to 3-bit Spritz which failed at around 4 million ($2^{21.9}$) outputs, the 3-bit variant of VMPC-R passed the test with over 122 trillion ($2^{46.8}$) outputs which still remained undistinguishable from random. We terminated the test only because of the limitations of computational power. We speculate (judging on the performance for $N < 8$) that VMPC-R would remain bias-free in the $N = 8$ test even after exhausting the complete internal state space. This would mean increasing the sample size 28-fold to $2^{51.6}$.

To the best of our knowledge - VMPC-R produces (by far) the highest quality output (in terms of statistical properties) among all the RC4-like ciphers we know of. It also remains the simplest RC4-like algorithm (that we were able to come up with) which is able to pass the distant-equalities test with samples not smaller than those we analysed. VMPC-R performed well in a number of further statistical tests and it did not show any other security problems - as far as we were able to analyse it. We believe that VMPC-R has the attributes to be considered a good replacement for RC4.

2 The distant-equalities statistical test

The test measures the numbers of occurrences of 8 different Events in the algorithm's output. Informally - Event k occurs when two words of the keystream are equal and the distance between them is k. We ran the test for $k \in \{1, 2, ..., 8\}$ More specifically: Let z_i denote the i-th output word of the algorithm. Then:

$$\text{Event } k\text{: } z_i = z_{i+k} \text{ for } k \in \{1, 2, ..., 8\}$$

In a random keystream each Event would occur with probability $p = 1/N$, where N is the algorithm's word size (the algorithm operates on $\mathbb{Z}_N$). The expected number of occurrences of each Event in n samples ($E = np$) and the standard deviation ($\sigma = \sqrt{np(1-p)}$) are given according to the binomial distribution. The test counts the numbers of occurrences of Events 1,2,...,8 and compares them with the model $E = np$ using σ as the measure of deviation (the test calculates $d = (f - E)/\sigma$, where f is the measured number), which is a common statistical practice. For large samples the binomial distribution can be approximated with the normal distribution, where e.g. the probability of exceeding the expected value by 3σ is $2^{-8.5}$, by 5σ: $2^{-20.7}$, by 7σ: $2^{-38.5}$.

3 RC4, Spritz and VMPC-R in the distant-equalities test

All the three algorithms were tested using a key scheduling algorithm which ensures uniform distribution of the internal state. We analysed scaled down variants of the algorithms (for $N < 256$) which magnified any non-random behavior the algorithms carried in their design. The approach to analyse RC4-like ciphers for smaller word sizes was applied e.g. in [16], [17], [18] and also in the Spritz paper [1].

3.1 Specification of the algorithms

Variable w used by Spritz can take on any value that is relatively prime with N. We tested the algorithm for all possible values of w ($\{1,3,5,7\}$ for $N = 8$ and $\{1,3,5,7,9,11,13,15\}$ for $N = 16$) and found no effect of the choice of w for the performance in the test (and also for the cycle lengths discussed in Section 5.4).

Table 1. Variables used by RC4 and Spritz

N : word size (the algorithm operates on $\mathbb{Z}_N$)
S : N-element permutation of integers $\{0, 1, ..., N-1\}$
i, j, k, z, w : integer variables
+ denotes addition modulo N

Table 2. RC4

repeat steps 1-4:
1. $i = i + 1$
2. $j = j + S[i]$
3. swap $S[i]$ with $S[j]$
4. output $S[S[i] + S[j]]$

Table 3. Spritz

repeat steps 1-6:
1. $i = i + w$
2. $j = k + S[j + S[i]]$
3. $k = i + k + S[j]$
4. swap $S[i]$ with $S[j]$
5. $z = S[j + S[i + S[z + k]]]$
6. output z

Table 4. VMPC-R

N : word size (the algorithm operates on $\mathbb{Z}_N$);
P, S : N-element permutations of integers $\{0, 1, ..., N-1\}$
a, b, c, d, e, f, n : integer variables
+ denotes addition modulo N
repeat steps 1-10:
1. $a = P[a + c + S[n]]$
2. $b = P[b + a]$
3. $c = P[c + b]$
4. $d = S[d + f + P[n]]$
5. $e = S[e + d]$
6. $f = S[f + e]$
7. output $S[S[S[c + d]] + 1]$
8. swap $P[n]$ with $P[f]$
9. swap $S[n]$ with $S[a]$
10. $n = n + 1$

3.2 Interpretation of the results

Table 5. RC4, Spritz and VMPC-R distant-equalities test results for N=8

Algorithm	Samples	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Event 8
RC4	2^{20}	4.74	−9.13	3.55	−2.75	4.66	0.33	0.61	−0.77
Spritz	2^{26}	0.74	13.82	−2.36	−1.44	−0.06	0.55	−0.09	−0.02
VMPC-R	$2^{46.8}$	−1.46	−0.27	−1.25	1.20	1.39	−0.65	0.56	−0.88

Table 6. RC4, Spritz and VMPC-R distant-equalities test results for N=16

Algorithm	Samples	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Event 8
RC4	2^{30}	27.82	−47.12	4.00	10.52	21.18	6.00	8.31	2.04
Spritz	2^{40}	−1.24	7.12	0.68	−0.24	−0.88	0.20	−0.08	1.08
VMPC-R	$2^{46.5}$	0.00	−1.99	0.82	0.46	−1.20	−0.79	−1.75	−1.06

Tables 5 and 6 present by how many standard deviations (σ) the measured numbers of Events 1,2,...,8 were different from their expected values, i.e. the tables report $d = (f - E)/\sigma$. We are hoping to get the absolute values of the deviations under 3 or less strictly - under 4.

3.3 RC4 results

The sample size of the $N = 8$ test (2^{20}) was limited by the cycle of length 955,496 that RC4 falls into. This refrained the deviations to develop into greater sizes. The test for $N = 16$, which did not meet the end of a cycle, revealed how severely biased the RC4 output is in the test.

In the $N = 8$ test RC4 needed only about 100,000 outputs to exceed -3σ in Event 2 ($z_i = z_{i+2}$). In the full sample of 2^{20} outputs the Event 2 deviation grew to over -9σ. Deviations in Event 1 ($z_i = z_{i+1}$) of 4.74σ, Event 3 ($z_i = z_{i+3}$) of 3.55σ and Event 5 ($z_i = z_{i+5}$) of 4.66σ were also beyond acceptance.

Greater sample size of 2^{30} for $N = 16$ revealed extreme deviations in all but the last Event with peaks of 27.82σ in Event 1 and -47.12σ in Event 2.

While there is no guarantee that the same deviations would hold for other word sizes it is apparent that the generic construction of RC4 shows significant flaws in the test. We would rather expect proper pseudo-random behavior regardless of the selected word size.

3.4 Spritz and VMPC-R results

Spritz performs well in all but Event 2. This means that the probability that two Spritz outputs separated by one discarded output are equal is biased:

$$Prob(output(x) = output(x + 2)) > 1/N$$

The result occurred for $N = 8$ and $N = 16$. We did not test it for other word sizes. However having tested hundreds of RC4-like algorithms with the test and having regularly observed repetitions of the biases for different word sizes we expect that in the full $N = 256$ version of Spritz the bias would also stand.

For $N = 8$ we observed 10,559,801 occurrences of Event 2 instead of the expected 10,517,885 in 84,143,080 samples (the longest Spritz cycle for $N = 8$) which means a measured probability of 0.125498 instead of 0.125. While for $N = 8$ we have $\sigma = 3033$ we get $d =$ (10,559,801 - 10,517,885) / 3033 = 13.82 meaning that Spritz exceeded the expected number by 13.82σ rather than the tolerable 3σ or less strictly 4σ. This translates to 3,969,190 or $2^{21.9}$ samples which are required to reveal the bias (which we identify by exceeding 3σ) for $N = 8$.

For $N = 16$: 63,500,272,687 occurrences of Event 2 rather than the expected 63,498,535,500 in 1,015,976,568,000 samples were observed which means a measured probability of 0.06250171 instead of 0.0625. $\sigma = 243987$ and $d =$ (63,500,272,687 - 63,498,535,500) / 243987 = 7.12. So Spritz exceeded the expected number by 7.12σ. This means that 180,343,952,000 or $2^{37.4}$ samples are required to reveal the bias (by exceeding 3σ) for $N = 16$.

VMPC-R performed levels-of-magnitude better in the test. For $N = 8$ it produced 122 trillion $2^{46.8}$ outputs without revealing any bias. This is $2^{24.9}$ or 31 million times more unbiased outputs of VMPC-R than the amount ($2^{21.9}$) which is enough to reveal the bias in Spritz. For $N = 16$ we terminated the test after $2^{46.5}$ samples due to limitations in computational power without finding any bias.

The authors of Spritz state in [1] and we agree that a proper algorithm should produce quality output regardless of the selected word size. We would like to point out that despite making that claim the authors of Spritz accepted their final candidate having found a bias for small word sizes. This is contradictive. The bias is reported to be minor (extrapolated to requiring 2^{81} outputs to distinguish the stream from random for $N = 256$). The bias however remains. It is not said which statistical property of the generated keystream reveals the bias. This may be the same bias we discuss in this paper or it may be another one.

4 VMPC one-way function: used in VMPC-R and used in Spritz

VMPC one-way function (Variably Modified Permutation Composition) was proposed by Bartosz Zoltak at FSE 2004 [2]. It transforms an N-element permutation S into another N-element permutation V:

$$V[x] = VMPC[S[x]] = S[M_1[S[M_2[S[x]]]]]$$

M_1, M_2 are any (non-secret) permutations such that $M_1[x] \neq M_2[x]$ for $x \in \{0, 1, ..., N-1\}$. An example VMPC function is $V[x] = P[P[P[x]] + 1]$, where + denotes addition modulo N. The VMPC function shows to be one-way. Applying it as a final transformation of the permutation (S) before releasing the output provides an additional layer of security against state-recovery attacks. The function was applied in VMPC-R in 2013 [3] in step 7: $S[S[S[c + d]] + 1]$ and also in the original VMPC stream cipher published at FSE 2004 [2].

Step 5 of the Spritz algorithm:

$$z = S[j + S[i + S[z + k]]]$$

is the VMPC function with parameters: $x = z + k$; $M_2[x] = x + i$; $M_1[x] = x + j$. Clearly the parameters z, k, j, i can change in each iteration. Also on average in one in N cases, when it happens that $i = j$, the Spritz transformation does not fit the definition of the VMPC function. This however does not change the fact that step 5 of the Spritz algorithm conceptually is the VMPC one-way function. We state this as a good thing because the function, appearing to be one-way, naturally fits the design of RC4-like ciphers and is an easy to use resource to strengthen the algorithm at a low cost. The function (i.e. in the $V[x] = S[S[S[x]] + 1]$ form) can be implemented with three elementary 1-cycle *mov* processor instructions per one output.

The M_i permutations corrupt the cycle structure of S in such a way that deriving any information about S from V requires guessing a number of elements of S (e.g. 34 guessed elements is the estimate for 256-element permutations). Any occurrence of $M_1[x] = M_2[x]$ (against the definition) would weaken the function's one-way properties. A function $V[x] = S[M_1[S[M_1[S[x]]]]]$ would be trivial to invert for any value of M_1. The original estimate appears to still stand that inverting the function for 256-element permutations requires an average effort of 2^{260}.

In strict terms one-way functions have not been proved to exist. The existence of one would imply $P \neq NP$. The VMPC function with no polynomial-time algorithm for inverting known can be regarded as believed-to-be-one-way.

5 VMPC-R

The name VMPC-R consists of two parts: VMPC stands for the name of the one-way function (summarized in Section 4) employed in step 7, and R stands for random. Specification of the algorithm can be found in Table 4.

5.1 The research process leading to VMPC-R

Our objective was to find an algorithm as simple as possible but to pass the distant-equalities test with considerable sample sizes even for small word sizes. Before reaching the final version we examined over 250 RC4-like candidates which all failed. We found only one algorithm which was able to pass the test. The variations we tested came from combining the following factors:

1. The number of permutations (1 or 2) and variables (2-7).
2. The way the variables were updated by the permutations.
3. The way the variables feedbacked each other.
4. The way the output word was computed from the variables and the permutations.
5. The way the variables controlled the swap operations on the permutations.

The purpose of the first stage was to find one candidate to carry out the extended testing on. Each version first underwent the test for $N = 8$. If the candidate failed, it was rejected.

Initially we tested approximately 50 designs which used a single permutation and different numbers of variables. Some versions used two swap operations controlled by different combinations of variables. We observed huge gaps separating each of the candidates from the random model and we were finally unable to build an algorithm even close to passing the test using a single permutation.

We concluded that an RC4-like cipher must use two permutations to pass the distant-equalities test. For the sake of simplicity we limited the search space to one swap operation per each permutation per one output. Adding the second permutation significantly raised the complexity of the algorithm and broadened the number of design variations.

Adding the second permutation did not make the search an instant success. Instead it significantly slowed it down due to the large number of candidates and the fact that they generally performed much better in the test and it took more time to find the deviations. The majority of our two-permutation candidates used 5 variables. From over 160 different combinations of them - none could pass the test with much over 1 billion outputs for $N = 8$.

As the final step we chose the combinations of factors 2 and 3 which appeared to work best and analysed a 7-variable scheme in 36 configurations of different variables taking part in the output computing step (factor 4) and in the two swap operations (factor 5). Out of them 20 quickly failed. Out of the remaining 16 we found a few which showed promising results. From them we chose the one which performed best. With the majority of other candidates struggling to pass the 1 billion outputs mark (for $N = 8$), the final version passed the test with over 122 trillion ($2^{46.8}$) undistinguishable from random outputs. This is how VMPC-R was born.

5.2 The extended battery of tests on VMPC-R

The final candidate underwent a series of additional statistical tests. Apart from the distant-equalities test, we measured several other parameters which were commonly exploited in distinguishing attacks against similar ciphers. Let N denote the algorithm's word size, z_i the i-th output word; n is the internal variable of VMPC-R. The battery included the following measurements:

- 8 Events in the distant-equalities test: $z_i = z_{i+k}$ for $k \in \{1, 2, ..., 8\}$
- N numbers of occurrences of each possible output value z_i
- N^2 numbers of occurrences of each possible pair $[n, z_i]$
- N^2 numbers of occurrences of each possible pair $[z_i, z_{i+1}]$
- N^3 numbers of occurrences of each possible combination $[n, z_i, z_{i+1}]$

5.3 VMPC-R results

To probe the algorithm's output we used over 100 computers and generated a total of over 2 quadrillion ($10^{15.3} = 2^{51}$) outputs. Table 7 presents the sample sizes we tested for different word sizes. Table 8 shows the results of the distant-equalities test. We do not quote the results of the remaining tests from the battery due to their large capacity. We found no bias in any of the tests.

Table 7. VMPC-R distant-equalities test sample sizes

Word size	Number of output words tested
N=2 (1-bit)	$504 = 10^{2.7} = 2^{9.0}$
N=3	$48\ 687 = 10^{4.7} = 2^{15.6}$
N=4 (2-bit)	$7.8 \text{ million} = 10^{6.9} = 2^{22.9}$
N=5	$365.8 \text{ million} = 10^{8.6} = 2^{28.4}$
N=6	$113.8 \text{ billion} = 10^{11.1} = 2^{36.7}$
N=7	$1.9 \text{ trillion} = 10^{12.3} = 2^{40.8}$
N=8 (3-bit)	$122 \text{ trillion} = 10^{14.1} = 2^{46.8}$
N=16 (4-bit)	$103 \text{ trillion} = 10^{14} = 2^{46.5}$
N=32 (5-bit)	$114 \text{ trillion} = 10^{14.1} = 2^{46.7}$
N=64 (6-bit)	$121 \text{ trillion} = 10^{14.1} = 2^{46.8}$
N=128 (7-bit)	$163 \text{ trillion} = 10^{14.2} = 2^{47.2}$
N=256 (8-bit)	$1.5 \text{ quadrillion} = 10^{15.2} = 2^{50.4}$

Table 8. VMPC-R distant-equalities test results

N	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Event 8
2	0.00	−0.03	0.33	−0.03	−0.03	−0.48	0.10	−0.39
3	−0.28	0.05	0.52	0.47	−1.48	1.49	−0.30	−0.47
4	−0.10	−0.28	−0.03	−1.27	0.24	2.26	0.38	0.08
5	−0.69	1.33	−0.29	−1.18	−0.68	1.67	1.28	−1.01
6	0.72	−0.14	−0.80	−0.74	0.49	0.40	0.09	−0.73
7	0.27	0.34	−0.19	1.01	−0.62	0.44	2.01	0.46
8	−1.46	−0.27	−1.25	1.20	1.39	−0.65	0.56	−0.88
16	0.00	−1.99	0.82	0.46	−1.20	−0.79	−1.75	−1.06
32	1.64	−0.46	0.55	−1.18	−0.29	−0.31	−1.45	0.46
64	−0.66	0.25	0.88	−0.77	0.37	0.46	0.51	0.73
128	0.83	−1.22	0.15	0.33	0.70	−0.24	0.72	−0.07
256	0.00	−0.24	−0.78	1.34	1.47	−1.23	−0.14	0.24

In the first test we forced the algorithm to work in a single-bit mode ($N = 2$). Even with that extent of simplification the algorithm passed the tests. Here the algorithm's state generated twelve 42 output-long cycles. As an exception from the other word sizes (where the longest cycle usually comprised the majority of the state space) for $N = 2$ we averaged the results for each of the observed 12 cycles. The averages (found in the table) as well as the results for each cycle separately were well within acceptable random deviations. This test comprised 98% of the state space (504 of the 512 possible values of the state).

In the tests for $N \in \{3, 4, 5, 6\}$ we limited the analyses to the size of the longest observed cycle. This was equivalent to examining 62% 82%, 33% and 78% of the complete state space for the respective word sizes. None of the measured probabilities showed any non-random behavior in the tests. We also probed several of the remaining shorter cycles (not reported in the table) and found no anomalies there, either.

For $N = 7$ with 1.9 trillion ($2^{40.8}$) outputs tested (about 10% of the state space) we did not encounter a repeated cycle and we observed proper pseudorandom behavior in the tests.

$N = 8$ (3-bit words) was the first of the big size tests which required significant computational power. It is the smallest size which offers state space large enough ($10^{15.5}$) not to hamper the scale of the test with cycles yet it is small enough to expose possible deviations vividly. We tested over 122 trillion ($2^{46.8}$) outputs and they showed to be undistinguishable from the random model in the tests.

We continued the test with over 100 trillion-word samples for $N \in \{16, 32, 64, 128\}$ and did not find any non-random anomalies in the algorithm's output in the tests.

For $N = 258$ (8-bit) we increased the sample size to over 1.5 quadrillion outputs ($2^{50.4}$) as $N = 256$ is the actual word size for possible practical applications. The results here in all the tests were also well within the acceptable deviations from the random model.

Summarizing the experiments - we did not manage to find any non-random patterns in over 2 quadrillion outputs of VMPC-R in any of the statistical tests performed for any of the tested word sizes. In the sections to follow we try to investigate some other aspects of the algorithm's cryptographic security.

5.4 Cycle lengths of VMPC-R and Spritz

We assess the probability of VMPC-R repeating a cycle to be negligibly low for $N = 256$. The size of the internal state is determined by the size of the P and S permutations and the 7 variables a, b, c, d, e, f, n to be $N!^2 \cdot N^7$. For $N = 256$ it is over 10^{1030} or 2^{3424} possible values. Because the n variable is increased by 1 modulo N in each iteration the internal state would be changed in N iterations before it could repeat. In these iterations P and S would undergo N swap operations indexed by n, a and f while a, b, c, d, e, f would be updated N times. The probability that the variables would repeat their values after these N iterations should not be significantly different from $N!^{-2} \cdot N^{-6}$ or 2^{-3416} for $N = 256$. In our statistical tests for $N = 8$ ($2^{51.6}$ possible values of the internal state) we did not encounter a single case of a repeated cycle in the sample of $2^{46.8}$ outputs. In Table 9 we show three longest cycles we observed for $N \in \{3, 4, 5, 6\}$.

Table 9. Cycle lengths observed in VMPC-R

Word size	Total state space	3 longest cycles
$N = 3$	78 732	48 687; 6 945; 6 126
$N = 4$	9 437 184	7 766 992; 833 100; 369 056
$N = 5$	1 125 000 000	365 826 825; 219 688 515; 155 601 705
$N = 6$	145 118 822 400	113 795 459 358; 10 758 771 978; 9 768 433 476

Table 10. Cycle lengths observed in Spritz

Word size	Total state space	5 longest cycles
$N = 8$	165 150 720	84 143 080; 14 349 456; 12 020 440; 9 566 304; 6 405 880

In Spritz the state space of its PRNG function is determined by permutation S and variables i, j, k, z to be $N! \cdot N^4$. For $N = 256$ it is over 10^{516} or 2^{1716}.

The longest cycle of Spritz for $N = 8$ is 84 143 080 while for VMPC-R (and $N = 8$) it is so long that we did not encounter the end of the cycle in a sample of $2^{46.8}$ outputs. This however should not be considered any weakness of Spritz while this is only the consequence of a much larger internal state of VMPC-R (2^{3424}) compared to 2^{1716} of Spritz (for $N = 256$) which mostly results from the fact that VMPC-R uses two permutations while Spritz - a single one.

5.5 First outputs of VMPC-R

Statistical properties of the first outputs of the cipher (generated directly after the key scheduling algorithm) were targets of several distinguishing attacks. E.g. in [17] Mantin and Shamir found that the second output of RC4 is equal 0 with twice the expected probability. In [5] Paul and Preneel discovered a bias in the first two output bytes of the RC4 keystream. [13] and [10] added to those.

Anomalies here can result from the flaws in either the keystream generator or the key setup algorithm. Given the proper statistical behavior of VMPC-R output in our tests and given the security features of the key scheduling algorithm discussed in Section 6.2 we do not expect the cipher's output to behave differently directly after the key setup than it did in our statistical tests.

We additionally verified that fact by separate tests where only the first 8 outputs of the cipher were analysed using the battery described in Section 5.2. We performed the tests both for the full and for several reduced variants of the key scheduling algorithm, one of them performing only about 30% of the operations of the full KSA. None of the tests showed any statistical anomalies in the first outputs of VMPC-R.

5.6 Key/state recovery attacks against VMPC-R

Unlike in case of distinguishing attacks, RC4-like stream ciphers usually show high resistance to key (or internal state) recovery attacks. For some time the fastest state recovery algorithm for RC4 was by Mister and Tavares [14]. At CRYPTO 2008 [8] Maximov proposed an improved attack against RC4 requiring about 2^{241} operations. One difficulty in mounting these attacks against this family of ciphers is the fact that many secret words of the internal state are used to produce a single word of output. In RC4 3 words of the internal permutation are used per one output word. VMPC-R uses 11 words of its permutations to produce one output (4 elements of P and 7 of S). We roughly estimate that the total number of possible values of the unique elements of P and S used to produce 50 VMPC-R outputs would be greater than the total keyspace of a 2048-bit (256-byte) secret key. We don't expect the key/state recovery attacks to be a significant threat to the security of the proposed cipher.

Another area where the algorithm's complexity might strengthen its resistance is attacks derived from fortuitous states introduced by Fluhrer and McGrew. In [16] they presented RC4 states in which only X elements of the permutation are involved in the computation of X successive outputs. They showed how this fact can be used to determine some parts of the permutation with nontrivial probability. Paul and Preneel note in [6] that their attack approach has its origins in the fortuitous states. We believe that VMPC-R using 11 permutation elements per round controlled by the set of 7 interdependent variables would make introducing fortuitous states significantly harder, if possible.

We also expect the level of complexity of the round function to be a considerable obstacle in mounting fault analysis attacks against VMPC-R as opposed to RC4 for which successful fault attacks were published by Hoch and Shamir in 2004 [11] or Biham, Granboulan and Nguyen in 2005 [12].

6 Key scheduling algorithm of VMPC-R

Table 11. VMPC-R key scheduling algorithm

$N, P, S, a, b, c, d, e, f, n$: as in Table 4 k : key size; $k \in \{1, 2, ..., N\}$ K : key; array of k integers v : initialization vector size; $v \in \{1, 2, ..., N\}$ V : initialization vector; array of v integers $R = \lceil k^2/(6N) \rceil \cdot N$ (number of rounds) for $N = 256$ and $k \in \{1, 2, ..., 39\}$ (keys up to 312 bits): $R = N$ i : temporary integer variable $+$ denotes addition modulo N
0. $a = b = c = d = e = f = n = 0$ $P[i] = S[i] = i$ for $i \in \{0, 1, ..., N-1\}$ 1. $KSARound(K, k)$ 2. $KSARound(V, v)$ 3. $KSARound(K, k)$ 4. $n = S[S[S[c + d]] + 1]$ 5. generate N outputs with VMPC-R (Table 4)
Function $KSARound(M, m)$ definition: 6. $i = 0$ 7. repeat steps 8-18 R times: 8. $a = P[a + f + M[i]] + i;$ $i = (i + 1) \mod m$ 9. $b = S[b + a + M[i]] + i;$ $i = (i + 1) \mod m$ 10. $c = P[c + b + M[i]] + i;$ $i = (i + 1) \mod m$ 11. $d = S[d + c + M[i]] + i;$ $i = (i + 1) \mod m$ 12. $e = P[e + d + M[i]] + i;$ $i = (i + 1) \mod m$ 13. $f = S[f + e + M[i]] + i;$ $i = (i + 1) \mod m$ 14. swap $P[n]$ with $P[b]$ 15. swap $S[n]$ with $S[e]$ 16. swap $P[d]$ with $P[f]$ 17. swap $S[a]$ with $S[c]$ 18. $n = n + 1$

6.1 Some comments on the design

The number of rounds (R). The *KSARound* function performs $R = \lceil k^2/(6N) \rceil \cdot N$ iterations. This value ensures that each word of a k-word key updates the internal state at least k times. This follows an intuition that the probability of deriving identical internal states from keys differing in only one bit or one word should not be different from the probability $N!^{-2} \cdot N^{-7}$ of deriving identical internal states from two random keys. For $N = 256$ and key sizes $k \in \{1, 2, ..., 39\}$ (keys up to 312 bits): $R = N$. For $N = 256$ and key sizes $k \in \{40, 41, ..., 55\}$ (keys from 320 to 440 bits) $R = 2N$. And so on.

Step 3 of the algorithm. Apart from intensifying the avalanche effect (discussed in Section 6.2) step 3 provides an additional layer of security against key-recovery attacks.

Let's consider a reduced version without step 3 and assume that a successful state recovery attack has been accomplished. Since all the data steps 2-5 operate on is known by the adversary it would be possible to revert these steps and obtain the value of the internal state after step 1. This would enable to process steps 2-5 for any new message and decrypt it without finding the actual secret key.

In the full KSA this attack would provide the internal state after step 3. However here the adversary would face a hard problem of reverting step 3 which takes the secret key as a parameter. As a result of the avalanche effect any new message encrypted with the same key but different initialization vector would have the internal state uncorrelated with the one the adversary holds which would give the adversary no advantage in decrypting any new message.

Step 5 of the algorithm. A number of approaches against RC4-like ciphers exploited the internal state right after the key-setup to mount distinguishing attacks. Some were mentioned in Section 5.5. One idea to improve the RC4 KSA was to drop several first outputs of the cipher. Although we believe VMPC-R KSA provides proper statistical behavior, we propose to add this step as an additional layer - just in case. A measurable benefit it provides is the intensification of the avalanche effect.

No non-secret values. Our objective was to keep all the state variables secret. Leaving the value of the counter variable (n in our algorithm) known raises unnecessary risk as the adversary knows the indices to the arrays used by some parts of the cipher's round function. In fact several attacks mentioned in Section 5.5 exploit this situation. Although we don't believe this to be a significant security improvement, the effort required to ensure this property (setting variable n to a secret value in step 4) is very small.

No obvious equivalent keys. To make introducing different keys producing the same internal state a harder task we use both the value of the word of the key ($M[i]$) and its index i as input. Both are mixed non-linearly in the $a = P[a + f + M[i]] + i$ step 8 (and analogously in steps 9-13).

Flexibility. The KSA's compatibility with a 256-byte array might open more possible alternative applications than it would be had the key size been limited to the more reasonable 256 or 512 bits. Such arrays containing additional information could be used to influence the internal state. Apart from the key and the initialization vector, the algorithm can accept any amount of parameters like additional or session keys, personal information or hardware ID by calling *KSARound* with these parameters after step 2 and before step 3.

6.2 Statistical properties of the KSA

We investigated the statistical properties of the KSA in two areas. One was whether the elements of the generated permutations and the variables follow the uniform distribution. The second was whether permutations and variables generated from different keys are uncorrelated. We performed all the tests on the KSA only for the full $N = 256$ version.

To verify the results in the first area we measured the probabilities of occurrences of each of the possible N values in all the N indices in both permutations (P and S) and in all the 7 variables (a, b, c, d, e, f, n) for different key sizes. All the 132864 measured probabilities stayed well within acceptable random deviations from the expected $1/N$.

The second objective was to verify whether the slightest change in the input key would cause the proper avalanche effect. We approached this test according to the strict avalanche criterion (SAC) defined by Webster and Tavares in [19]. The criterion is satisfied when changing a single input bit changes each output bit with probability 0.5. We measured the number of unchanged corresponding elements of the permutations and the number of cases of unchanged variables generated with two (slightly) different keys.

We tested the consequences of a change of a single byte (this included flipping a single bit) in a pseudorandom key and changing a byte in four different types of non-random keys. We then repeated the tests with the operation of changing the byte substituted by an operation of appending a new byte to the key.

To get the worst-case result we chose several scenarios which appeared to hinder the avalanche effect the most. All the tested keys were of maximum length $k = 256$ bytes (2048 bits) and we changed the byte of the key which was the last to be input to the KSA. In any other case (smaller keys or changing any other byte of the key) the avalanche effect would be more intensive as the changes of the key would be digested by the algorithm more times or earlier. Additionally we reduced the number of rounds of the $KSARound$ function from the original $R = 11008$ (for $k = N = 256$) to only $R = 256$. We chose the following pairs of keys for the test:

- K_1: 256 random bytes; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255] \neq K_1[255]$
- K_1: 255 random bytes; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255]$=random
- $K_1[i] = i$, $i \in \{0, ..., 255\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255] \neq K_1[255]$
- $K_1[i] = i$, $i \in \{0, ..., 254\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255]$=random
- $K_1[i] = 255 - i$, $i \in \{0, ..., 255\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255] \neq K_1[255]$
- $K_1[i] = 255 - i$, $i \in \{0, ..., 254\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255]$=random
- $j \in \{0, ..., 255\}$: $K_1[i] = j$, $i \in \{0, ..., 255\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255] \neq K_1[255]$
- $j \in \{0, ..., 255\}$: $K_1[i] = j$, $i \in \{0, ..., 254\}$; $K_2[i] = K_1[i]$, $i \in \{0, ..., 254\}$; $K_2[255]$=random

To obtain a significant security margin for the measured avalanche effect we performed the tests both for the full version of the KSA and for the reduced version using only steps 1 and 5. In the tests we compared the generated permutations and the 7 variables generated by the KSA for K_1 and K_2 and found that in each test the numbers of unchanged elements in the resulting permutations were well within acceptable random deviations from the expected 1 and that the probability that any of the 7 variables remained unchanged was not significantly deviated from the expected $1/N$.

Step 1 alone almost provided the proper avalanche effect. The result we recorded for step-1-only variant was about 1.3 unchanged corresponding elements of the generated permutations

rather than 1 we would expect. For shorter 128-byte (1024-bit) keys the tests for step-1-only variant produced even better results at around 1.05 identical corresponding permutation elements (as expected, the shorter key with the other factors unchanged intensified the avalanche effect).

The KSA limited only to steps 1 and 5 was enough to achieve 1 in all the tests. Steps 2 and 3 approximately double the amount of mixing performed by steps 1 and 5 - intensifying the avalanche effect and providing the full algorithm with a significant security margin in terms of the avalanche effect.

7 Test values

Tables 12-14 contain example output values of the algorithms.

Table 12. Test-output of VMPC-R (1)

input:								
K; $k = 9$	{11, 22, 33, 144, 155, 166, 233, 244, 255}							
V; $v = 8$	{255, 250, 200, 150, 100, 50, 5, 1}							
output of KSA:								
P index	0	1	2	3	252	253	254	255
P value	97	218	106	125	139	86	36	126
S index	0	1	2	3	252	253	254	255
S value	152	143	19	154	92	25	24	157
output of CSPRNG:								
output index	0	1	2	3	254	255	256	257
output value	49	161	79	69	85	237	96	243
output index	1000	1001	10000	10001	100000	100001	1000000	1000001
output value	181	184	136	99	67	27	253	231

Table 13. Test-output of VMPC-R (2)

input:								
K; $k = 32$	{104, 9, 46, 231, 132, 149, 234, 147, 224, 97, 230, 127, 124, 109, 34, 171, 88, 185, 158, 23, 116, 69, 90, 195, 208, 17, 86, 175, 108, 29, 146, 219} (X=123; repeat 32 times:{$X = X \cdot 134775813+1$; output=X mod 256})							
V; $v = 32$	{149, 234, 147, 224, 97, 230, 127, 124, 109, 34, 171, 88, 185, 158, 23, 116, 69, 90, 195, 208, 17, 86, 175, 108, 29, 146, 219, 72, 105, 14, 71, 100} (X=132; repeat 32 times:{$X = X \cdot 134775813+1$; output=X mod 256})							
output of KSA:								
P index	0	1	2	3	252	253	254	255
P value	76	44	167	7	250	147	240	51
S index	0	1	2	3	252	253	254	255
S value	239	59	110	207	98	23	178	227
output of CSPRNG:								
output index	0	1	2	3	254	255	256	257
output value	219	178	157	119	2	155	62	20
output index	1000	1001	10000	10001	100000	100001	1000000	1000001
output value	3	239	236	81	195	11	186	127

Table 14. Test-output of VMPC-R (3)

input:								
K; $k = 256$	{147, 224, 97, 230, 127, 124, 109, 34, 171, 88, 185, 158, 23, 116, 69, 90, 195, 208, 17, 86, 175, 108, 29, 146, 219, 72, 105, 14, 71, 100, 245, 202, 243, 192, 193, 198, 223, 92, 205, 2, 11, 56, 25, 126, 119, 84, 165, 58, 35, 176, 113, 54, 15, 76, 125, 114, 59, 40, 201, 238, 167, 68, 85, 170, 83, 160, 33, 166, 63, 60, 45, 226, 107, 24, 121, 94, 215, 52, 5, 26, 131, 144, 209, 22, 111, 44, 221, 82, 155, 8, 41, 206, 7, 36, 181, 138, 179, 128, 129, 134, 159, 28, 141, 194, 203, 248, 217, 62, 55, 20, 101, 250, 227, 112, 49, 246, 207, 12, 61, 50, 251, 232, 137, 174, 103, 4, 21, 106, 19, 96, 225, 102, 255, 252, 237, 162, 43, 216, 57, 30, 151, 244, 197, 218, 67, 80, 145, 214, 47, 236, 157, 18, 91, 200, 233, 142, 199, 228, 117, 74, 115, 64, 65, 70, 95, 220, 77, 130, 139, 184, 153, 254, 247, 212, 37, 186, 163, 48, 241, 182, 143, 204, 253, 242, 187, 168, 73, 110, 39, 196, 213, 42, 211, 32, 161, 38, 191, 188, 173, 98, 235, 152, 249, 222, 87, 180, 133, 154, 3, 16, 81, 150, 239, 172, 93, 210, 27, 136, 169, 78, 135, 164, 53, 10, 51, 0, 1, 6, 31, 156, 13, 66, 75, 120, 89, 190, 183, 148, 229, 122, 99, 240, 177, 118, 79, 140, 189, 178, 123, 104, 9, 46, 231, 132, 149, 234} (X=234; repeat 256 times:{$X = X \cdot 134775813+1$; output=X mod 256})							
V; $v = 8$	{255, 250, 200, 150, 100, 50, 5, 1}							
output of KSA:								
P index	0	1	2	3	252	253	254	255
P value	10	34	13	239	209	9	154	220
S index	0	1	2	3	252	253	254	255
S value	253	106	200	178	75	251	129	209
output of CSPRNG:								
output index	0	1	2	3	254	255	256	257
output value	201	85	155	17	187	48	55	198
output index	1000	1001	10000	10001	100000	100001	1000000	1000001
output value	110	179	189	210	4	15	253	83

8 Conclusions

We applied a simple distant-equalities statistical test to analyse the output of three RC4-like ciphers: RC4, Spritz and VMPC-R. The results for RC4 were very bad. Spritz revealed a bias which can be detected for $N = 8$ after observing about $2^{21.9}$ output words. We confronted these results with those of VMPC-R. This algorithm was created in a research project aimed at finding the simplest RC4-like cipher capable of passing the distant-equalities test. En route to the final version of VMPC-R over 250 RC4-like variants were analysed, some using a single permutation and some using two permutations along with several integer variables. As a result we proposed an algorithm which passed the distant-equalities test without revealing any bias in a sample of $2^{46.8}$ outputs - 31 million times larger than that needed to reveal the bias in Spritz (for $N = 8$). VMPC-R also passed a number of additional statistical tests and showed no security problems to the extent we were able to analyse it. We hope that VMPC-R can be considered a worthwhile candidate to replace RC4.

References

1. Ronald L. Rivest, Jacob C. N. Schuldt: Spritz - a spongy RC4-like stream cipher and hash function. http://people.csail.mit.edu/rivest/pubs/RS14.pdf. Presented at CRYPTO 2014 Rump Session.
2. Bartosz Zoltak: VMPC One-Way Function and Stream Cipher. Proceedings of FSE 2004, LNCS, vol. 3017, Springer-Verlag, 2004, pages 210-225.
3. Bartosz Zoltak: VMPC-R Cryptographically Secure Pseudo-Random Number Generator Alternative to RC4. Cryptology ePrint Archive: Report 2013/768. http://eprint.iacr.org/2013/768.
4. Bartosz Zoltak: Statistical weaknesses in 20 RC4-like algorithms and (probably) the simplest algorithm free from these weaknesses - VMPC-R. Cryptology ePrint Archive: Report 2014/315. http://eprint.iacr.org/2014/315.
5. Souradyuti Paul, Bart Preneel A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. Proceedings of FSE 2004, LNCS, vol. 3017, Springer-Verlag, 2004, pages 245-259.
6. On the (In)security of Stream Ciphers Based on Arrays and Modular Addition Souradyuti Paul and Bart Preneel Proceedings of ASIACRYPT 2006, LNCS, vol. 4284, Springer-Verlag, 2006, pages 69-83.
7. Alexander Maximov: Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers. Proceedings of FSE 2005, LNCS, vol. 3557, Springer-Verlag, 2005, pages 342-358.
8. Alexander Maximov, Dmitry Khovratovich: New State Recovery Attack on RC4 Proceedings of CRYPTO 2008, LNCS, vol. 5157, Springer-Verlag, 2008, pages 297-316.
9. Itsik Mantin: Predicting and Distinguishing Attacks on RC4 Keystream Generator. Proceedings of Eurocrypt 2005, LNCS vol. 3494 of LNCS, Springer-Verlag, 2005, pages 491-506
10. Yukiyasu Tsunoo, Teruo Saito, Hiroyasu Kubo, Maki Shigeri, Tomoyasu Suzaki, Takeshi Kawabata: The Most Efficient Distinguishing Attack on VMPC and RC4A ECRYPT Stream Cipher Project, Report 2005 / 037
11. Jonathan J. Hoch, Adi Shamir Fault Analysis of Stream Ciphers. Proceedings of CHES 2004, LNCS, vol. 3156, Springer-Verlag, 2004
12. Eli Biham, Louis Granboulan, Phong Q. Nguyen: Impossible Fault Analysis of RC4 and Differential Fault Analysis of RC4. Proceedings of FSE 2005, LNCS, vol. 3557, Springer-Verlag, 2005, pages 359-367.
13. Scott Fluhrer, Itsik Mantin, Adi Shamir: Weaknesses in the key scheduling algorithm of RC4. Proceedings of SAC 2001, LNCS, vol. 2259, Springer-Verlag 2001.
14. Serge Mister, Stafford E. Tavares: Cryptanalysis of RC4-like Ciphers. Proceedings of SAC 1998, LNCS, vol. 1556, Springer-Verlag, 1999.
15. Lars R. Knudsen, Willi Meier, Bart Preneel, Vincent Rijmen, Sven Verdoolaege: Analysis Methods for (Alleged) RC4. Proceedings of ASIACRYPT 1998, LNCS, vol. 1514, Springer-Verlag, 1998.
16. Scott R. Fluhrer, David A. McGrew: Statistical Analysis of the Alleged RC4 Keystream Generator. Proceedings of FSE 2000, LNCS, vol. 1978, Springer-Verlag, 2001.
17. Itsik Mantin, Adi Shamir: A Practical Attack on Broadcast RC4. Proceedings of FSE 2001, LNCS, vol. 2355, Springer-Verlag, 2002.
18. Jovan Dj. Golic: Linear Statistical Weakness of Alleged RC4 Keystream Generator. Proceedings of EUROCRYPT 1997, LNCS, vol. 1233, Springer-Verlag, 1997.
19. A. F. Webster; Stafford E. Tavares. On the design of S-boxes. Proceedings of CRYPTO 1985, LNCS, vol. 218, Springer-Verlag, 1986.