Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers.

Alexander Maximov

Dept. of Information Technology, Lund University, Sweden
P.O. Box 118, 221 00 Lund, Sweden
movax@it.lth.se

Abstract. At FSE 2004 two new stream ciphers VMPC and RC4A have been proposed. VMPC is a generalisation of the stream cipher RC4, whereas RC4A is an attempt to increase the security of RC4 by introducing an additional permuter in the design. This paper is the first work presenting attacks on VMPC and RC4A. We propose two linear distinguishing attacks, one on VMPC of complexity 2^{54}, and one on RC4A of complexity 2^{58}. We investigate the RC4 family of stream ciphers and show some theoretical weaknesses of such constructions.

Keywords: RC4, VMPC, RC4A, cryptanalysis, linear distinguishing attack.

1 Introduction

Stream ciphers are very important cryptographic primitives. Many new designs appear at different conferences and proceedings every year. In 1987, Ron Rivest from RSA Data Security, Inc. made a design of a byte oriented stream cipher called RC4 [1]. This cipher found its application in many Internet and security protocols. The design was kept secret up to 1994, when the alleged specification of RC4 was leaked for the first time [2]. Since that time many cryptanalysis attempts were done on RC4 [3–7].

At FSE 2004, a new stream cipher VMPC [8] was proposed by Bartosz Zoltak, which appeared to be a modification of the RC4 stream cipher. In cryptanalysis, a linear distinguishing attack is one of the most common attacks on stream ciphers. In the paper [8] it was claimed that VMPC is designed especially to resist distinguishing attacks.

At the same conference, FSE 2004, another cipher RC4A [9] was proposed by Souradyuti Paul and Bart Preneel. This cipher is another modification of RC4.

In our paper we point out a general theoretical weakness of such ciphers, which, in some cases, can tell us without additional calculations whether a new construction is weak against distinguishing attacks. We also investigate VMPC and RC4A in particular and find two linear distinguishing attacks on them. VMPC can be distinguished from random using around 2^{54} bytes of the keystream, whereas the attack on RC4A needs only 2^{58} bytes. This is the first paper that proposes attacks on VMPC and RC4A.

This paper is organized as follows. In Section 2 we describe RC4, RC4A, and the VMPC ciphers. In Section 3 we study digraphs on an instance of VMPC, and then we demonstrate a theoretical weakness of the RC4 family of stream ciphers in general. We propose our distinguishers for both VMPC and RC4A in Sections 4 and 5. Finally, we summarize the results and make our conclusions in Section 6.

1.1 Notations

The algorithms VMPC, RC4A and RC4 are byte oriented stream ciphers. For notation purposes we consider VMPC-n, RC4A-n, and RC4-n to be n-bit oriented ciphers, i.e., the originals are when $n = 8$. Therefore, in the design of these ciphers, + means addition modulo 2^n. For simplicity in formulas, let q be the size of permuters used in these ciphers, i.e.

$$q = 2^n. \qquad (1)$$

The ciphers have an internal state consisting of one or two permuters of length q, and a few iterators. The idea of these designs is derived from the RC4 stream cipher, therefore, we call ciphers with a structure similar to RC4 as *the RC4 family of stream ciphers*. We denote by O_t the n-bit output symbol at time t. When a permuter $P[\cdot]$ is applied k times, e.g., $P[P[\ldots P[x]\ldots]]$, then, for simplicity, we sometimes denote it as $P^k[x]$.

1.2 Preliminaries: A Linear Distinguishing Attack

In a *linear distinguishing attack* one can observe a keystream of some length (known plaintext attack), and give an answer: whether the stream comes from the considered cipher, or from a truly random source. Distinguishers are usually based on statistical analysis of the given stream. At any point t in the stream we observe b linear combinations, the joint value of which is called a *sample* at time t. If the stream is completely random, then the sample is from the *random distribution* denoted as D_{Random}. If the stream is the keystream from the considered cipher, then the sample is from the *cipher distribution* denoted as D_{Cipher}.

To give an answer whether the given stream is from D_{Random} or D_{Cipher} one has to collect N samples from the stream at different points. These N samples form an *empirical distribution*, named also *type* and denoted as D_{Type}. If the distance from D_{Type} to D_{Cipher} is less than the distance to D_{Random}, then we conclude that the stream is from the cipher, otherwise it is decided to be from a random source.

The distance between two distributions is given as

$$\delta = |D_{\text{A}} - D_{\text{B}}| = \sum_{\text{all } x} |\Pr\{x|x \in D_{\text{A}}\} - \Pr\{x|x \in D_{\text{B}}\}|. \tag{2}$$

From statistical analysis the following fact is well known. The closer the distributions D_{Cipher} and D_{Random} are to each other, the larger the number of samples N should be, in order to distinguish with a negligible probability of error. The distance $\epsilon = |D_{\text{Cipher}} - D_{\text{Random}}|$ is then called the *bias*. The bias and the number of required samples N, from which we form our type D_{Type}, are related by the formula $N = \frac{\text{const}}{\epsilon^2}$, where the constant influences on the probability of the decision error. For more details we refer to [10]. However, the following relation is enough to have a rather negligible probability of error, and we use this formula in our paper.

$$N = \frac{1}{\epsilon^2} \tag{3}$$

1.3 Cryptanalysis Assumptions

We start our analysis of the RC4 family of stream ciphers by making a few reasonable assumptions.

(1) We assume that the initialisation procedure is perfect, i.e., all internal variables (except known iterators) are from the uniform distribution. In practice this is not true, but we make this assumption as long as we do not investigate the initialisation procedures;
(2) In our distinguishers we construct a type D_{Type} by collecting samples from the given keystream. Each derived sample at time t is from some *local distribution* of the keystream. We assume that at any time the internal state of a cipher is uniformly distributed and we don't have any knowledge about it. This assumption will be used to investigate different local distributions in the next sections. In our simulations we checked that the internal state of VMPC is roughly uniformly distributed. But for RC4A the internal state is not uniformly distributed;
(3) We consider that adjacent samples are independent. In the real life it is not true, because between two consecutive samples the internal states of a cipher are dependent. It means that samples might have a strong dependency, which may influence on the resulting type D_{Type}. To reduce these dependencies we suggest to skip few samples before accept one, then the consecutive adjacent samples will be much less dependent on each other.

2 Descriptions of VMPC-n, RC4-n, and RC4A-n

The stream cipher RC4-n [1] was designed by Ron Rivest in 1987. It produces an infinite pseudo-random sequence of n-bit symbols, which is, actually, the keystream. Encryption is then performed in a typical way for stream ciphers: `Ciphertext = Plaintext` $\oplus$ `Keystream`. The structure of RC4-n is shown in Figure 1(left).

The stream cipher VMPC-n [8] was proposed at FSE 2004 by Bartosz Zoltak. This cipher is also byte oriented ($n = 8$), and is a generalised version of RC4-n. The structure of VMPC-n is shown in Figure 1(right).

The stream cipher RC4A-n[9] was proposed at FSE 2004 by Souradyuti Paul and Bart Preneel. This cipher is an attempt to hide the correlation between the internal states and the keystream. The authors suggested to introduce a second permuter in the design. The structure of RC4A-n is shown in Figure 1(bottom).

Internal variables:
i, j – integers $\in [0 \ldots q-1]$
$P[0 \ldots q-1]$ – a permuter of integers $0 \ldots q-1$
The RC4-n cipher
1. $P[\cdot]$ – are initialised with the secret key
   $i = j = 0$
2. Loop until get enough n-bit symbols
   | $i++$
   | $j+=P[b]$
   | `swap`$(P[i], P[j])$
   | `output` $\leftarrow P[P[i]+P[j]]$

Internal variables:
i, j – integers $\in [0 \ldots q-1]$
$P[0 \ldots q-1]$ – a permuter of integers $0 \ldots q-1$
The VMPC-n cipher
1. $j, P[\cdot]$ – are initialised with the secret key
   $i = 0$
2. Loop until get enough n-bit symbols
   | $j = P[j + P[i]]$
   | `output` $\leftarrow P[P[P[j]]+1]$
   | `swap`$(P[i], P[j])$
   | $i++$

Internal variables:
i, j_1, j_2 – integers $\in [0 \ldots q-1]$
$P_1[0 \ldots q-1], P_2[0 \ldots q-1]$ – two permuters of integers $0 \ldots q-1$
The RC4A-n cipher
1. $P_1[\cdot], P_2[\cdot]$ – are initialised with the secret key
   $i = j_1 = j_2 = 0$
2. Loop until get enough n-bit symbols
   | $i++$
   | $j_1+=P_1[i]$
   | `swap`$(P_1[i],\ P_1[j_1])$
   | `output` $\leftarrow P_2[P_1[i]+P_1[j_1]]$
   | $j_2+=P_2[i]$
   | `swap`$(P_2[i],\ P_2[j_2])$
   | `output` $\leftarrow P_1[P_2[i]+P_2[j_2]]$

Fig. 1. The structures of RC4-n (left), VMPC-n (right), and RC4A-n (bottom) ciphers.

3 Investigation of the RC4 Family of Stream Ciphers

In this section we approximate different *local distributions* of the accessible keystream in the RC4 family of stream ciphers, with the assumptions that were made in Section 1.3. Since in the real cipher the internal state is not from the uniform distribution, the real local distribution differs from our approximation. However, in practice we will show that this does not make our distinguishers worse.

3.1 Digraphs Approach, on the Instance of VMPC-n

In this subsection we give the idea of how a distinguisher for VMPC can be built. In the previous work [5] the cipher RC4-n was analysed. The authors suggested to observe two consecutive output symbols O_t, O_{t+1}, and the *known* variable i jointly. For RC4-5 they could calculate theoretical probabilities $\Pr\{(i, O_t = x, O_{t+1} = y)\}$, for all possible n^3 values of the triple (i, x, y) (let us denote such distribution as $D_{(i,O_t,O_{t+1})}$). But for RC4-8 they could only approximate the bias for the distribution above due to the high complexity of calculations, and show that a distinguisher needs around $2^{30.6}$ samples (the required length of the plaintext to know).

We use a similar idea to create a distinguisher for VMPC-n. For this purpose we investigate the pair (O_t, O_{t+1}) in the following scheme.

i – known value at time t
$j, P[\cdot]$ – are from a random source

1. $O_t = P[P^2[j] + 1]$
2. $\texttt{swap}(P[i], P[j])$
3. $j' = j + P[i + 1]$
4. $O_{t+1} = P[P^3[j'] + 1]$

Below we give the explicit algorithm to calculate the approximated distribution table $D_{(i,O_t,O_{t+1})}$. For each value i, in each cell of a table T we want to store an integer number $T[i, x, y]$ of possible pairs $(i, P[\cdot])$, which cause the corresponding output pair $(O_t = x, O_{t+1} = y)$. It means, that the probability of any triple (i, O_t, O_{t+1}) is then calculated as:

$$\Pr\{(i, O_t = x, O_{t+1} = y)\} = \frac{T[i, x, y]}{q \cdot q!}. \tag{4}$$

Algorithm 1: *Recursive construction of the approximated distribution table* $D_{(i,O_t,O_{t+1})}$
Prepare the permuter: $P[i] = \infty$ at all positions, i.e., all cells of the permuter are undefined. In the algorithm the operation *define* $P[i]$ means that for the cell i in the permuter $P[\cdot]$ we need to try all possible values $0 \ldots (q - 1)$. Note, we cannot select a value which has been already used in another cell of the permuter in a previous step. Before making a step back by the recursion, restore the value $P[i] = \infty$. In the case when the cell $P[i]$ was already defined (is not ∞) due to previous steps, then we just go to the next step directly.
Do the following steps recursively:
· for all $i = 0 \ldots q - 1$;
· for all $j = 0 \ldots q - 1$;
· define $P[j]$;
· define $P^2[j]$;
· define $P[P^2[j] + 1] \Rightarrow$ remember $x = P[P^2[j] + 1]$;
· define $P[i]$;
· swap$(P[i], P[j])$;
· define $P[i + 1] \Rightarrow$ calculate $j' = j + P[i + 1]$;
· define $P[j']$, then $P^2[j']$, then $P^3[j']$;
· define $P[P^3[j] + 1] \Rightarrow$ remember $y = P[P^3[j] + 1]$;
· $T[i, x, y] += (q - r)!$, where r is the actual number of currently defined cells in the permuter $P[\cdot]$.

As we can see from the algorithm, its complexity is $O(2^{11n})$ [1]. In our simulations we could manage to calculate the approximation of $D_{(i,O_t,O_{t+1})}$ only for the reduced version VMPC-4. The bias of such table appeared to be around $\epsilon \approx 2^{-8.7}$. It means that we can distinguish VMPC-4 from random having plaintext of length around 2^{18} 4-bits symbols. For notation purposes, let $D^{\mathrm{VMPC}-n}_{(i,O_t,O_{t+1})}$ be the distribution $D_{(i,O_t,O_{t+1})}$ for VMPC-n, and similar for $D^{\mathrm{RC4}-n}_{(i,O_t,O_{t+1})}$.

The calculation of a similar distribution table for VMPC-8 meets computational difficulties, as well as for RC4-8 in [5]. One of the ideas in [5] was to approximate the biases from small n's to a larger n, but we decided not to go this way. Instead, in the next sections we will present only precise theoretical results on VMPC-8, and on the RC4 family of stream ciphers in general.

[1] The complexity to construct such a table with a similar algorithm for RC4-n is $O(2^{6n})$ [5].

3.2 Theoretical Weakness of the RC4 Family of Stream Ciphers

The recursive Algorithm 1 is trivial and slow, but we use it to show the further theoretical results. We prove that the approximated distribution table $D_{(i,O_t,O_{t+1})}$ cannot be the uniform distribution when n is larger than some threshold n_0. Moreover, we prove that *each* probability of the approximated distribution $D_{(i,O_t,O_{t+1})}$ differs from the corresponding probability in the case of a random source. In other words, the approximated distribution $D_{(i,O_t,O_{t+1})}$ is biased and we find the lower bound of the bias $\epsilon_{\min}$.

Theorem 1. *For VMPC-n, where $n \geq 8$, under the assumptions made in Section 1.3, the following hold.*

1. *Each probability* $\Pr\{(i, O_t = x, O_{t+1} = y)\} \neq 1/q^3$ *(in a random case it should be $1/q^3$).*
2. *The bias* $|D_{\text{Random}} - D^{\text{VMPC}-n}_{(i,O_t,O_{t+1})}|$ *is bounded by*

$$q^{-8n} \leq \epsilon_{\min} = \frac{|\delta_{\min}| \cdot q \cdot (q-9)!}{q!} \leq \epsilon = |D_{\text{Random}} - D^{\text{VMPC}-n}_{(i,O_t,O_{t+1})}|, \tag{5}$$

where $|\delta_{\min}|$ is the minimum value, such that

$$(q-1)(q-2) \cdot \ldots \cdot (q-8) + \delta_{\min} \equiv 0 \pmod{q}.$$

3. *For VMPC-8, we have* $\epsilon_{\min} \approx 2^{-56.8}$.

Proof:
1) Consider Algorithm 1. In the last step the value of r, the number of currently placed positions in the permuter, can be at most 9. It means that when the algorithm is finished, each cell in $D^{\text{VMPC}-n}_{(i,O_t,O_{t+1})}$ can be written in the form $k \cdot (q-9)!$, for some integer number k.
On the other hand, for a truly random sequence, the probability must be $\Pr\{(i, O_t, O_{t+1})\} = 1/q^3$. *From (4) it follows that $\frac{k \cdot (q-9)!}{q \cdot q!}$ must be equal to $\frac{1}{q^3}$, i.e.,*

$$k \quad \text{must be equal to} \quad \frac{q \cdot (q-1) \cdot \ldots \cdot (q-8)}{q^2}. \tag{6}$$

Since k is an integer, then q must divide $(q-1) \cdot \ldots \cdot (q-8)$. It is easy to show that starting from $n \geq 8$ this is not true.
2) We now try to choose k such that $\Pr\{(i, O_t, O_{t+1})\}$ is as close to $1/q^3$ as possible. Let $|\delta_{\min}|$ be the smallest value such that $(q-1) \cdot \ldots \cdot (q-8) + \delta_{\min}$ is divisible by q. Then $\Pr\{(i, O_t, O_{t+1})\} = \frac{1}{q^3} \pm \frac{q \cdot |\delta_{\min}| \cdot (q-9)!}{q^3 \cdot q!}$. *The minimum value of $|D_{\text{Random}} - D^{\text{VMPC}-n}_{(i,O_t,O_{t+1})}|$ is then derived as*

$$\epsilon_{\min} = q^3 \cdot \frac{q \cdot |\delta_{\min}| \cdot (q-9)!}{q^3 \cdot q!} = \frac{|\delta_{\min}| \cdot q \cdot (q-9)!}{q!}. \tag{7}$$

3) for VMPC-8, the minimum $\delta_{\min}$ is 128. Hence, the lower bound for the bias is $\epsilon_{\min} \approx 2^{-56.8}$.
$\square$

For RC4-n a maximum of 6 positions can be fixed, if we use a similar algorithm. Hence, all cells of the distribution table $D^{\text{RC4}-n}_{(i,O_t,O_{t+1})}$ can be written in the form $k \cdot (q-6)!$. By similar arguments as above, we conclude:

Corollary 1. *For RC4-n, $n \geq 4$, under the assumptions made in Section 1.3, the following hold.*

1. *Each probability in* $D^{\text{RC4}-n}_{(i,O_t,O_{t+1})} \neq 1/q^3$;
2. *The minimum value* $|D_{\text{Random}} - D^{\text{RC4}-n}_{(i,O_t,O_{t+1})}|$ *is bounded by*

$$q^{-5n} \leq \epsilon_{\min} = \frac{|\delta_{\min}| \cdot q \cdot (q-6)!}{q!} \leq \epsilon = |D_{\text{Random}} - D^{\text{RC4}-n}_{(i,O_t,O_{t+1})}|, \tag{8}$$

where $|\delta_{\min}|$ is the minimum value, such that

$$(q-1)(q-2) \cdot \ldots \cdot (q-5) + \delta_{\min} \equiv 0 \pmod{q};$$

3. For RC4-n, $n = 4, \ldots, 8$, we have the following lower bounds.

	n=4	n=5	n=6	n=7	n=8
$\delta_{\min}$	$+8$	-8	-8	-8	-120
$\epsilon_{\min}$	$2^{-15.46}$	$2^{-21.28}$	$2^{-26.65}$	$2^{-31.83}$	$2^{-33.01}$

$\square$

The above theorem shows us the way how one can think when designing a new cipher from the RC4 family of stream ciphers to avoid these weaknesses. For the case of VMPC-8, for instance, we can say that the structure seem to be weak in advance, without deep additional investigations of the cipher.

On the contrary, for RC4A-8 our theorem gave us a very small lower bound, so that a hypothetical distinguisher would be slower than an exhaustive search. It means that this cipher would resist distinguishing attacks better than, for example, VMPC-8 or RC4-8. Note, these conclusions were made with the assumptions from Section 1.3. However, in the next sections we investigate digraphs for both ciphers VMPC-n and RC4A-n in detail.

4 Our Distinguisher for VMPC-n

4.1 What the probability that $O_t = O_{t+1} = 0$, when $i = 0$ and $j = 1$, should be?

If VMPC-n would be a truly random generator, then the answer to the question of this section would be $1/q^2$, because when i and j are fixed, then $\Pr\{O_t = 0, O_{t+1} = 0 | i = 0, j = 1, \text{Random source}\} = 1/q^2$. In the case of VMPC-$n$ this is not true. The only case when the desired outputs can be produced is depicted in Figure 2 (left). All the other permuters will lead to other pairs of outputs $(O_t, O_{t+1}) \neq (0, 0)$. As an example, in Figure 2 (right) we show one of the cases, which contradicts the desired conditions.

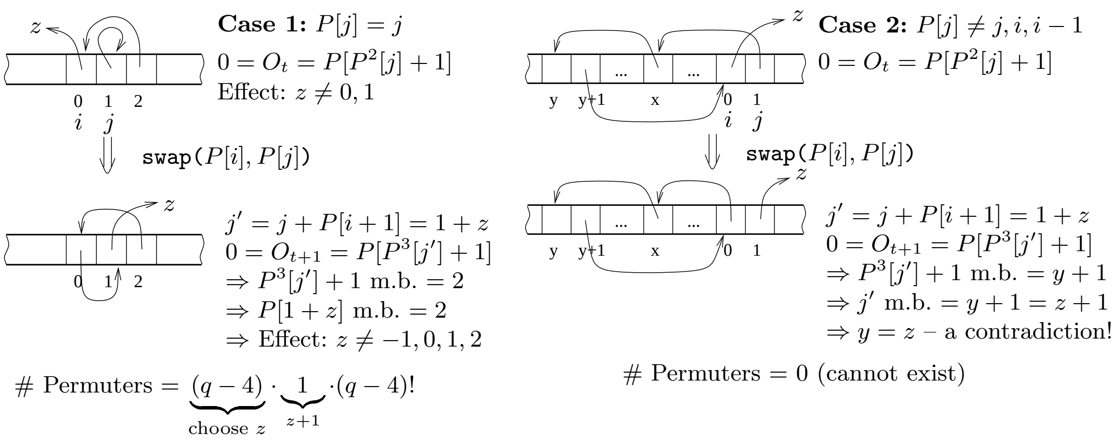

Fig. 2. Condition: $O_t = O_{t+1} = 0$, $i = 0$, $j = 1$. The only case when the condition is satisfied (left), and one of the cases when it is not (right).

By this small investigation we have shown that

$$\Pr\{O_t = O_{t+1} = 0 | i = 0, j = 1, \text{VMPC-}n\} = \frac{(q-4)(q-4)!}{q!} = \frac{q-4}{q(q-1)(q-2)(q-3)} \approx 1/q^3$$

is significantly smaller compared to $\Pr\{O_t = O_{t+1} = 0 | i = 0, j = 1, \text{Random source}\} = 1/q^2$. If we now assume that for the other values of j the probability $\Pr\{O_t = O_{t+1} = 0 | i = 0, j \neq 1, \text{VMPC-}n\} \approx 1/q^2$ – like in a random case, then we can derive that $\Pr\{O_t = O_{t+1} = 0 | i = 0\}$ is equal to $(\frac{1}{q} \cdot \frac{1}{q^3} + \frac{q-1}{q} \cdot \frac{1}{q^2})$ (in a random case it should be $1/q^2$). Then, we have a bias $\epsilon \approx 2^{-3n}$, and our hypothetical distinguisher needs to observe the event $O_t = O_{t+1} = i = 0$ from around 2^{6n} samples (2^{7n} bytes of the keystream). It means that VMPC-8 can be distinguished from random having around 2^{56} bytes of keystream. But this estimated bias is still too rough for VMPC-8, and in the next section we show how to compute the exact probability $\Pr\{O_t = O_{t+1} = 0|i = 0\}$ for VMPC-8.

4.2 Calculating $\Pr\{O_t = O_{t+1} = 0|i = 0\}$, when j and $P[\cdot]$ are random.

We could calculate the complete distribution table $D_{(i,O_t=x,O_{t+1}=y)}$ for VMPC-4, and the bias appeared to be $\epsilon \approx 2^{-8.7}$. Unfortunately, we could not apply Algorithm 1 for VMPC-8, because the complexity is 2^{88} – infeasible for a common PC. Instead, we propose to consider only two events $\{O_t = O_{t+1} = 0\}$ and its complement for $i = 0$. We distinguish between the following two binary distributions:

$$D_{\mathrm{VMPC}-n} = \left.\begin{pmatrix} \Pr\{O_t = O_{t+1} = 0\} \\ 1 - \Pr\{O_t = O_{t+1} = 0\} \end{pmatrix}\right|_{i=0}. \quad \text{and} \quad D_{\mathrm{Random}} = \left.\begin{pmatrix} 1/q^2 \\ 1 - 1/q^2 \end{pmatrix}\right|_{i=0} \tag{9}$$

Here we give the algorithm to calculate the probability $\Pr\{O_t = O_{t+1} = 0|i = 0\}$.

Algorithm 2: *Recursive computation of the probability* $\Pr\{O_t = O_{t+1} = 0|i = 0\}$
We use the same operation *define* $P[i]$ as in Algorithm 1.
Do the following steps recursively:
- for all $j = 0 \dots q - 1$;
- define $P[j]$, then $P^2[j]$;
- Since $O_t = 0$, then *fix* the position $P[P^2[j]+1] = 0$. If this position is already defined ($\neq \infty$), and the value is not 0, or pointer to 0 is already used, then track back by the recursion;
- define $P[i = 0]$;
- swap($P[i]$, $P[j]$);
- set $P[i + 1] = P[1]$, if possible, otherwise return by recursion;
- calculate $j' = j + P[i + 1]$ which is the same as $j + P[1]$;
- Since $O_{t+1} = 0$, and 0 is already placed in the permuter $P[\cdot]$, then we know the value $P^3[j'] + 1$, hence, we also know the value $P^3[j'] = c$. We can calculate the number of permuters of size q, where $P^3[j'] = c$, and r positions are fixed from the previous steps, by the subalgorithm of complexity $O(q)$, given in Appendix A.

The Algorithm 2 has complexity $O(2^{5n})$, i.e., to calculate $\Pr\{O_t = O_{t+1} = 0|i = 0\}$ for VMPC-8 we need to make only 2^{40} operations. After simulation we got the following result.

Theorem 2. *For VMPC-8, under the assumptions made in Section 1.3,*

$$\Pr\{O_t = O_{t+1} = 0|i = 0\} = \frac{15938227062862998000}{256 \cdot 4096374767995023500000},$$

and the bias is $\epsilon \approx 2 \cdot 2^{-23.98322} \approx 2^{-23}$. *I.e., we can distinguish VMPC-8 from random having around* 2^{46} *samples, or* $2^8 \cdot 2^{46} = 2^{54}$ *bytes of the keystream, when the two events from the equation* (9) *are considered. The cipher and random distributions are the following,*

$$D_{\mathrm{Random}} = \left.\begin{pmatrix} 2^{-16} \\ 1 - 2^{-16} \end{pmatrix}\right|_{i=0}, \quad D_{\mathrm{VMPC}-8} = \left.\begin{pmatrix} 2^{-16} - 2^{-23.98} \\ 1 - 2^{-16} + 2^{-23.98} \end{pmatrix}\right|_{i=0}. \tag{10}$$

□

4.3 Simulations of the Attack on VMPC-n

Our theoretical distinguisher from the previous subsection is based on a few assumptions from Section 1.3. First of all, by simulations we have checked the distribution of the internal state of VMPC-n for different values of n, and we did not find any noticeable anomalies. From this we conclude that the internal state in real is distributed close to the uniform distribution, and our theoretical distinguisher should work. Secondly, we can argue that the samples are quite independent. It happens because each sample is connected to the known variable i, and the distance between two samples (for a fixed i) is q rounds of the internal loop.

Theorem 2 says that the complexity of the attack on VMPC-8 is $O(2^{54})$, and, due to such a high complexity, we could not perform simulations of our attack on this cipher. Instead, we could perform simulations on the reduced version VMPC-4, and show the attack in practice.

VMPC-4 has one permuter of size 16, and the internal indices i and j are taken modulo 16. In our simulations we made $N = 2^{34}$ iterations and from 2^{34} received samples we have constructed the type (empirical distribution) with probabilities $\Pr\{O_t = x, O_{t+1} = y|i\}$. Below we show this table (type) partly.

$N = 2^{34}$	i=0				i=1				...
$x \Rightarrow$	0	1	2	...	0	1	2	...	
	To get the probability of the event $(O_t = x, O_{t+1} = y)\|i$ the corresponding cell should be divided by 16^2. In the case of a random source each such event has the probability $1/16^2$.								
$y \Rightarrow 0$	**0.92474**	0.99866	1.00432		0.99287	0.99086	0.99890		
1	1.00085	0.98815	1.01204		0.99309	0.99656	0.99068		
2	1.00519	1.00569	1.00343	...	0.99496	1.06880	1.06524	...	...
3	1.00631	0.99999	0.99562		1.00080	0.99260	0.99767		
⋮	⋮	⋮	⋮		⋮	⋮	⋮		
15	0.99744	0.98926	1.00845		1.00052	0.99124	0.99495		

This table represents the type D_{Type} and we can see that many probabilities are far away from $1/16^2$, and the most biased probability is in the cell $(0, 0)$, which corresponds to $\Pr\{O_t = O_{t+1} = 0|i = 0\} = \frac{0.924744}{16^2}$. When the type (the table with probabilities) is derived, one can analyze two possible distinguishers for VMPC-4.

(1) In the first scenario we consider the whole distribution table, i.e., we consider all events of the form $(i, O_t = x, O_{t+1} = y)$. The probability of each event in this case is $1/16^3$. I.e., each cell of the table (type) should be divided by $1/16^3$.
The bias of the received type is $\epsilon_0 = 2^{-8.679648}$, which is close to the theoretical value calculated in the previous section $\epsilon = 2^{-8.7}$. However, we could not calculate a theoretical bias for VMPC-8, therefore, we consider the second scenario;

(2) In this scenario we observe only two events $\{O_t = O_{t+1} = 0|i = 0, \textit{the others}\}$ – as in (9). As we have mentioned, the probability of the event $(O_t = O_{t+1} = 0)|i = 0$ is much lower than the corresponding probability in the case of a random source. In this example, the received bias appears to be $\epsilon_0 = 2 \cdot \frac{1.0-0.924744}{16^2} \approx 2^{-10.73205}$, which, again, is close to the theoretical value $\epsilon = 2^{-10.755716}$ (calculated in a similar way as for VMPC-8 in Theorem 2). For other values of n the simulation results are presented in the following table.

	n=3	n=4	n=5	n=6	n=7	n=8
Theoretical bias, ϵ	$2^{-7.551}$	$2^{-10.756}$	$2^{-13.871}$	$2^{-16.934}$	$2^{-19.967}$	$2^{-22.98}$
	Simulations of the Attack on VMPC-n					
Number of rounds made, N_0	2^{30}	2^{30}	2^{30}	2^{35}	—	—
The real bias, ϵ_0	$2^{-7.558}$	$2^{-10.732}$	$2^{-13.931}$	$2^{-16.912}$	—	—

Our simulations show that the attack on VMPC-n works in practice. We have also shown that the dependency of the adjacent samples does not influence much on the type.

5 Our Distinguisher for RC4A-n

5.1 Building a Distinguisher

In this section we investigate the cipher RC4A-n (see Figure 1(bottom)), and propose our distinguisher for RC4A-8. We again idealize the situation by the preliminary assumptions from Section 1.3, i.e., at any time t the values $j_1, j_2, P_1[\cdot]$, and $P_2[\cdot]$ are considered from the uniform distribution, and unknown for us. We would like to investigate the following scheme.

i – known value at time t-even
$j_1, j_2, P_1[\cdot], P_2[\cdot]$ – are from a random source
1. $O_t = P_2[P_1[i] + P_1[j_1]]$
2. `swap(`$P_2[i], P_2[j_2]$`)`
3. $O_{t+1} = \ldots$
4. $O_{t+2} = P_2[P_1[i+1] + P_1[j_1 + P_1[i+1]]]$

For cryptanalysis of RC4A-n, we use ideas as before. Our methodology of finding anomalies for both VMPC-n and RC4A-n was just to consider the distribution tables like $D_{(i,O_t,O_{t+2})}$ for small values of n, using an Algorithm 1-like procedure. If some anomaly is found then we concentrate on them in particular for larger values of n, and try to understand why anomalies exist.

For RC4A-n we have noticed that $\Pr\{O_t = O_{t+2} |\ i \text{ is even}\} \neq 1/q$, i.e., does not correspond to the random distribution, whereas the other probabilities $\Pr\{O_t \neq O_{t+2} |\ i \text{ is even}\}$ are equal to each other, but not equal to $1/q$. From the other hand, all probabilities $\Pr\{O_t = O_{t+2} |\ i \text{ is odd}\} = 1/q$ – correspond to the random distribution. So, our target is to calculate the probabilities $\Pr\{O_t = O_{t+2} |\ i \text{ is even}\}$ for RC4A-8. We have used a similar idea as in the Algorithm 2, but much simpler. Our optimized search algorithm to find all such probabilities has complexity $O(2^{6n})$. The result of this work is the following.

Theorem 3. *For RC4A-n, under the assumptions made in Section 1.3, consider the following vector of events, and its random distribution,*

$$\text{Events} = \begin{pmatrix} O_t = O_{t+2}|i = 0 \\ O_t = O_{t+2}|i = 2 \\ \vdots \\ O_t = O_{t+2}|i = q-2 \\ \text{other cases} \end{pmatrix}, \quad D_{\text{Random}} = \begin{pmatrix} 1/q^2 \\ 1/q^2 \\ \vdots \\ 1/q^2 \\ 1 - 1/(2q) \end{pmatrix}. \tag{11}$$

For RC4A-8, the bias $D_{\text{RC4A}-8}$ is $\epsilon \approx 2 \cdot 2^{-30.05}$. Hence, our distinguisher needs around 2^{58} bytes of the keystream. □

5.2 Checking the Assumptions

By simulations we found that the internal state of RC4A-n is not close to the uniform distribution. We could clearly see these anomalies running simulations many times for different n each time sampling from at least $N = 2^{30}$ rounds of the loop. To begin counting anomalies, we would like to note that the internal variables $j_1, P_1[\cdot]$ are updated independently from $j_2, P_2[\cdot]$ as follows.

One-Round-Update for $j_*, P_*[\cdot]$, where $*$ is 1 or 2
1. $i++$;
2. $j_* += P_*[i]$
3. `swap(`$P_*[i], P_*[j_*]$`)`

It means that all anomalies found for $j_1, P_1[\cdot]$ are true for $j_2, P_2[\cdot]$ as well.

We found an event for which the probability is far from the probability of this event in the case of a random source. In particular, $\Pr\{j_1 = i+1\} \approx \frac{q-1}{q^2}$, when in the random case it should be $1/q$. Other probabilities are $\Pr\{j_1|i, j_1 \neq i+1\} \approx \frac{q^2-q+1}{q^2(q-1)}$. For example, for RC4A-4, it appeared that $\Pr\{j_1 = i+1\} \approx 0.05859375$, and the others are $\Pr\{j_1|i, j_1 \neq i\} \approx 0.06276042$ – the difference is noticeable. Some other less notable non-uniformities in the internal state also were found.

5.3 Simulations of the Attack on RC4A-n

Despite finding the non-uniformity of the internal state of RC4A-n we make a set of simulations to see how our distinguisher behaves itself. We will consider the attack scenario as in Theorem 3.

	n=3	n=4	n=5	n=6	n=7	n=8
Theoretical bias, ϵ	$2^{-10.014}$	$2^{-14.005}$	$2^{-18.001}$	$2^{-22.00}$	$2^{-26.00}$	$2^{-29.05}$
	Simulations of the Attack on RC4A-n					
Number of rounds made, N_0	2^{30}	2^{30}	2^{34}	2^{40}	2^{40}	—
The real bias, ϵ_0	$2^{-8.9181}$	$2^{-12.2703}$	$2^{-15.073}$	$2^{-18.042}$	$2^{-20.025}$	—

Note that the number of actual samples N_0 in our simulations is larger than $1/\epsilon_0^2$. From (3) it means that we have distinguished the cipher with a very small probability of error, and the real theoretical bias without pre-assumptions should be close to what we get in our simulations. From the table above we see that the bias in practice (when the internal state is not from the unoform distribution) is larger than the approximated value of the bias (the uniformly distributed internal state), for $n = 3, \ldots, 7$. The same behaviour of the distinguisher we expect for $n = 8$ as well. Since we could not perform simulations for $n = 8$, we decided to leave theoretical bias as the lower bound of the attack, i.e., $\epsilon = 2^{-29.05}$ for $n = 8$, the complexity is $O(2^{58})$. However, in the real life we expect this bias to be even larger, and complexity of the attack lower.

6 Results and Conclusions

In this paper we have shown some theoretical weaknesses of the RC4 family of stream ciphers. We have also investigated recently suggested stream ciphers VMPC-n and RC4A-n, and found linear distinguishing attacks on them. They are regarded as academic attacks which show weak places in these ciphers. The summarizing table of our results is below:

Cipher	Theoretical Lower Bound for ϵ, $n=8$	Our Distinguishers Complexity (# of symbols)					
		$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$
RC4-n (1987)	2^{-33} (Corr.1)	—	—	—	—	—	$2^{30.6}$ (from [5])
VMPC-n (2004)	$2^{-56.8}$ (Thr.1)	2^{23}	* 2^{29}	2^{35}	2^{41}	2^{48}	$\mathbf{2^{54}}$
RC4A-n (2004)	—	2^{18}	2^{28}	2^{36}	2^{44}	2^{52}	$\mathbf{2^{58}}$

The distinguisher for VMPC-8 that we propose is the following [2]:

Distinguisher for VMPC-8:

1. Observe $N = 2^{54}$ output bytes. Calculate the number L of occurences such that $a = O_t = O_{t+1} = 0$.
2. Calculate two distances:
   $\lambda_{\text{Random}} = |2^{-16} - 2^8 \cdot L/N|$
   $\lambda_{\text{VMPC}} = |(2^{-16} - 2^{-23.98322}) - 2^8 \cdot L/N|$
3. If $\lambda_{\text{Random}} > \lambda_{\text{VMPC}}$ then **keystream of VMPC-8**, else **a random sequence**.

[2] The distinguisher for RC4A-8 is in a similar fashion as for VMPC-8.

* In the first scenario from Subsection 4.3 the attack complexity for VMPC-4 is $O(2^{18})$.

If the internal state of a cipher from the RC4 family is uniformly distributed, then, based on our discussions in Section 3, we conclude that such ciphers are not very secure. When the internal state is non-uniformly distributed then the real bias would more likely be larger rather than smaller, and the complexity of the attack would be lower, in most cases. That effect we could observe on the example of RC4A-n. It seems that the security level of such constructions depends more on the degree of the recursive relations between output symbols and internal states, rather than on the length of the permuter(s).

One of the solutions to protect against of such distinguishing attacks is to increase the number of accesses to the permuter(s) in the loop. This solution will increase the relation complexity between adjacent outputs. Another solution is to discard some output symbols before to accept one. Unfortunately, both the suggestions significantly decrease the speed of these ciphers – the main purpose of such designs (speed) is then destroyed.

Acknowledgements

We thank Willi Meier for his useful suggestions on this research direction that made this paper possible. We also thank Thomas Johansson and anonymous reviewers for their editing advises and critical comments.

References

1. N. Smart. Cryptography: An Introduction, 2003.
2. B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, New York, 2nd edition, 1996.
3. J.D. Golić. Linear statistical weakness of alleged RC4 keystream generator. In W. Fumy, editor, *Advances in Cryptology—EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 226–238. Springer-Verlag, 1997.
4. L.R. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege. Analysis methods for (alleged) RC4. In K. Ohta and D. Pei, editors, *Advances in Cryptology—ASIACRYPT'98*, volume 1998 of *Lecture Notes in Computer Science*, pages 327–341. Springer-Verlag, 1998.
5. S. R. Fluhrer and D. A. McGrew. Statistical analysis of the alleged RC4 keystream generator. In B. Schneier, editor, *Fast Software Encryption 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 19–30. Springer-Verlag, 2000.
6. I. Mantin and A. Shamir. Practical attack on broadcast RC4. In M. Matsui, editor, *Fast Software Encryption 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer-Verlag, 2001.
7. S. Paul and B. Preneel. Analysis of non-fortuitous predictive states of the RC4 keystream generator. In T. Johansson and S. Maitra, editors, *Progress in Cryptology—INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 52–67. Springer-Verlag, 2003.
8. B. Zoltak. VMPC one-way function and stream cipher. In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 210–225. Springer-Verlag, 2004.
9. S. Paul and B. Preneel. A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 245–259. Springer-Verlag, 2004.
10. D. Coppersmith, S. Halevi, and C.S. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.

[0] The work described in this paper has been supported in part by Grant VR 621-2001-2149, in part by the Graduate School in Personal Computing and Communication PCC++, and in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

Appendix A: Subalgorithm for Algorithm 2

Problem statement: We are given a permuter template of size q, where r positions are already placed, whereas the rest are undefined. We want to calculate the number of permuters satisfying the given template, such that $P^3[j'] = c$, where j' and c are some known positions in the permuter.

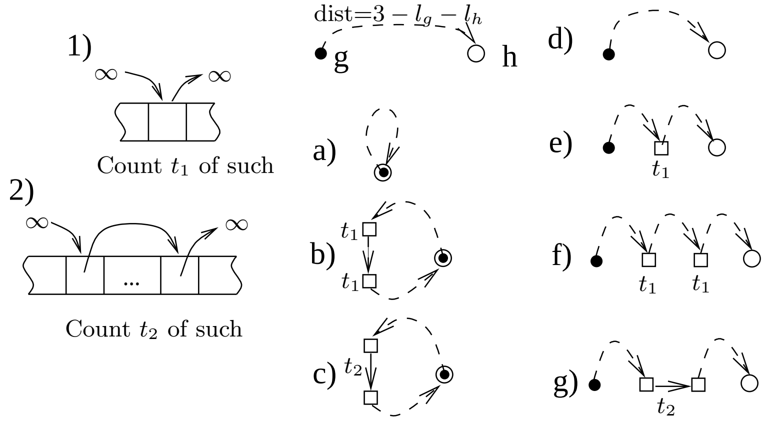

Fig. 3. Possibilities to connect g and h, used in subalgorithm.

Sub-Algorithm:[a]

1. Go forward by the path $j' \to P[j'] \to P^2[j'] \to P^3[j']$, as much as possible, but not more then 3 steps. Let g be the point in this path where we have stopped, and l_g be the number of steps we made (from 0 to 3).
2. Go backward by the path $c \to P^{-1}[c] \to P^{-2}[c] \to P^{-3}[c]$, as much as possible, but not more then 3 steps. Let h be the point in the path where we have stoped, and l_h be the number of steps we made (from 0 to 3).
3. if ($l_g = 3$ and $g \neq c$) or ($l_h = 3$ and $h \neq j'$) then return 0;
   if ($l_g = 3$ and $g = c$) or ($l_h = 3$ and $h = j'$) then return $(q-r)!$;
   if ($l_g + l_h \geq 3$) return 0;
4. Count the number t_1 of positions $x \neq g, h$ in the permuter $P[\cdot]$ for which $P[x] = P^{-1}[x] = \infty$ (see Fig. 3(1)).
   Count the number t_2 of positions $x \neq g, h$, for which $P[x] \neq \infty, g, h$, and $P^{-1}[x] = P^2[x] = \infty$ (see Fig. 3(2)).
5. Now there could be 7 possibilities to connect positions g and h, and they are depicted in Figure 3(a–g):
   a) $g = h$, $l_g + l_h = 0$ $\Rightarrow$ add $(q-r-1)!$ combinations;
   b) $g = h$, $l_g + l_h = 0, t_1 \geq 2$ $\Rightarrow$ add $t_1(t_1-1)(q-r-3)!$ combinations;
   c) $g = h$, $l_g + l_h = 0$ $\Rightarrow$ add $t_2(q-r-2)!$ combinations;
   d) $g \neq h$, $l_g + l_h = 2$ $\Rightarrow$ add $(q-r-1)!$ combinations;
   e) $g \neq h$, $l_g + l_h = 1$ $\Rightarrow$ add $t_1(q-r-2)!$ combinations;
   f) $g \neq h$, $l_g + l_h = 0, t_1 \geq 2$ $\Rightarrow$ add $t_1(t_1-1)(q-r-3)!$ combinations;
   g) $g \neq h$, $l_g + l_h = 0$ $\Rightarrow$ add $t_2(q-r-2)!$ combinations;

[a] The complexity of the subalgorithm is $O(q)$