VMPC Stream Cipher

Bartosz Zoltak, bzoltak@vmpcfunction.com; bzoltak@wp.pl

Abstract. The VMPC Stream Cipher is a simple encryption algorithm, designed as a proposed practical application of the VMPC one-way function. The general structure of the Cipher is based on an internal 256-element permutation. The VMPC Cipher, together with its Key Scheduling Algorithm, were designed in particular to eliminate some of the known weaknesses characteristic of the alleged RC4 keystream generator.

1. Introduction

VMPC is an abbreviation of Variably Modified Permutation Composition.

The VMPC function is a combination of triple permutation composition and integer addition. It differs from a simple triple permutation composition with one integer addition operation performed on some of the elements of the permutation. The consequence of this addition operation is corruption of cycle structure of the transformed permutation - the fundamental source of the function's resistance to inverting.

The VMPC function has a simple formal definition and the value of the function can be computed with 3 one-clock-cycle instructions of an Intel 80486 and newer or compatible processor per byte.

Inverting the simplest variant of the function by the fastest known inverting algorithm is estimated to require an average computational effort of about 2^{260} operations.

The VMPC Stream Cipher is based on the VMPC function. Because the requirement for a stream cipher is that its output is undistinguishable from a random-data-stream, the Cipher employs two other mechanisms, apart from the computation of the VMPC function. They are updates of an internal 8-bit variable (s) and a swap operation on some elements of the internal permutation (P).

The Key Scheduling Algorithm (KSA) of the VMPC Stream Cipher transforms a cryptographic key of length from 128 to 512 bits (and an Initialization Vector (IV)) into a 256-element internal permutation (P).

2. The VMPC function

For a detailed description of the VMPC function, please refer to "VMPC One-Way Function" by Bartosz Zoltak (possible to download from http://www.VMPCfunction.com or from http://eprint.iacr.org).

2.1. Definition of the VMPC function

Notation:

n, P, Q : P and Q: n-element permutations. For simplicity of further implementations P and Q are one-to-one mappings $A \rightarrow A$, where $A = \{0,1,\ldots,n-1\}$

k : Level of the function; $k<n$

+ : addition modulo n

Definition:

A k-level VMPC function, referred to as $VMPC_k$, is such transformation of P into Q, where

$$\mathbf{Q[\ x\ ] = P\ [\ P_k\ [\ P_{k-1}\ [\ldots[\ P_1\ [\ P\ [\ x\ ]\ ]\ ]\ \ldots]\ ]\ ],}$$

$x \in \{0,1,\ldots,n-1\}$,

P_i is an n-element permutation such that $P_i[x] = f_i\,(P[x])$, where f_i is any function such that $P_i[x] \neq P[x] \neq P_j[x]$ for $i \in \{1\ldots k\}$, $j \in \{1\ldots k\} / \{\ i\ \}$.

For simplicity of further implementations f_i is assumed to be $f_i\,(x) = x + i$

For simplicity of future references notation Q=VMPC(P) is assumed to be equivalent to $Q=VMPC_1(P)$

<u>*Example:*</u>

$Q=VMPC_1(P)$ is such transformation of P into Q, where:

$Q[\ x\ ] = P[P_1[P[x]]]$,

$P_1[x]=P[x]+1$.

($Q[x]=P[P[P[x]]+1]$, where "+" denotes addition modulo n)

2.2. Difficulty of inverting the VMPC function

n-element permutation P has to be recovered given information from n-element permutation, Q, where Q=VMPC$_k$(P) (e.g. n=256, k=1: Q[x]=P[P[P[x]]+1]).

By definition each element of Q is formed by k+2 (e.g. 3), usually different, elements of P. One element of Q (e.g. Q[33]=25) can be formed by many possible configurations of P elements (e.g. P[33]=10, P[10]=20, P[21]=25 or P[33]=1, P[1]=4, P[5]=25, etc.).

It cannot be said which of the configurations is more probable. One of the configurations has to be picked (usually k+1 (e.g. 2) elements of P have to be guessed) and the choice must be verified using all those other Q elements, which use at least one of the P elements from the picked configuration.

Each element of P is usually used to form k+2 (e.g. 3) different elements of Q. As a result, usually (k+2)*(k+1) (e.g. 6) new elements of Q need to be inverted (all k+2 elements of P used to form each of those Q elements need to be revealed) to verify the P elements from the picked configuration.

This would not be difficult for a simple (e.g. triple) permutation composition, where the cycle structure of P is retained by Q (some cycles are only shortened).

In Variably Modified Permutation Composition however the cycle structure of P is corrupted by the addition operation(s) and cannot be easily recovered from Q.

Due to that it is usually impossible to find two different elements of Q, which use at least k+1 (e.g. 2) exactly the same elements of P. (This can be done easily for a simple permutation composition)

In fact only such element of Q can usually be found, name it Q[r], which uses only one of the k+2 (e.g. 3) elements of P, used to form another Q element. This forces the k remaining (e.g. 1) elements of P, used to form Q[r], to be guessed to make the verification of the initial pick possible.

However at each new guessed element of P, there usually occur k+1 (e.g. 2) new elements of Q which use this element of P and which need to be inverted to verify the guess.

The algorithm falls into a loop, where at every step usually k (e.g. 1) new elements of P need to be guessed to verify the previously guessed elements. It quickly occurs that the k+2 (e.g. 3) elements of P picked at the beginning of the process indirectly depend on all n (e.g. 256) elements of Q.

The described scenario is the case usually and it is sometimes possible to benefit from coincidences (where for example it is possible to find two elements of Q, which use more than one (e.g. 2) exactly the same P elements (e.g. Q[2]=3: P[2]=4, P[4]=8, P[9]=3 and Q[5]=8: P[5]=9, P[9]=3, P[4]=8)).

The actual algorithm of inverting VMPC was optimized to benefit from the possible coincidences. The average number of P elements which need to be guessed - for n=256 - has been reduced to only about 34 for 1-level VMPC function, to about 57 for 2-level VMPC, to about 77 for 3-level VMPC and to about 92 for 4-level VMPC function.

Searching through half of the possible states of these P elements takes on average about 2^{260} steps for 1-level VMPC function, about 2^{420} for 2-level VMPC, about 2^{550} for 3-level VMPC and about 2^{660} steps for 4-level VMPC function.

A detailed algorithm of inverting the VMPC function is described in “VMPC One-Way Function” by Bartosz Zoltak.

3. Design objectives for the VMPC Stream Cipher and its KSA

The Cipher should not generate biased digraph probabilities (characteristic of RC4, as described by Fluhrer and McGrew in [5]) or biased trigraph probabilities or biased single-output probabilities.

The Cipher should require no initial outputs to be discarded directly after running the KSA.

Probability that the Cipher’s output will enter a short cycle should be negligibly low.

Output generated by the Cipher should be free from statistical biases.

Effort required to recover the internal state from the Cipher's output should be higher than a brute-force search of all possible 512-bit keys.

The KSA should resist related-key attacks and attacks against the scheme of using the Initialization Vector (IV), like attacks described by Fluhrer, Mantin and Shamir in [4] including the WEP attack.

The KSA should provide random-like diffusion of changes of one byte of the key of size up to 512 bits onto the generated permutation and onto output generated by the Cipher.

4. Description of the VMPC Stream Cipher

The Cipher generates a stream of 8-bit values from a 256-element permutation. The initial state of the permutation is determined by the VMPC Key Scheduling Algorithm described in section 5.

Notation:

P : 256-byte table storing the permutation

s : 8-bit variable initialized by the VMPC Key Scheduling Algorithm

n : 8-bit variable

Table 1.1. VMPC Stream Cipher

1. Set n to 0
2. Add modulo 256 n-th element of P to s **3. Set s to s-th element of P** **4. Output s-th element of permutation VMPC(P)** **5. Swap n-th element of P with s-th element of P** **6. Increment modulo 256 n**
7. Go to step 2 if more output is needed

Table 1.2. VMPC Stream Cipher – pseudo code

```
To generate Len bytes of output execute:
1. n = 0
2. Repeat steps 3-6 Len times:
   3. s = P[ (s + P[n]) and 255 ]
   4. Output = P[ (P[P[s]]+1) and 255 ]
   5. Temp = P[n]
      P[n] = P[s]
      P[s] = Temp
   6. n = (n + 1) and 255
```

5. Description of the VMPC Key Scheduling Algorithm

The VMPC Key Scheduling Algorithm transforms a cryptographic key and (optionally) an Initialization Vector into a 256-element permutation P.

Notation: as in section 4, with:

c : fixed length of the cryptographic key in bytes, $c \in \{16...64\}$

K : c-element table storing the cryptographic key

z : fixed length of the Initialization Vector in bytes, $z \in \{16...64\}$

V : z-element table storing the Initialization Vector

m : 16-bit variable

Table 2.1. VMPC Key Scheduling Algorithm

1. Set s to 0 **2. Set i-th element of P to i for $i \in \{0,1,...,255\}$**
3. Set m to 0 **4. Add modulo 256 (m modulo 256)-th element of P to s** **5. Add modulo 256 (m modulo c)-th element of K to s** **6. Set s to s-th element of P** **7. Swap (m modulo 256)-th element of P with s-th element of P** **8. Increment m** **9. Go to step 4 if m is lower than 768**
10. If Initialization Vector is not used: terminate the algorithm
11. Set m to 0 **12. Add modulo 256 (m modulo 256)-th element of P to s** **13. Add modulo 256 (m modulo z)-th element of V to s** **14. Set s to s-th element of P** **15. Swap (m modulo 256)-th element of P with s-th element of P** **16. Increment m** **17. Go to step 12 if m is lower than 768**

Table 2.2. VMPC Key Scheduling Algorithm – pseudo code

```
1. s = 0
2. for i from 0 to 255: P[i]=i

3. for m from 0 to 767: execute steps 4-6:
   4. n = m and 255
   5. s = P[ (s + P[n] + K[m mod c]) and 255 ]
   6. Temp = P[n]
      P[n] = P[s]
      P[s] = Temp

7. If Initialization Vector is used: execute step 8:

8. for m from 0 to 767: execute steps 9-11:
    9. n = m and 255
   10. s = P[ (s + P[n] + V[m mod z]) and 255 ]
   11. Temp = P[n]
       P[n] = P[s]
       P[s] = Temp
```

6. Analysis of the VMPC Stream Cipher

6.1. Recovering the Cipher's internal state

Over 2^{900} operations are estimated to be required to recover the Cipher's internal state from its output. A method similar in its foundations to the Forward Tracking Algorithm, proposed by S. Mister and S.E. Tavares in [7], was applied to break the VMPC Stream Cipher. On average over 2^{900} computational operations are estimated to be required to recover the Cipher's internal state from its output.

6.2. Digraph and trigraph probabilities

Frequencies of occurrence of each of the possible 2^{16} pairs of consecutive output values (Output[x], Output[x+1]) were measured in a stream of $2^{40.1}$ output bytes. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 65536.

Frequencies of occurrence of each of the possible 2^{24} triplets of two consecutive output values and the n variable (Output[x], Output[x+1], n) were measured in a stream of $2^{41.85}$ output bytes. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 16777216.

Frequencies of occurrence of each of the possible 2^{24} trigraphs of consecutive output values (Output[x], Output[x+1], Output[x+2]) were measured in a stream of $2^{41.6}$ output bytes. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 16777216.

6.3. Single output probabilities

Frequencies of occurrence of each of the possible 2^8 output values (Output[x]) were measured in a stream of $2^{41.85}$ output bytes. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 256.

Frequencies of occurrence of each of the possible 2^{16} configurations of an output value and the n variable (Output[x], n) were measured in a stream of $2^{39.4}$ output bytes. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 65536.

6.4. First outputs probabilities

Frequencies of occurrence of each of the possible 2^8 values on each of the first 256 byte-positions of the keystream generated directly after running the KSA were measured in a sample of $2^{40.3}$ bytes of the Cipher's output for $2^{32.3}$ different keys. None of the measured frequencies showed a statistically significant deviation from its expected value of 1 / 256.

[In [1] Mantin and Shamir show that the second output of RC4 takes on value 0 with probability 1/128 rather than 1/256.]

6.5. Short cycles

6.5.1. Probability of entering a cycle no longer than X

Following Knuth's [8], probability of entering a cycle no longer than X for an n-element random permutation is X/n.

To compare cycle lengths in the output of the VMPC Stream Cipher to cycle lengths in a random permutation, the Cipher was scaled down to use N element permutations for $N \in \langle 4,10 \rangle$ and perform all addition operations modulo N.

The total number of $N!*N^2$ possible internal states of the Cipher is determined by all possible configurations of permutation P and variables s and n.

The observed cycle lengths, listed in the Appendix, do not show an appreciable difference from a model of cycles in a random $(N!*N^2)$-element permutation.

Probability of entering a cycle no longer than X by the VMPC Stream Cipher is conjectured from this to be approximately $X / (256!*256^2)$. An example estimate is that probability that the Cipher's output will enter a cycle no longer than 2^{1000} is about $1 / 2^{700}$.

6.5.2. Finney states

In [6] Finney defined a theoretical class of internal states of RC4 which produce a short cycle of length 65280 by swapping P[n]=1 in each step (the KSA of RC4 prevents the cipher's internal state from entering this class). The class is diagnosed by n+1=s and P[n+1]=1.

Such phenomena is possible because step s=s+P[n] of the state-transformation function of RC4 retains the linear structure of P[n] in variable s (P[n], after the increment of n, is always equal 1).

The VMPC Stream Cipher uses an additional table-lookup (s=P[s+P[n]], which, assuming that P was properly initialized, corrupts a possible linear structure of P[n] (or s) and prevents situations analogous to Finney states from occurring.

6.6. Equal neighboring outputs probabilities

Frequencies of occurrence of situations where there occurs a given number (0,1,2,3,4,5 and over 5) of direct (generated consecutively) and indirect (separated by one more output) equal neighboring outputs in the consecutive 256-byte sub-streams of the Cipher's output and the average total number of direct and indirect equal neighbors - showed no statistically significant deviation from their expected values in a sample of $2^{43.1}$ bytes of the Cipher's output.

6.7. Statistical tests on the Cipher's output

Keystreams generated by the VMPC Cipher were tested by two popular batteries of statistical tests – the DIEHARD battery [9] and the NIST statistical tests suite [10]. No bias was found by any of the 15 tests included in the DIEHARD battery and by any of the 16 tests from the NIST suite.

7. Analysis of the VMPC Key Scheduling Algorithm

The VMPC KSA has been tested for diffusion of changes of the cryptographic key onto the generated permutation and onto the Cipher's output. A change of one byte of the cryptographic key of size 128, 256 and 512 bits shows to cause a random-looking change in the generated permutation and in the VMPC Cipher's output – according to tests described in sections 7.1, 7.2 and 7.3.

The KSA has been designed to provide the diffusion without the use of the Initialization Vector and tests were run without the IV. The Initialization Vector would obviously mix the generated permutation further, which would improve the diffusion effect.

7.1. Numbers of equal permutation elements probabilities

Frequencies of occurrence of situations where in two permutations, generated from keys differing with one byte, there occurs a given number (0,1,2,4,5) of equal elements on the corresponding positions and the average number of equal elements on the corresponding positions - showed no statistically significant deviation from their expected values in samples of $2^{33.2}$ pairs of 128, 256 and 512-bit keys.

7.2. Numbers of equal Cipher's outputs probabilities

Frequencies of occurrence of situations where in two 256-byte streams generated by the VMPC Stream Cipher directly after running the VMPC KSA for keys differing with one byte, there occurs a given number (0,1,2,4,5) of equal elements on the corresponding positions and the average number of equal elements on the corresponding positions - showed no statistically significant deviation from their expected values in samples of $2^{33.2}$ pairs of 128, 256 and 512-bit keys.

7.3. Equal corresponding permutation elements probabilities

Frequencies of occurrence of situations where the elements separately on each of the corresponding positions of the permutations, generated from keys differing with one byte, are equal - showed no statistically significant deviation from their expected values in samples of $2^{33.2}$ pairs of 128, 256 and 512-bit keys.

8. Conclusions

A proposition of a stream cipher which employs the VMPC one-way function has been described together with some analyses of the cipher's cryptographic strength, of the statistical properties of the cipher's output and of the statistical properties of the cipher's Key Scheduling Algorithm.

The analyses performed so far show that the cipher is secure in a sense of difficulty of recovering its internal state from its output, in a sense of difficulty of distinguishing the cipher's output from a random data-stream and from the standpoint of statistical properties of the cipher's KSA.

More detailed descriptions of the tests outlined in sections 6.6, 7.1, 7.2, 7.3, test vectors and the current developments in the analysis of the VMPC Stream Cipher are to be found at http://www.VMPCfunction.com.

Appendix. Cycle lengths observed in the output of the VMPC Stream Cipher

The observed cycle lengths in the output of the scaled down variants of the Cipher for N ∈ <4,10> are listed in Table A.1. N denotes the number of elements in the P permutation. All addition operations performed by the scaled down variants of the Cipher are additions modulo N.

Table A.1. VMPC Stream Cipher cycle lengths

N	*Cycle lengths*
4	200, 88, 40, 36, 12, 8
5	1 860, 640, 295, 110, 45, 25, 20, 5
6	15 510, 5 580, 2 508, 936, 516, 510, 252, 90, 12, 6
7	215 089, 23 821, 3 990, 2 485, 1 015, 392, 70, 56, 28, 14
8	2 401 728, 79 504, 53 512, 42 120, 2 136, 1 032, 288, 96, 24, 16 (2 different cycles of length 16 possible), 8
9	20 355 471, 2 908 098, 2 728 890, 1 359 855, 949 725, 609 174, 299 592, 125 091, 27 306, 13 068, 6 219, 5 067, 2 853, 2 538, 180, 90, 18 (3 different cycles of length 18 possible), 9
10	113 748 840, 99 425 590, 75 813 290, 37 178 940, 20 169 740, 9 955 030, 3 239 140, 2 349 150, 572 500, 363 830, 45 520, 8 730, 7 520, 700, 390, 370, 40 (17 different cycles of length 40 possible), 20, 10 (2 different cycles of length 10 possible)

10. References

[1] Itsik Mantin, Adi Shamir, "A Practical Attack on Broadcast RC4"

[2] Alexander L. Grosul, Dan S. Wallach, "A Related-Key Cryptanalysis of RC4"

[3] Lars R. Knudsen, Willi Meier, Bart Preneel, Vincent Rijmen, Sven Verdoolaege, "Analysis Methods for (Alleged) RC4"

[4] Scott Fluhrer, Itsik Mantin, Adi Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4"

[5] Scott R. Fluhrer, David A. McGrew, "Statistical Analysis of the Alleged RC4 Keystream Generator"

[6] H. Finney, "An RC4 Cycle That Can't Happen"

[7] S.Mister, S.E. Tavares, "Cryptanalysis of RC4-like Ciphers"

[8] Donald E. Knuth "The Art of Computer Programming", vol. 1. *Fundamental Algorithms,* Third Edition, Addison Wesley Longman, 1997.

[9] DIEHARD battery of statistical tests with documentation, http://stat.fsu.edu/~geo/diehard.html

[10] NIST statistical tests suite with documentation, http://csrc.nist.gov/rng