The Most Efficient Distinguishing Attack on VMPC and RC4A

Yukiyasu Tsunoo[1], Teruo Saito[2], Hiroyasu Kubo[2], Maki Shigeri[2], Tomoyasu Suzaki[2], and Takeshi Kawabata[1]

[1] NEC Corporation
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
{tsunoo@BL, t-kawabata@pb}.jp.nec.com
[2] NEC Software Hokuriku Ltd.
1, Anyoji, Hakusan, Ishikawa 920-2141, Japan
{t-saito@qh, h-kubo@ps, m-shigeri@pb, t-suzaki@pd}.jp.nec.com

Abstract. This paper reports the greater bias found in the output sequence of VMPC, a modified RC4 stream cipher proposed in 2004. Using the bias with approximately 2^{38} output bytes allows us to distinguish VMPC from truly random sequence. Distinguishing attack can also break RC4A, an algorihm based on RC4, more efficiently than any existing attacks. With about 2^{23} output bytes, distinguishing attack makes it possible to distinguish RC4A from truly random sequence.

Key words: RC4, RC4A, VMPC, pseudo-random number generator, distinguishing attack

1 Introduction

Over 2 decades, so many stream ciphers have been proposed. Most of them employ LFSR (Linear Feedback shift Register), which is easy to implement on hardware, but is slow in performance when software-implemented. In 1987, R. Rivest designed a new stream cipher, RC4, which is suitable for software-implementation [6]. In early days, the algorithm of RC4 was kept in secret. However, its source code was made public at Cypherpunks mailing list by somebody. RC4 is implemented in many applications, including TLS /SSL and WEP/WPA, and we can say that RC4 is the stream cipher in the widest use. Simultaneously, however, many ways to break RC4 have been proposed. Typically, many of them are classified as *Distinguishing Attack*, which makes use of the bias in output sequence.

In 2004, some new stream ciphers were proposed, to which resistance to the attacks aimed at RC4 are added. They are exemplified by VMPC, a stream cipher proposed by B. Zoltak [7], and RC4A, an RC4 family algorithm improved by S. Paul and B. Preneel [5]. (For detail, see section 2.3.) However, it was reported in 2005, that A. Maximov had broken both algorithms with distinguishing attack. According to his paper, we can distinguish VMPC and RC4A from truly random sequence, using about 2^{54} and 2^{58} output bytes, respectively.

We studied VMPC further, to find greater bias in its output sequence. Making use of the bias allows us to distinguish VMPC output sequence from truly random sequence, with only about 2^{38} output bytes. We also found that distinguishing attack breaks RC4A more efficiently than the attack proposed in paper [4], using about 2^{23} output bytes. These are the best results as far as we know.

Section 2 overviews the feature of RC4 family ciphers and existing attack methods aimed at them. Section 3 shows the presence of bias in VMPC output sequence and explains that making use of the bias allows the distinction of VMPC output sequence from truly random sequence. Section 4 describes that exploiting the bias in RC4A output sequence leads to more efficient cryptanalysis than any other. Section 5 concludes this paper.

2 RC4 Family and These Security

2.1 Description of RC4

RC4 stream cipher holds array S that has 256 entries, each of which holds 1 byte (= 8 bits) and 2 index-pointers, either of which holds 1 byte. It swaps 2 entries of array S, to generate a keystream. RC4 is very resistant to the attacks originally aimed at LFSR family stream cipher. It is also resistant to the attacks based on exhaustive search for secret internal state, because the cipher holds tremendous amount of memory.

The algorithm of RC4 is shown in Fig. 1. Hereafter, + means addition mod 256, and l represents the size of secret key in bytes. The algorithm of RC4 consists of roughly 2 phases: For initialization of secret internal state, KSA (Key Scheduling Algorithm) permutes 256 entries of array S, using the value of secret key K of n-bit ($40 \leq n \leq 256$). Then, PRGA (Pseudo-Random Generation Algorithm) swaps the entries of array S that are permuted by KSA, and outputs one entry of the array S as a keystream. The keystream outputs 8 bits at a time t. Encryption is performed by generating keystreams according to the size of plaintexts and applying bit-wise exclusive OR on plaintexts.

KSA(K)	PRGA(S)
Initialization:	Initialization:
For $i = 0 \dots 255$	$i = 0$
$S[i] = i$	$j = 0$
$j = 0$	Generation loop:
Scrambling:	$i = i + 1$
For $i = 0 \dots 255$	$j = j + S[i]$
$j = j + S[i] + K[i \bmod l]$	Swap($S[i]$, $S[j]$)
Swap($S[i]$, $S[j]$)	Output $z = S[S[i] + S[j]]$

Fig. 1. Algorithm of RC4

RC4 can take $n(\neq 8)$-bit as the size of processing and consequently, $N(=2^n)$ as the number of entries for the array S. Since RC4, which is used in many applications, is 8-bit oriented cipher we discusse RC4 ($n=8, N=256$) only.

2.2 Previous Attacks on RC4

Plenty of attacks on RC4 have been proposed since its algorithm was published in 1994. In this subsection, we summarize the results of such attacks of importance.

We consider the strongest attack on RC4 is the one proposed by Mantin and Shamir in 2001 [3]. They showed that 0 appears in the second output byte of RC4 at the probability that is around twice as great as expectation value. Though this attack works only where broadcast applications are in practical use, exploiting this bias and approximately 2^8 output bytes distinguishes RC4 output sequence from truly random sequence. However, this attack can be avoided by the countermeasure of dropping the first 2 bytes of keystream.

The attack proposed by Fluhrer and McGrew in 2000 [2] is also considered to be one of the strongest attacks on RC4. Making use of close correlation between consecutive output bytes of RC4, they proved that RC4 output sequence can be distinguished from truly random sequence. They also showed that RC4 distinguisher can be built, for which about $2^{30.6}$ output bytes are used. This attack is considered to be very strong, taking the fact that the countermeasure of dropping the first 2 bytes of keystream does not work on this attack into account.

Another powerful attack is the one proposed by Paul and Preneel in 2004 [5]. They unveiled the unknown bias in the first 2 bytes of RC4 output. With about 2^{24} output bytes, this bias is used to distinguish RC4 output sequence from truly random sequence. They also reported that RC4 distinguisher can be built, experimentally, in spite of the countermeasure of dropping the first 256 bytes of keystream, if about 2^{32} output bytes are available.

Each one of 3 attacks mentioned above is a distinguishing attack, which exploits the bias in output sequence. Some attacks based on Key Recovery have been also proposed. In 2001, Fluhrer et al. showed the presence of weak key in KSA of RC4 at higher probability [1]. They also proved that RC4 is completely broken, if a part of secret key is known. This attack is considered to be practical threatening, because it can be applied to the ciphers which concatenates a fixed secret key with IV (Initial Vector), including RC4 implemented in WEP, actually.

2.3 Description of VMPC and RC4A

This subsection describes the algorithms of two target ciphers of our attack, VMPC and RC4A.

VMPC is the stream cipher proposed by Zoltak in 2004 [7]. Its algorithm is shown in Fig. 2, where V represents an IV. PRGA of VMPC has more complex structure than that of RC4, to gain more resistance against the distinguishing attack applied to RC4 as well as to amend the weakness in KSA of RC4.

```
KSA(K, V)
Initialization:
  For i = 0 … 255
    S[i] = i
  j = 0
Scrambling:
  For i = 0 … 767
    i = i mod 256
    j = S[j + S[i] + K[i mod l]]
    Swap(S[i], S[j])
  For i = 0 … 767
    i = i mod 256
    j = S[j + S[i] + V[i mod l]]
    Swap(S[i], S[j])

PRGA(S)
Initialization:
  i = 0
Generation loop:
  j = S[j + S[i]]
  Output z = S[S[S[j]] + 1]
  Swap(S[i], S[j])
  i = i + 1
```

Fig. 2. Algorithm of VMPC

In the same year, Paul and Preneel proposed the attack against RC4, and RC4A, the modified RC4 to which resistance to their attack is provided. The algorithm of RC4A is shown in Fig. 3. RC4A is made through improvement on the RC4, i.e., providing 2 S arrays that are independent from each other, so that RC4 should not have bias in consecutive output byte. The algorithm of KSA for RC4A is not described in detail. In this paper, KSA for RC4 is used as that for RC4A, following the manner of paper [5]. To be more specific, in KSA of RC4, the array S_1 is initialized, using the secret key K. WK, 16 bytes of keystream, are generated from the array S_1 in PRGA of RC4. Then, the array S_2 is initialized in KSA of RC4, using WK.

2.4 Maximov's Attack

In 2005, Maximov evaluated the resistance of RC4 Family of Stream Cipher against distinguishing attack [4]. Using the method proposed by the paper [2], Maximov investigated the bias in consecutive bytes output from RC4 cipher theoretically, to indicate the lower bound of security of RC4 and VMPC. His work revealed that RC4 and VMPC have no resistance to distinguishing attack, because of their structure. According to the paper [4], use of around 2^{54} output bytes distinguishes VMPC from truly random sequence.

No problem was found in the security of RC4A against the attack which exploits the correlation between consecutive output bytes. However, strong correlation was found between keystreams output at every second times, and it was shown that making use of the bias makes it possible to build a distinguisher. The paper [4] claims that with approximately 2^{58} output bytes, RC4A can be distinguished from truly random sequence.

```
KSA(K)
  RC4_KSA(K, S1)

  For i = 0 … l – 1
    WK[i] = RC4_PRGA(S1)

  RC4_KSA(WK, S2)
```

```
PRGA(S1, S2)
Initialization:
  i = 0
  j1 = j2 = 0
Generation loop:
  i = i + 1
  j1 = j1 + S1[i]
  Swap(S1[i], S1[j1])
  Output z = S2[S1[i] + S1[j1]]
  j2 = j2 + S2[i]
  Swap(S2[i], S2[j2])
  Output z = S1[S2[i] + S2[j2]]
```

Fig. 3. Algorithm of RC4A

We performed further investigation of the attacks described above and found a new weakness in VMPC output sequence. Exploiting the weakness allows building more effective distinguisher than ever. We also show further investigation of structural weakness of RC4A indicated in paper [4] leads to more efficient cryptanalysis of the cipher than that of Maximov.

3 The New Weakness of VMPC

This section shows a new bias found in the VMPC output sequence. It also explains that usage of the bias provides a powerful distinguisher of VMPC.

3.1 Strong Bias in the First Two Output Bytes

To explain the presence of the bias in VMPC output sequence, we make following two assumptions, providing that KSA is implemented with a given secret key and that the initialization of the array S has completed. (Time $t = 1$, when the first keystream is output)

1. $j = 0$
2. $S[A] = 0$ (A denotes the entry of $S[0]$)

Fig. 4 illustrates the status transition of array S, at the time when two assumptions described above are satisfied and then keystream is generated through PRGA.

When $t = 1$, index i is fixed to 0 by the algorithm. Index j is updated by PRGA of VMPC as follows;

$$j = S[j + S[i]] = S[0 + S[0]] = S[A] = 0$$

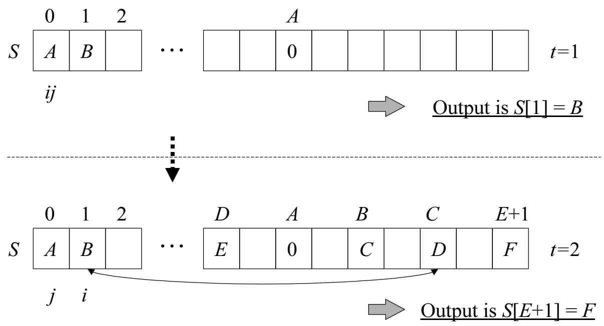

Fig. 4. PRGA of VMPC and Status Transition of Array S

Thus, j stays being 0. The keystream is output as follows;

$$z = S[S[S[j]] + 1] = S[S[A] + 1] = S[1] = B$$

Since index i = index j, entries of array S are not swapped, when $t = 1$.

When $t = 2$, index i becomes 1. Index j is updated by PRGA of VMPC as follows;

$$j = S[j + S[i]] = S[0 + S[1]] = S[B] = C$$

Then, the output keystream is obtained as follows;

$$z = S[S[S[j]] + 1] = S[S[D] + 1] = S[E + 1] = F$$

Since two equations, $i = 1$ and $j = C$ hold, the entry from array $S[1]$ is swapped with the one from array $S[C]$.

Compare 2 bytes of the output keystreams. Both B and F are entries of array S. Since entries of array S are not swapped, if $t = 1$, it is apparent that $B \neq F$, i.e. B and F take two different values from each other. Thus, if two assumptions described above are satisfied, the first 2 bytes of the keystream VMPC outputs always take different values from each other.

3.2 Probability That First Two Output Bytes Are Equal

This subsection discusses the probability that O_1 and O_2, first 2 output bytes of VMPC are equal. For truly random sequence, the probability that first 2 output bytes are equal is ideally 2^{-8}. The bias described in section 3.1 is dependent not on KSA structure, but on PRGA structure. Thus, hereafter, we assume that KSA permutes the completely randomized array S.

P_1 and P_2, the probabilities that assumption 1 and 2 described in section 3.1 hold true, respectively, are as follows; Either assumption holds true, only if 8-bit takes a certain value. Thus, if array S is randomized by KSA, probabilities P_1 and P_2 are determined by the inverse number of bit width.

$$P_1 = 2^{-8} \tag{1}$$

$$P_2 = 2^{-8} \tag{2}$$

Then, the probability that first 2 output bytes are equal is obtained as follows, using probabilities P_1 and P_2;

$$\begin{aligned} P[O_1 = O_2] = & P[O_1 = O_2 | j = 0 \cap S[A] = 0] \cdot P[j = 0 \cap S[A] = 0] \\ & + P[O_1 = O_2 | j \neq 0 \cup S[A] \neq 0] \cdot P[j \neq 0 \cup S[A] \neq 0] \\ = & \, 0 \cdot P_1 \cdot P_2 + 2^{-8} \cdot (1 - P_1 \cdot P_2) \\ = & \, 2^{-8} \cdot (1 - 2^{-16}) \end{aligned}$$

This value is significantly smaller than 2^{-8}, the ideal probability that first 2 output bytes of truly random sequence are equal.

3.3 Distinguisher Based on the Weakness

This subsection describes the amount of data needed to build the distinguisher based on the probability obtained in section 3.2. The needed amount of data is determined in paper [3].

When event e occurs, $O(\frac{1}{pq^2})$ samples are required to distinguish X (distribution of event that occurs at probability p) from Y (distribution of event that occurs at probability $p(q+1)$) at success probability that cannot be neglected.

As for our attack, event e denotes the probability that first 2 output bytes are equal. X represents the distribution of the event e concerning about truly random sequence, whereas Y denotes the distribution of the event e as far as it concerns the output bytes of VMPC. Thus, it is assumed that $p = 2^{-8}$ and $q = -2^{-16}$, and the data of amount needed for cryptanalysis becomes $O(2^{40})$. Note this value is based on the assumption that KSA of VMPC performs thoroughly random permutation. Therefore, it is a theoretical amount of data determined through using structural weakness in PRGA. Thus, theoretically, VMPC can be distinguished from truly random sequence, if approximately 2^{40} output bytes are provided.

The discussion given above is based on the assumption that KSA of VMPC performs thoroughly random permutation. However, the structure of KSA, as well as PRGA, takes the permutation of array S as a basic component. Thus, it cannot be said that KSA performs thoroughly random permutation. In fact, it is reported in paper [5] that distinguisher for RC4 can be built with smaller amount of data than theoretically needed amount of data, exploiting the feature that permutation performed by KSA of RC4 is not random.

We made an experiment to investigate the secret internal state of VMPC just after KSA operation, so that we might obtain the probability that two assumptions described in section 3.1 hold. To be specific, we determined the probability by means of giving secret keys as input randomly, and then, checking the value for secret internal state. If KSA of VMPC performs thoroughly random permutation, equations (1) and (2) theoretically support that $P_1 \cdot P_2 = 2^{-16}$. However, experimental results indicate the probability that the assumptions hold true is about 2^{-15}. This also implies that KSA of VMPC, as well as RC4, does not perform thoroughly random permutation.

Taking the results of experiment into account, $p = 2^{-8}$ and $q = -2^{-15}$, in effect, and then, the amount of data needed for cryptanalysis is $O(2^{38})$. This means providing around 2^{38} output bytes distinguishes VMPC from truly random sequence at fairly high probability. It can be said that this distinguisher is much stronger than that based on Maximov's method, which requires 2^{54} bytes.

4 More Efficient Attack of RC4A

RC4A is designed as the algorithm that is resistant to the attack proposed by paper [5]. However, it was broken by the attack proposed in paper [4], which exploits the bias in the output from PRGA of RC4A that appears at every second times. Namely, less close correlation between consecutive output bytes was focused, when improvement are made to RC4 to produce RC4A, and less attention was paid to evaluate the correlation between output bytes that are not consecutive. We show that our attack breaks RC4A more efficiently the attack proposed in paper [4], making use of the feature of RC4A, i.e. the bias in keystream output at every second times.

4.1 Bias in the Correlation Between the First and Third Output Bytes

This subsection describes the bias in the correlation between the first and third output bytes of RC4A. Basic idea of this attack based on the application of attack against RC4 shown in paper [5].

To prove the presence of bias in the correlation between the first and third ourput bytes of RC4A, we make assumption described below, providing that KSA is implemented with a given secret key, that the initialization of the array S has completed, and that time $t = 1$, when the first keystream is output.

1. $S_1[1] = 2$ (equal assumption to paper [5])

Fig. 5 illustrates the status transition of arrays S_1 and S_2, when keystream is generated through PRGA, and the assumption described above is satisfied.

Keystreams output when $t = 1$ and $t = 3$ become $S_2[A + 2]$ and $S_2[C + 2]$, respectively, because keystreams are output, looking up the entries of array S_2. Take a close look at status transition of array S_2 that is output when $t = 2$. If $t = 2$, index $i = 1$, and the index is updated by the following equation;

$$j_2 = j_2 + S_2[i] = 0 + S_2[1] = B$$

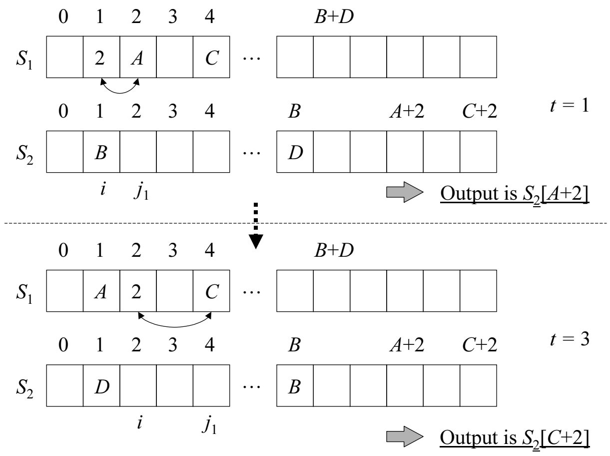

Fig. 5. PRGA of RC4A and Status Transition of Arrays S_1 and S_2

Thus, when $t = 2$, $S_2[1]$, an entry of array S_2 is swapped with another entry, $S_2[B]$. Then, pay attention to the case where $S_2[A + 2]$, keystream output at $t = 1$, is not swapped. Since A and C, entries of S_1 take two different values from each other, it is apparent that $S_2[A+2] \neq S_2[C+2]$, if $t = 2$ and if entries of $S_2[A+2]$ are not swapped. This means the first and third output bytes always take two different values from each other. Following two assumptions must be satisfied, so that this kind of correlation between entries should appear.

2. $A \neq 0xff$ (Because $A + 2 \neq 1$, A is an entry of $S_1[2]$)
3. $B \neq A + 2$ (B is an entry of $S_2[1]$)

Assumption 2 must be satisfied so that $S_2[i] = S_2[A + 2]$ may hold, whereas assumption 3 should be satisfied so that $S_2[j_2] = S_2[A+2]$ may stand. Attention must be paid to that entries of $S_2[A + 2]$ are swapped, unless all of these 3 assumptions are satisfied, and then, the first and third output bytes do not always take two different values from each other.

4.2 Probability That the First and Third Output Bytes Are Equal

This subsection discusses the probability that O_1 and O_3, the first and third output bytes of RC4A are equal. Such probability is ideally 2^{-8}, when those two bytes come from truly random sequence. The bias described in section 4.1 is dependent not on KSA structure, but on PRGA structure. Thus, hereafter, we assume that KSA makes the entries of array S permuted completely randomly.

P_1, P_2, and P_3, the probabilities that assumptions 1, 2 and 3 described in section 4.1 hold true, respectively are as follows;

$$P_1 = \frac{1}{256} \tag{3}$$

$$P_2 = \frac{255}{256} \quad (4)$$

$$P_3 = \frac{255}{256} \quad (5)$$

Then, the probability that the first and third output bytes of RC4A are equal is obtained as follows;

$$\begin{aligned}
P[O_1 = O_3] = & P[O_1 = O_3 | S_1[1] = 2 \cap A \neq 0xff \cap B \neq A + 2] \cdot \\
& P[S_1[1] = 2 \cap A \neq 0xff \cap B \neq A + 2] \\
& + P[O_1 = O_3 | S_1[1] \neq 2 \cup A = 0xff \cup B = A + 2] \cdot \\
& P[S_1[1] \neq 2 \cup A = 0xff \cup B = A + 2] \\
& = 0 \cdot P_1 \cdot P_2 \cdot P_3 + 2^{-8} \cdot (1 - P_1 \cdot P_2 \cdot P_3) \\
& = 2^{-8} \cdot (1 - \frac{255^2}{256^3}) \\
& \simeq 2^{-8} \cdot (1 - 2^{-8.01})
\end{aligned}$$

This value is significantly smaller than 2^{-8}, the ideal probability that first and third output bytes of truly random sequence are equal.

4.3 Distinguisher Based on the Weakness and Experimental Result

This subsection discusses the amount of data needed to build the distinguisher, based on the probability obtained in section 4.2. To build distinguisher of RC4A output sequence, $p = 2^{-8}$ and $q = -2^{-8.01}$ are assumed to be distribution of event, and the data of amount needed for cryptanalysis becomes $O(2^{24.02})$. Note that this value is based on the assumption that KSA of RC4A performs thoroughly random permutation. Therefore, it is a theoretical amount of data determined through using structural weakness in PRGA. Thus, theoretically, RC4A can be distinguished from truly random sequence, if approximately $2^{24.02}$ output bytes are available.

Since it cannot be said that KSA of RC4A, as well as that of VMPC performs thoroughly random permutation, we conducted an experiment to determine the probability that 3 assumptions for equality between the first and third output bytes hold simultaneously. To be specific, we determined the probability by means of giving secret keys as input randomly, and then, checking the value for secret internal state. If KSA of RC4A performs thoroughly random permutation, equations (3), (4) and (5) theoretically should support that $P_1 \cdot P_2 \cdot P_3 = \frac{255^2}{256^3} \simeq 2^{-8.01}$. However, experimental results indicates that the probability these assumptions hold true is about $2^{-7.55}$. This also implies that KSA of RC4A, as well as VMPC and RC4, does not perform thoroughly random permutation.

Taking the results of this experiment into account, the amount of data needed for cryptanalysis is $O(2^{23.1})$. This means that providing around $2^{23.1}$ output bytes distinguishes RC4A from truly random sequence. It can be said that this method is much more efficient than Maximov's.

Actually, we made an experimental attack on RC4A, taking steps described below.

1. Change the secret key randomly for 2^m times, to generate 3-byte keystream of RC4A for each secret key.
2. Count how many times the first and third output bytes of keystream are equal, for each one of 2^m keystreams.
3. If the number of data totals to 2^m, and if x, i.e. the number of times that the first and third output bytes of keystream are equal satisfies the following inequality, output sequences are regarded as non-random ones, and are rejected.

$$\mu - x > \frac{\sigma}{2}$$

where μ represents mean and σ denotes standard deviation. Thus,

$$2^{m-8} - x > \frac{\sqrt{2^{m-8} - 2^{m-16}}}{2}$$

4. Providing 100 groups that are independent from one another and consist of 2^m secret key mentioned in step 1, iterate steps 1, 2, and 3, to determine the success probability of the attack.

Following the way similar to the one described above, and based on the number of times that the first two output bytes of RC4A are equal, we also obtained the success probability of attack on RC4A, for comparison. [1] We found that when 2^{23} output bytes are used, the success probability of the attack based on the times the first and third output bytes of keystream are equal results in 53%, a significantly higher probability than that of the attack associated with the first two bytes of keystream, i.e. truly random sequence. Thus, through our experiment, it was examined that our method distinguishes RC4A from truly random sequence, using approximately 2^{23} output bytes.

5 Conclusion

In this paper, we have indicated the presence of great bias in output sequence of VMPC, a one of RC4 family stream ciphers. Exploiting this bias with about 2^{38} output bytes allows attackers to distinguish VMPC from truly random sequence. We have also shown that distinguishing attack breaks RC4A, a improved algorithm of RC4, more efficiently than any other existing attacks. Our method distinguishes RC4A from truly random sequence, using around 2^{23} output bytes, though it has not developed into the attack to determine all information on a secret key or secret internal state. However, results show that our method is the most efficient attack, as far as we know, and that it is the most powerful attack, because it offers cryptanalysis which can be performed with practical amount of computation.

[1] We assumed that there is no correlation between the first two output bytes of RC4A.

Acknowledgement.

The authors would like to thank Etsuko Tsujihara for her useful comments and suggestions.

References

1. S. Fluhrer, I. Mantin and A. Shamir: “Weaknesses in the Key Scheduling Algorithm of RC4,” *Selected Areas in Cryptography, SAC 2001*, LNCS 2259, pp.1-24, Springer-Verlag, 2001.
2. S. Fluhrer, and D. McGrew: “Statistical Analysis of the Alleged RC4 Keystream Generator,” *Fast Software Encryption, FSE 2000*, LNCS 1978, pp.19-30, Springer-Verlag, 2000.
3. I. Mantin, and A. Shamir: “A Practical Attack on Broadcast RC4,” *Fast Software Encryption, FSE 2001*, LNCS 2355, pp.152-164, Springer-Verlag, 2001.
4. A. Maximov: “Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of the RC4 Family of Stream Ciphers,” *Fast Software Encryption, FSE 2005*, available at http://www.it.lth.se/movax/
5. S. Paul, and B. Preneel: “A New Weakness in the RC4 Keystream Generator,” *Fast Software Encryption, FSE 2004*, LNCS 3017, pp.245-259, Springer-Verlag, 2004.
6. B. Schneier: “Applied Cryptography,” John Wiley & Sons, 2nd edition, 1996.
7. B. Zoltak: “VMPC One-Way Function and Stream Cipher,” *Fast Software Encryption, FSE 2004*, LNCS 3017, pp.210-225, Springer-Verlag, 2004.