NEW ATTACKS ON RC4A AND VMPC

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Mehmet Karahan
August, 2015

New Attacks on RC4A and VMPC
By Mehmet Karahan
August, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. İbrahim Körpeoğlu(Advisor)

Prof. Dr. Ali Aydın Selçuk

Assoc. Prof. Dr. Hamza Yeşilyurt

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

NEW ATTACKS ON RC4A AND VMPC

Mehmet Karahan
M.S. in Computer Engineering
Advisor: Assoc. Prof. Dr. İbrahim Körpeoğlu
August, 2015

RC4 is one of the most widely used stream cipher, designed by Ronald Rivest in 1987. RC4 has attracted a lot of attention of the community due to its simple design. In the last twenty years, lots of analyses about RC4 have been published by cryptanalysts. In these analyses, statictical biases and their applications stand out as the main weaknesses of RC4. To resist against this kind of weaknesses, many different varients of RC4 were designed. RC4A and VMPC are two of them, both proposed in FSE 2004. Here, we first reproduce two attacks against RC4 that depend on statistical biases; the linear correlation attack (Sepehrdad et al., 2010), and the plaintext recovery attacks (Alfardan et al., 2013). Then, we modify and apply them against RC4A and VMPC. We observe some previously undiscovered linear correlations and statistical biases for these two ciphers. Then, we try to identify the strong and weak aspects of these ciphers by evaluating the experimental results. We propose modifications for RC4, RC4A and VMPC according to these aspects and show that small changes in the design of these ciphers can increase or decrease their resistance against statistical bias attacks significantly.

Keywords: RC4,RC4A,VMPC,statistical bias.

ÖZET

RC4A VE VMPC ÜZERINE YENI ATAKLAR

Mehmet Karahan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Assoc. Prof. Dr. İbrahim Körpeoğlu
Ağustos, 2015

RC4, 1987 yılında Ronald Rivest tarafından tasarlanan, en yaygın olarak kullanılan akan şifre algoritmalarından biridir. RC4 basit tasarımı nedeniyle bilim camiasının çok dikkatini çekti. Son yirmi yılda, kripto analiz uzmanları tarafından RC4 ile ilgili pek çok analiz çalışması yayınlandı. Bu analizlerde, istatiksel önyargılar ve uygulamaları RC4'ün en önemli zayıflığı olarak dikkat çekmektedir. Bu zayıflıklara olan direnci arttırmak için, RC4'un çok farklı türleri tasarlandı. RC4A ve VMPC bunlardan ikisidir ve her ikisi de FSE 2004 konferansında önerilmiştir. Burada öncelikle, RC4'e yönelik istatiksel önyargılara dayanan iki atak ki bunlar; doğrusal korelasyon ve açık metin ele geçirme atağı, tekrar uygulayacağız. Sonra, bu atakları düzenleyerek, RC4A ve VMPC üzerinde uygulayacağız. Bu iki algoritma için, daha önce keşfedilmemiş bazı doğrusal korelasyonlar ve istatiksel önyargılar gözlemledik. Ayrıca, deneysel sonuçları değerlendirerek bu iki algoritmanın güçlü ve zayıf yönlerini tespit etmeye çalıştık. Tespit edilen bu özelliklere göre RC4, RC4A ve VMPC için değişiklikler önerdik ve bu algoritmaların tasarımındaki küçük değişikliklerin, istatiksel önyargı ataklarına karşı dirençlerini önemli ölçüde arttırabileceğini ya da azaltabileceğini gösterdik.

Anahtar sözcükler: RC4,RC4A,VMPC,istatiksel önyargı.

Acknowledgement

First of all, I would like to thank to Prof. Dr. Ali Aydın Selçuk because of his excellent guidance, support and mentoring. I was very lucky to have a chance to work with him.

I would like to thank to my advisor Assoc. Prof. Dr. İbrahim Körpeoğlu who accepted to be my advisor after Ali Aydın Selçuk had left Bilkent University. He allocated his plenty time for reading and making correction of my thesis.

I also thank to Assoc. Prof. Dr. Hamza Yeşilyurt who acccepted to be in the jury of my thesis defense. He guided and motivated me for my workings about cryptography in BS in Math.

I thank to my friends in Bilkent University, METU IAM and TUBITAK-UEKAE for their support.

Finally, I greatly thank to my family for their support, encouragement and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the dawn of humanity, information is the most valuable asset in human beings' life. The technique to keep information secure in communication named as cryptography. The word 'cryptography' comes from two Greek words 'kryptos' meaning hidden and 'graphein' meaning writing. In ancient times, cryptography was generally used for military and diplomatic purposes. In 3000 B.C., cryptography was first used in hieroglyphics to decorate thumbs of kings, which told their life and great acts. Spartans, a warrior society, developed a cryptographic device, a cylinder named Scytale, to send and receive messages secretly. Cesar cipher and Polybus square are:were two other important ancient ciphers. The main property of these ciphers is:was that they all depend on transposition or substitution. In transposition ciphers, order of letters in plaintext are rearranged systematically. In substitution ciphers, each letters of plaintext are replaced by another letters systematically. The idea of S-P boxes in modern cryptographic functions comes from these two operations. Until the 1900s, some mathematical methods have:had been applied in the area of cryptography but they are:were still manual methods. With the start of World War II, machines were started to be used widely and the winner of the war is:was determined by these cryptographic machines. In modern times, as technology grows up, any kind of information began to gain importance. Thus we need cryptography in every field of life and computers become the main primitive in cryptography.

In modern cryptography, security primitives can be analyzed under three main part; unkeyed, symmetric key and public key. Block and stream ciphers are two important primitives of symmetric key cryptography. We will focus on stream ciphers in this thesis. The idea of stream ciphers comes from one time pad which is perfectly secure according to Shannon definition of secrecy;

Definition 1. *Let our plaintext space is M and ciphertext space is C. Then, encryption algorithm is perfectly secure if for all $m \in M$ and for all $c \in C$;*

$$Pr(M = m|C = c) = Pr(M = m)$$

which means that ciphertext gives no information about plaintext without knowledge of key.

Definition 2 (Vernam Cipher)**.** *((5)) Let our plaintext $m = m_1 m_2 \dots m_t$ where each $m_i \in \{0, 1\}$ and key $k = k_1 k_2 \dots k_t$ where each $k_i \in \{0, 1\}$. Then ciphertext $c = c_1 c_2 \dots c_t$ is computed as;*

$$c_i = m_i \oplus k_i, \quad 1 \leq i \leq t.$$

If key k is randomly chosen and used only once, then the cipher is called one time pad.

One time pad is perfectly secure but has some limitations. First of all, generating random key is main problem since the key size is unlimited. Second problem is key exchange since each key is used only once and length of key is equal to length of plaintext.

Stream ciphers generate pseudo-random keystream by using a fixed size key. Key exchange is not a problem for stream ciphers since parties share small fixed size key and by using this key, they can produce keystream as long as they need.

Stream ciphers could be analyzed in two parts; ciphers that are based on feedback shift registers(FSRs) and ciphers that are not. FSRs based stream ciphers such as Trivium (6), Salsa20, and VEST (7) are well suited for hardware implementation. On the other hand, there are some stream ciphers which are

designed for software implementation such as SEAL (8) and RC4. In addition to these ciphers, Output Feedback(OFB) and Cipher Feedback(CFB) modes of block ciphers are also used as stream cipher.

Among these ciphers, RC4 is the most widely used stream cipher. Besides, the efficient and simple design of RC4 has aroused the interest of cryptanalysts. During last two decades, a big number of papers on the analysis of RC4 have been published at journals and conferences. Because of these anaylses, which reveal weak and strong aspects of RC4, some variants of RC4 have been proposed.

In this thesis, we make security analysis of two RC4 variants, RC4A (1) and VMPC (2) under plaintext recovery attacks (3) and linear correlation attack (9) which have been applied to RC4. This thesis organized as follows: in Chapter 3, we state these two attacks and discuss about result of these attacks against RC4. In Chapter 4, we summarize previous attacks against RC4A and apply these two attacks to RC4A. In Chapter 4, we summarize previous attacks against VMPC and apply these two attacks to VMPC. Finally, in Chapter 6, we conclude with some open problems.

Chapter 2

Preliminaries

We start with defining some notations which we will use in this work. Then, we define RC4 stream cipher and give information about the usage area of RC4 in section 2.2. Then, we state RC4A in section 2.3 and VMPC in section 2.4. Finally, we summarize previous works on these ciphers in section 2.5.

2.1 Notation

The algorithms RC4, RC4A, and VMPC are byte oriented stream ciphers. We consider RC4-n, RC4A-n, and VMPC-n to be n-bit oriented ciphers. We compare the results of attacks against these ciphers with their modified versions which are defined as follows;

- RC4-m is the modified version of RC4 in which internal variable j initializes in PRGA from its value in KSA.
- RC4A-m is the modified version of RC4A in which internal variables j_1, and j_2 initialize in PRGA from their value in KSA.
- VMPC-m is the modified version of VMPC in which internal variable j

initializes to 0 in PRGA.

2.2 RC4 Stream Cipher

RC4 is the most famous stream cipher which was designed by Ron Rivest in 1987. The design of RC4 was a trade secret until 1994 when the algorithm was anonymously posted to the Cypherpunks mailing list.

Generally, stream ciphers are based on linear feedback shift registers(LFSRs). The performance of these ciphers is efficient in hardware but slow in software. The main property of RC4 is that its implementation in both software and hardware is very easy and efficient.

Algorithm 1 KSA

for $i = 0$ to 255 **do**
  $S[i] \longleftarrow i$
end for
$j \longleftarrow 0$
for $i = 0$ to 255 **do**
  $K[i] \longleftarrow k[i \mod \ell]$
  $j \longleftarrow j + S[i] + K[i]$
  $swap(S[i], S[j])$
end for

Table 2.1: KSA of RC4 Stream Cipher.

RC4 has two main algorithms, Key Scheduling Algorithm (KSA) and the Pseudorandom Generation Algorithm (PRGA). KSA takes a key k as an input which has length ℓ typically between 5 to 32 bytes and by extending the key k, it permutes the identity array S. The second algorithm PRGA takes the permuted array S as an input and by using shuffle-exchange model, it produces the output bytes.

Algorithm 2 PRGA

initialization:
$i \longleftarrow 0$
$j \longleftarrow 0$
loop:
$i \longleftarrow i + 1$
$j \longleftarrow j + S[i]$
$swap(S[i], S[j])$
$Z \longleftarrow S[S[i] + S[j]]$

Table 2.2: PRGA of RC4 Stream Cipher.

2.2.1 Usage of RC4

Its simple design and efficiency made RC4 the most widely used stream cipher. RC4 has been used in many software applications and security protocols like Microsoft Windows, Secure SQL, Apple OCE, BitTorrent Protocol Encryption, Secure Shell, Remote Desktop Protocol, Kerboros, SASL, Gpcode.AK, PDF, Skype, WEP, WPA, WPA2, TLS/SSL.

The most famous usage of RC4 is the standardization of it in web and network security protocols like WEP, WPA, WPA2. Wired Equivalent Privacy (WEP), was the standard security protocol for WiFi security in IEEE 802.11, which uses RC4 as its core module. In time, serious security flaws have been found in WEP due to the usage of RC4. Thus, WEP need to be replaced by a new protocol. However, it was widely used by the industry which means replacing it by a totally new protocol will be costly and impractical. Thus, in 2003, WiFi Protected Access (WPA) was announced as the new standard security protocol which uses a patch to resolve essential security flaws of WEP. WPA also uses RC4 as its core module. One year later, in 2004, WiFi Protected Access II (WPA2) became available which uses AES block cipher as its core module. But, it is not easy to migrate to WPA2 since for hardware based applications using WEP, and WPA, it is neither cost effective nor easy to migrate completely away from the RC4 core. Because of this reason, WEP and WPA, which use RC4 as their core module, are still widely

used in network security protocols.

2.3 RC4A Stream Cipher

The stream cipher RC4A was proposed in (1) in 2004 which is designed by Souradyuti Paul and Bart Preneel . They tried to avoid changing general structure of RC4 while designing RC4A. The main design principle of RC4A is to reduce the correlations between output bytes and internal variables by making output bytes depend on more random variables. They also did not want to degrade the speed of RC4.

Algorithm 3 KSA

for $i = 0$ to 255 **do**
  $S_1[i] \longleftarrow i$
  $S_2[i] \longleftarrow i$
end for
$j_1 \longleftarrow 0$
$j_2 \longleftarrow 0$
for $i = 0$ to 255 **do**
  $j_1 \longleftarrow j_1 + S_1[i] + K_1[i \mod \ell]$
  $swap(S_1[i], S_1[j_1])$
end for
for $i = 0$ to 255 **do**
  $j_2 \longleftarrow j_2 + S_2[i] + K_2[i \mod \ell]$
  $swap(S_2[i], S_2[j_2])$
end for

Table 2.3: KSA of RC4A Stream Cipher (1).

Algorithm 4 PRGA

initialization:

$i \longleftarrow 0$

$j_1 \longleftarrow 0$

$j_2 \longleftarrow 0$

loop:

$i \longleftarrow i + 1$

$j_1 \longleftarrow j_1 + S_1[i]$

$swap(S_1[i], S_1[j_1])$

$Z^{(1)} \longleftarrow S_2[S_1[i] + S_1[j_1]]$

$j_2 \longleftarrow j_2 + S_2[i]$

$swap(S_2[i], S_2[j_2])$

$Z^{(2)} \longleftarrow S_1[S_2[i] + S_2[j_2]]$

Table 2.4: PRGA of RC4A Stream Cipher (1).

The key scheduling algorithm of RC4A is very similar to the KSA of RC4. The only difference is that, in RC4, KSA generates one state table by using one key, RC4A generates two state table by using two different keys. First key k_1 produces randomly and the second key k_2 is generated by using the first one. Designers mainly focus on pseudorandom generation algorithm of RC4A. There are two variables j_1 and j_2 corresponding to two internal tables S_1 and S_2. Each round, two bytes are generated. First byte is generated from S_1 by using index pointer $S_2[i] + S_2[j_2]$ and the second byte is generated from S_2 by using index pointer $S_1[i] + S_1[j_1]$. Thus, secret internal state of RC4A consists of arrays S_1, S_2 and variables i, j_1, j_2. Then, state space of RC4A is $N!^2 \times N^3$. For $N = 256$, $N!^2 \times N^3 \approx 2^{3392}$ while it is only 2^{1700} for RC4.

2.4 VMPC Stream Cipher

The stream cipher "Variably Modified Permutation Composition" (VMPC) was proposed in (2) in 2004, which is designed by Bartosz Zoltak. Design principles of algorithm could be summarized as follows;

1. After running KSA, no initial keystream outputs should be required to discarded.
2. The KSA should be strong against related key attacks.
3. The KSA should perform throughly random permutation on internal variables S and j.
4. Keystream outputs should not have any statistical biases.
5. There should be no short cycle in the keystream outputs.
6. Complexity of recovering the internal state from keystream output should be higher than brute force search.

Algorithm 5 KSA

for $i = 0$ to 255 **do**

$S[i] \longleftarrow i$

end for

$j \longleftarrow 0$

for $i = 0$ to 767 **do**

$i \longleftarrow i \ mod \ 256$

$j \longleftarrow S[j + S[i] + K[i \mod \ell]]$

$swap(S[i], S[j])$

end for

for $i = 0$ to 767 **do**

$i \longleftarrow i \ mod \ 256$

$j \longleftarrow S[j + S[i] + V[i \mod \ell]]$

$swap(S[i], S[j])$

end for

Table 2.5: KSA of VMPC Stream Cipher (2).

Algorithm 6 PRGA

initialization:

$i \longleftarrow 0$

loop:

$j \longleftarrow S[j + S[i]]$

$Z \longleftarrow S[S[S[j] + 1]]$

$swap(S[i], S[j])$

$i \longleftarrow i + 1$

Table 2.6: PRGA of VMPC Stream Cipher (2).

Definition 3 ((2))**.** *A k-level $VMPC$ function, referred as $VMPC_k$, is such transformation of S into the Z,*

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

where $x \in \{0, 1, \ldots, N\}$ and $P_k[x] = (P[x] + k) \bmod N$

In the Table 2.6, we can see implementation of 1-level VMPC function. All the previous analysis have done against 1-level VMPC function and in this thesis we also work on this function. In the KSA, there are two main differences from RC4. Firstly, the length of loop where array S is updated through shuffle exchange is $3N = 767$ and the update of array S. Secondly, there is an optional second loop in KSA where designer want to improve the diffusion by using initialization vector. However, designer claims that diffusion could be provided without using initialization vector. Thus, security analysis of KSA is performed without using initialization vector.

2.5 Previous Works

In this section, we summarize the previous works on RC4, RC4A, and VMPC. Attacks on RC4 can be categorized under four main directions (10);

1. Weak Keys, Key Collisions, Key Recovery from State
   First of all, KSA algorithm is not a one-to-one function. Thus, there exists key collisions which means that two different keys produce the same internal state. In 2000, Grosul and Wallach (11) first proposed the idea of colliding keys. To improve Grosul's result, in 2007, Biham and Dunkelman (12) gave a method which depends on differential changes in keys. In 2009, Matsui (13) introduced a practical method to produce colliding keys and gave examples of colliding keys when key length is equal to 24, 43 and 64 bytes. In 2011, Chen and Miyaji (14) experimentally found the shortest and the most practical colliding key pairs which have length 22 bytes. In 2013, Maitra et al. (15) discovered some methods to get near colliding key pairs which means that produced states are not totally the same but differ only by a few bytes.

   Secondly, KSA is not an easily reversible algorithm, so getting key efficiently from state forms one of the main part of the KSA attacks. The first paper to recover key from state proposed in 2007 by Paul and Maitra (16). In 2008

and 2009, there are four more papers published on this type of attack. (17; 18; 19; 20)

The keys which causes some certain biases in internal state or keystream outputs are named as weak keys. In 1995, Roos (21) and Wagner (22) independently defined two different sets of weak keys.

2. State Recovery from the Keystream
   State consists of array S, and variables i, j. Thus state space of RC4 is $N! \times N^2$. For $N = 256$, state space is $256! \times 256^2 \approx 2^{1700}$. These attacks try to reduce the complexity of search space.

   In 1998, Knudsen et al. (23) introduced the first full state recovery attack which reduces the complexity of getting state with exhaustive search considerably. In 2000, Golic (24) reduced the required number of known keystream outputs for the attack of Knudsen (23). In 2003, Shiraishi et al. (25) gave a different method which assumes some entries of the state are known. In 2007, Tomasevic et al. (26) introduced a method which reduces the complexity of full state recovery attack which proposed by Knudsen (23). In 2008, Maximov and Khovratovich (27) gave the best state recovery attack. In 2008, Golic and Morgari (28) published another paper which analyzes the attack of Maximov and they claimed that they improved data and time complexity of Maximov's attack. (27).

3. Biases and Distinguishers
   As a stream cipher, the main aim of RC4 is to produce pseudo-random keystreams. Because of this reason, studies on RC4 mainly focus on the biases and their application distinguishers.

   In 1995, Roos (21) discovered the first biased correlation between internal state in PRGA and key bytes. In 1996, Jenkins (29) published two biased correlations between internal state in PRGA and keystream outputs. In 2001, Mantin (30) analyzed the Jenkins' correlation (29) in detailed and generalized it. In 2000, Fluhrer and Mcgrew (31) gave first detailed list of long term biases in consecutive bytes which are named as digraph biases. In 2001, Mantin and Shamir (32) identified and proved that second keystream output has a positive bias towards zero which is the first practical and most

important bias in RC4. In 2002, Mironov (33) observed that first output byte of RC4 has a sinusoidal distribution and has also negative bias towards zero. In 2005, Mantin (34) analyzed the repetition of digraphs in RC4 output bytes and observed that the pattern ABSAB where A and B represent byte values and S represents a random string occurs more frequently. In 2008, Basu et al. (35) identified that the equality of two consecutive bytes has a positive bias under certain conditions. In 2010, Pouyann et al. (9) gave a method which reveals all correlations in PRGA. By using this method, they confirmed all correlations that have found until that time and explored 48 new correlations. In 2011, Gupta et al. (36) proved that there is a relation between key length and some certain biases in the output bytes. In 2011, Maitra et al. (37) proved that all initial output bytes except first one have a bias towards zero. In 2013, Maitra et al. (10) identified that there exists a long term correlation between the bytes Z_{kN} and Z_{kN+2}. In 2013, Isobe et al. (38) identified and proved two new short term biases. In 2013, Alfardan et al. (3) experimentally listed all short term biases to exploit these biases in plaintext recovery attack. In this work, they did not give proof of new described biases. In 2013, Sarkar, Gupta, paul, Maitra (39) proved the biases which are discovered in (3).

4. Key Recovery from the Keystream

   In these attacks, secret key is tried to be recovered by using output bytes and exploitable biases related to them. The main target of these attacks is the use of RC4 in protocols such as WEP, WPA and TLS.

   In 2001, Fluhrer et al. (40) claimed theoretically that 4 million packets (plaintext-ciphertext pair) were enough to recover the key with success probability 0.5 by using incremental IVs in WEP. In 2004, Korek (41) reduced the key recovery complexity and introduced a practical attack which recovers the key with 100000 packets. In 2006, Klein (42) used a new strategy and claimed that packet complexity was reduced to 25000 with success probability 0.5. In 2011, Sepehrdad et al. (43) implemented this attack and showed that packet complexity is 60000. In 2007 , Tews, Weinmann and Pyshkin (44) proposed a new key recovery attack which reduces packet

complexity to 40000. In the same year, Vaudenay and Vuagnoux (45) introduced another attack which has packet complexity equal to 32750. In 2009, Tews and Beck (46) improved attack (45) and reduced the packet complexity to 24200.

To fix the vulnerabilities in WEP, WPA uses Temporal Key Integrity Protocol (TKIP) which generates a new key dynamically for each packet. In WPA, brute force attack to get temporary encryption key (TK) has complexity 2^{128}. First attack against RC4 in WPA proposed in 2004 by Moen, Raddum, and Hole (47) and they claimed that it is possible to recover TK with complexity 2^{105} by using two packets. During 2011-2012, Sepehrdad, Vaudenay and Vuagnoux (48; 43) gave a new TK recovery attack with complexity 2^{96} by using 2^{38} packets.

There are three important attacks against RC4A which we will give detailed analysis of them in Chapter 4. Briefly, the first paper about RC4A is published by Maximov (49) in 2005. He observed that equality of consecutive odd output bytes is not equal to random association. In the same year, Tsunoo et al. (50) observed and proved the biased correlation between first and third bytes. In 2013, Sarkar (51) showed that second byte of RC4A has a positive bias towards 2.

There are four important attacks against VMPC which we will give detailed analysis of them in Chapter 5. In 2005, Maximov (49) observed that there exist long term biases in consecutive output bytes of VMPC. In the same year, Tsunoo et al. (50) proved the biased correlation between first and second bytes. In 2012, Li et al. (52) improved the result of Tsunoo et al. (50). In 2013, Sarkar (51) introduces a new biased correlation between second and fourth bytes of VMPC.

Chapter 3

Attacks on RC4

In section 3.1, we summarize the plaintext recovery attacks. In subsection 3.1.1 we compare RC4 and RC4-m against single byte bias plaintext recovery attack. Then, we summarize linear correlation attack in section 3.2.

3.1 Plaintext Recovery Attacks

In 2013, Alfardan et al. have presented two plaintext recovery attacks on RC4 in TLS (3). First one is the single-byte bias attack in which they use short-term (involving only initial 256 bytes) biases in RC4 and try to get initial 256 plaintext bytes. Second one is the double-byte bias attack in which they use long-term (not just initial 256 bytes) biases in the distribution of consecutive bytes and try to get plaintext bytes at any position. They apply these attacks against RC4-8.

In this work, we will apply these attacks to the RC4A-4 and VMPC-4 stream ciphers since the complexities of these attacks on these ciphers are infeasible for a standard PC for 8-bit oriented versions. For consistency, we will first apply these attacks to the RC4-4.

3.1.1 Single Byte Bias Plaintext Recovery Attack

We first summarize important single byte biases, then we describe attack in detailed and finally we give experimental results of this attack.

3.1.1.1 Single Byte Biases

Until the work of Alfardan et al. (3), there are some well known single byte biases in RC4 keystream outputs;

Theorem 1 ((32))**.** *The probability that the second output byte is equal to zero is*

$$Pr(Z_2 = 0) \approx \frac{2}{N}$$

Theorem 2 ((10))**.** *For* $3 \leq r \leq 255$*, the probability that the* r*th output byte is equal to zero is*

$$Pr(Z_r = 0) = \frac{1}{256} + \frac{c_r}{256^2}$$

where $c_3 = 0.351089$ *and* $c_4 = 1.337057 \geq c_r \geq c_{255} = 0.242811$ *where* $c_4, \ldots, c_{255}$ *is a decreasing sequence.*

Theorem 3 ((10))**.** *Assume key length is equal to* ℓ*. Then, the probability that* ℓ*th output byte is equal to* $256 - \ell$ *is*

$$Pr(Z_\ell = -\ell) \geq \frac{1}{2^{16}}$$

In addition to these known biases, Alfardan et al. have observed some new short-term biases which can be analyzed in two parts.

1. **Isolated short-term biases:**
   There are three positive biased events. Two of them are $Z_2 = 172$, and $Z_4 = 2$ which have proved by Sarkar et al. (39). The other one is $Z_3 = 131$ which have proved by Isobe et al. (38). There are four negative biased events. Three of them are $Z_1 = 129$, $Z_2 = 129$, and $Z_{256} = 0$ which

have proved by Sarkar et al. (39). The other one is $Z_2 = 2$ which have proved by Sarkar (51). Among these biased events, the most interesting one is the $Z_1 = 129$ since it only observed when key length ℓ is equal to $2, 4, 8, 16, 32, 64, 128$ which are non-trivial factors of 256.

2. **Regular short-term bias:**
   There are two positive biased events. First one is the $Z_r = r$ which is observed for all r and decrease according to r. It is proven by Isobe et al. (38). The second one is an extension of Theorem 3. The biased event $Z_r = -r$ is not only observed when r is equal to key length ℓ, but also when r is equal to multiples of ℓ.

3.1.1.2 Attack

It is a broadcast attack which means fixed-plaintext bytes are encrypted under a large number of independent keys. In this attack (3), they use all short-term biases in RC4 and try to get initial 256 plaintext bytes.

First of all, by using a large number of independent keys, they get statistical distribution of Z_r for all $r \in \{1, 2, \ldots, 256\}$. Thus, they empirically get the probabilities;

$$p_{r,k} := Pr(Z_r = k), \quad k = 0 \times 00, \ldots, 0 \times FF$$

They use maximum-likelihood approach and recover plaintext bytes with maximum certainty by applying the following steps;

Assume we have a sample $\{C_1, \ldots, C_S\}$ which consists of encryptions of the same plaintext with S independent keys. Then, for any position r and any possible plaintext byte candidate μ, following values are calculated;

$$I_k^{(\mu)} = |\{j | C_{j,r} = k \oplus \mu\}_{1 \leq j \leq S}| \quad (0 \times 00 \leq k \leq 0 \times FF)$$

The vector $(I_{0\times00}^{(\mu)}, \ldots, I_{0\times FF}^{(\mu)})$ represent the number of matching between $\{C_{j,r}\}_{1 \leq j \leq S}$ and encryption of μ with key k. By using these induced distributions and empirical distributions $p_{r,0\times00}, \ldots, p_{r,0\times FF}$, the probability that candidate plaintext byte μ is encrypted to ciphertext bytes $\{C_{j,r}\}_{1 \leq j \leq S}$, which is a

Algorithm 7 Single-byte bias attack

input: $\{C_j\}_{1\leq j\leq S}$ - S independent encryptions of fixed plaintext P
r - byte position
$(p_{r,k})_{0\times 00\leq k\leq 0\times FF}$- keystream distribution at position r
output: P_r^* - estimate for plaintext byte P_r
$N_{0\times 00}\leftarrow 0,\ldots,N_{0\times FF}\leftarrow 0$
for $j=1$ to S **do**
  $I_{C_{j,r}}\leftarrow I_{C_{j,r}}+1$
end for
for $\mu=0\times 00$ to $0\times FF$ **do**
  for $k=0\times 00$ to $0\times FF$ **do**
    $I_k^{(\mu)}\leftarrow I_{k\oplus\mu}$
  end for
  $\lambda_\mu\leftarrow\sum_{k=0\times 00}^{0\times FF} I_k^{(\mu)} log p_{r,k}$
end for
$P_r^*\leftarrow argmax_{\mu\in\{0\times 00,\ldots,0\times FF\}}\lambda_\mu$
return P_r^*

Table 3.1: Single Byte Bias Attack (3).

multinomial distribution, calculated as follows;

$$\lambda_\mu=\frac{S!}{I_{0\times 00}^{(\mu)}!\ldots I_{0\times FF}^{(\mu)}!}\prod_{k\in\{0\times 00,\ldots,0\times FF\}} p_{r,k}^{N_k^{(\mu)}}$$

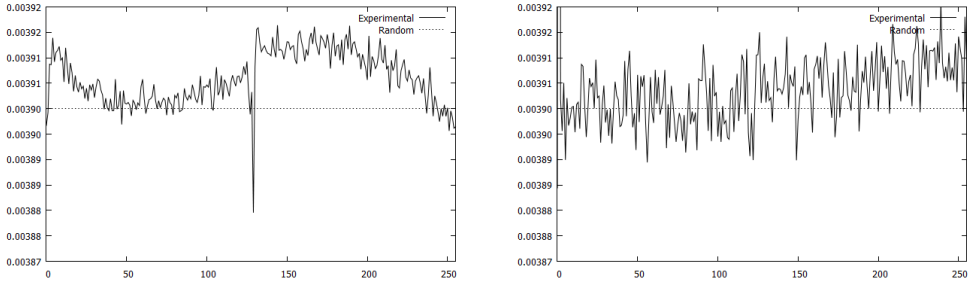
For all $0\times 00\leq\mu\leq 0\times FF$, the probability λ_μ calculated and maximum-likelihood plaintext byte value determined.

There are two possible optimization in above calculations. Firstly, $(I_{0\times 00}^{(\mu)},\ldots,I_{0\times FF}^{(\mu)})$ and $(I_{0\times 00}^{(\mu')},\ldots,I_{0\times FF}^{(\mu')})$ are permutations of each other because $I_k^{(\mu)}=I_{k\oplus\mu'\oplus\mu}^{(\mu')}$. Thus, the term $S!/(I_{0\times 00}^{(\mu)}!\ldots I_{0\times FF}^{(\mu)}!)$, will be equal for all μ, which means that this value can be ignored. Secondly, taking $log(\lambda_\mu)$ instead of λ_μ will slightly reduce the complexity of attack.

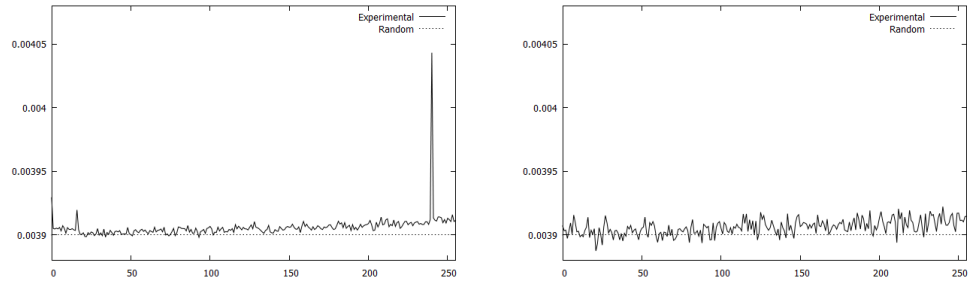
3.1.1.3 Experimental Results

We can not simulate attack against 8-bit versions because of high complexity, but we get the distribution of RC4-8 and RC4-m-8 keystream outputs to observe the

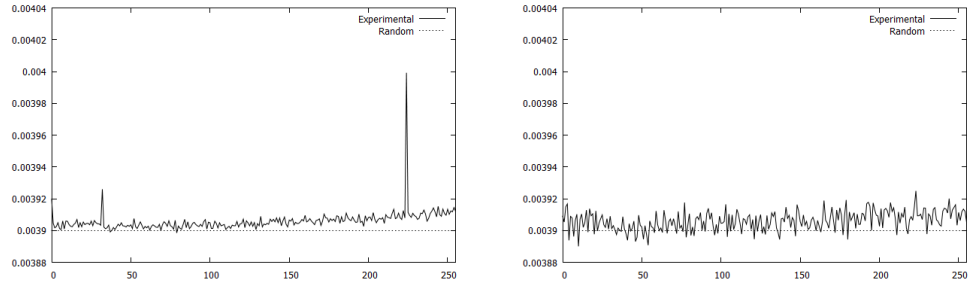
effects of modification clearly.



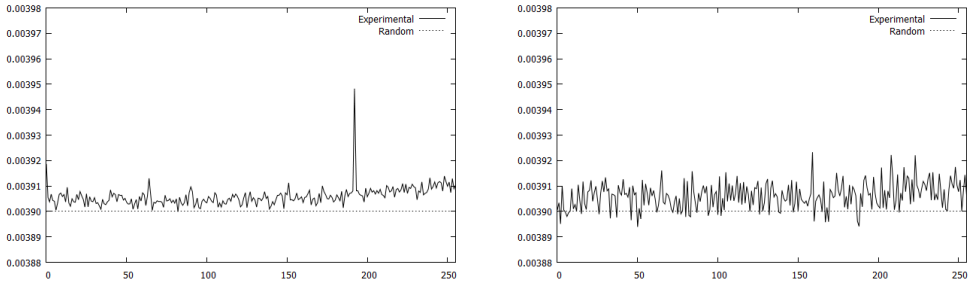
(a) Z_1 (b) Z_1 of RC4-m-8



(c) Z_{16} (d) Z_{16} of RC4-m-8



(e) Z_{32} (f) Z_{32} of RC4-m-8



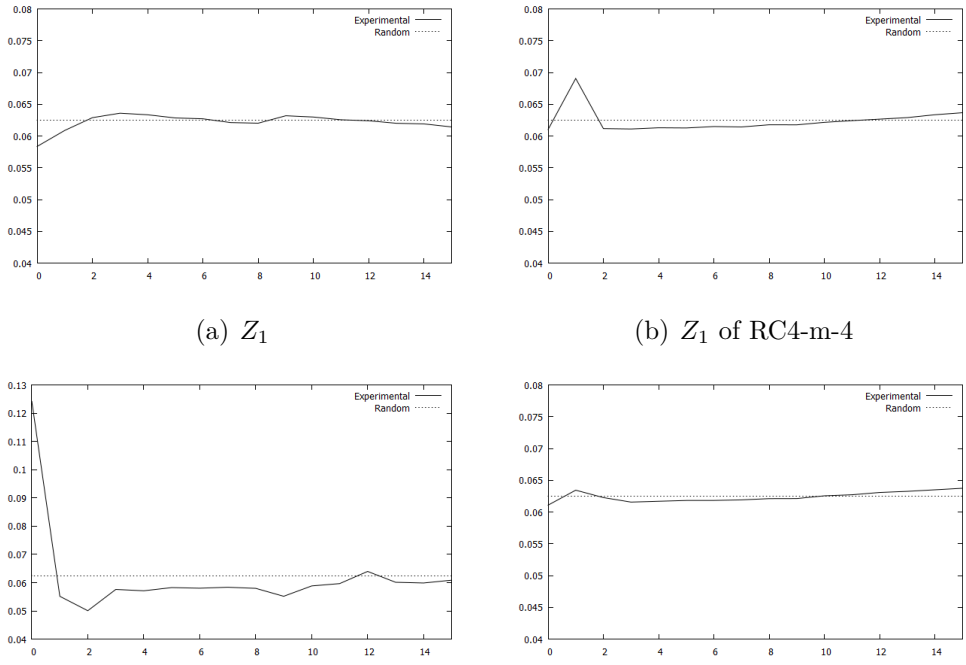
(g) Z_{64} (h) Z_{64} of RC4-m-8

Figure 3.1: Measured distributions of RC4-8 and RC4-m-8 keystream outputs Z_1, Z_{16}, Z_{32}, and Z_{64}.

As it is seen in Figure 3.1, the important biases mentioned above for RC4-8 are not observed for RC4-m-8. On the other hand, for each keystream output positions, relatively smaller biases are observed in modified version.

We simulated single-byte plaintext recovery attack against RC4-4 and RC4-m-4. First of all, by using 2^{28} independent keys, we estimate the probabilities $\{p_{r,k}\}_{1\leq r\leq 256, 0\times 00\leq k\leq 0\times FF}$.

Figure 3.2 shows distribution of keystream outputs Z_1, and Z_2 for RC4-4 and RC4-m-4. The biases $Z_r = 0$, $Z_2 = 0$, $Z_r = r$, also seen in reduced version of RC4. Here, we need to take key length equal to plaintext size, because a large number of strong biases are observed in keystream distributions when we take it smaller. Thus, we take key length 64 bits which is equal to plaintext size. Because of this reason, the bias $Z_r = -r$ is not observed.



(a) Z_1 (b) Z_1 of RC4-m-4

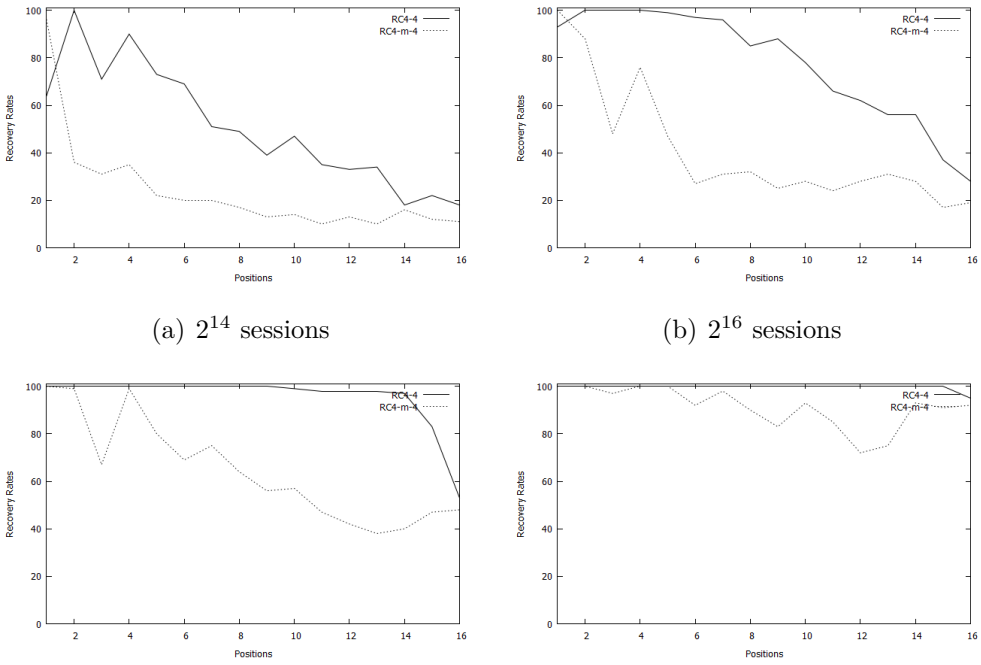
(c) Z_2 (d) Z_2 of RC4-m-4

Figure 3.2: Measured distributions of RC4-4 and RC4-m-4 keystream outputs Z_1 and Z_2.

We analyze single-byte plaintext recovery attack under four different session

size, $2^{14}, 2^{16}, 2^{18}, 2^{20}$. We run the Algorithm 7 for RC4-4 and RC4-m-4 100 times for each sessions.

- With $S = 2^{14}$ sessions, a few positions of plaintext recovered with high rates. This is the result of number and power of biases observed in that positions. For example, recovery rate of second position is 100% because of the existence of strong bias towards 0. In general, recovery rates for RC4-m-4 is almost half of the rates of RC4-4.
- With $S = 2^{16}$ sessions, the first 8 positions are recovered with rate more than 90% for RC4-4.
- With $S = 2^{18}$ sessions, first 14 positions recovered with rate very close to 100% for RC4-4. But for RC4-m-4, only three positions recovered with 100% and the recovery rates of other positions changes between 42% and 75%.
- With $S = 2^{20}$ sessions, recovery rates for RC4-4 are almost equal 100% for all positions and recovery rates for RC4-m-4 are very close to rates of RC4-4.



(a) 2^{14} sessions (b) 2^{16} sessions

(c) 2^{18} sessions (d) 2^{20} sessions

Figure 3.3: Recovery rates for RC4-4 and RC4-m-4.

As a result, in PRGA, initializing j from the value in KSA reduces the biases in keystream outputs and also reduces the success rate of single-byte plaintext recovery attack.

3.1.2 Double Byte Bias Plaintext Recovery Attack

3.1.2.1 Multi Byte Biases

Multi byte bias is the correlation between two or more positions in the keystream outputs. Most of these biases are long term. In 2000, Fluhrer and McGrew (4) analyzed the distribution of consecutive keystream outputs (Z_r, Z_{r+1}). They observed the most extensive set of multi byte biases as seen in Table 3.2. Alfardan et al. (3) experimentally checked distribution of all consecutive keystream outputs. They verified the biases observed by Fluhrer and McGrew for 128 bit keys

and they did not observe any additional bias in consecutive bytes.

In addition to Fluhrer McGrew biases, in 2005, Mantin (34) observed a positive bias towards the pattern $ABSAB$ where A, B are the byte values and S is the random string. In 2012, Sen Gupta et al. (10) observed a bias in keystream positions (Z_r, Z_{r+2}) towards the value $(0, 0)$. However, in double byte bias plaintext recovery attack, only the biases in consecutive keystream outputs are used so biases which observed by Mantin and Sen Gupta et al. can not be used.

Byte pair	Conditon on i	Probability
$(0, 0)$	$i = 1$	$2^{-16}(1 + 2^{-9})$
$(0, 0)$	$i \neq 1, 255$	$2^{-16}(1 + 2^{-8})$
$(0, 1)$	$i \neq 0, 1$	$2^{-16}(1 + 2^{-8})$
$(i + 1, 255)$	$i \neq 254$	$2^{-16}(1 + 2^{-8})$
$(255, i + 1)$	$i \neq 1, 254$	$2^{-16}(1 + 2^{-8})$
$(255, i + 2)$	$i \neq 0, 253, 254, 255$	$2^{-16}(1 + 2^{-8})$
$(255, 0)$	$i = 254$	$2^{-16}(1 + 2^{-8})$
$(255, 1)$	$i = 255$	$2^{-16}(1 + 2^{-8})$
$(255, 2)$	$i = 0, 1$	$2^{-16}(1 + 2^{-8})$
$(129, 129)$	$i = 2$	$2^{-16}(1 + 2^{-8})$
$(255, 255)$	$i \neq 254$	$2^{-16}(1 - 2^{-8})$
$(0, i + 1)$	$i \neq 0, 255$	$2^{-16}(1 - 2^{-8})$

Table 3.2: Fluhrer-McGrew biases for consecutive pairs of byte values (4).

3.1.2.2 Attack and Experimental Results

Double byte plaintext recovery attack (3) makes it possible to recover the plaintext at any position. In this attack, first and last positions are assumed to be known and attacker tries to recover positions between them. Also, it is not a broadcast attack, it works for plaintexts repeatedly encrypted under a single key. Attack mainly makes use of biases in consecutive keystream outputs, so the only biases observed by Fluhrer and McGrew can be used which are listed in Table 3.2. Thus the probabilities;

$$p_{r,k_1,k_2} = Pr[(Z_{256q+r}, Z_{256q+r+1}) = (k_1, k_2)]$$

are taken from Table 3.2 for the appropriate r, k_1, k_2 and all other probabilities are assumed to be equally likely to appear in the keystream distribution.

Assume we have plaintext $P = P_1||\dots||P_L$, L byte where L is a multiple of 256. P is encrypted repeatedly with a single key and ciphertext C is obtained. Let C_j denote the jth encryption of P and let $C_{j,r}$ denote the rth byte of C_j.

As in the algorithm 7, the most likely plaintext candidate pair (μ_r, μ_{r+1}) for position r could be computed by using ciphertext bytes $\{(C_{j,r}, C_{j,r+1})\}_{1\le j\le S}$ and probabilities $\{p_{r,k_1,k_2}\}_{0\times 00\le k_1,k_2\le 0\times FF}$. Since we have overlapping byte pairs in this attack, it could be more accurate to estimate plaintext candidate correctly than by just considering individual byte pairs. Thus, differently from the single byte attack, we compute estimated likelihood $\lambda_{P'} = \lambda_{\mu_1||\dots||\mu_L}$ for any plaintext candidate $P' = \mu_1||\dots||\mu_L$ which is a recursive computation;

$$\lambda_{\mu_1||\dots||\mu_{\ell-1}||\mu_\ell} = \delta_{\mu_\ell|\mu_{\ell-1}} \cdot \lambda_{\mu_1||\dots||\mu_{\ell-1}} \quad (\ell \le L)$$

where $\delta_{\mu_\ell|\mu_{\ell-1}} = Pr(P_\ell = \mu_\ell | P_{\ell-1} = \mu_{\ell-1})$ which means that μ_ℓ is dependent on $\mu_{\ell-1}$. For the base case, we assume that $\lambda_{\mu_1} = Pr(P_1 = \mu_1)$ is known and then estimated likelihood can be written as $\lambda_{P'} = Pr(P_1 = \mu_1) \prod_{\ell=2}^{L} \delta_{\mu_\ell|\mu_{\ell-1}}$.

The algorithm chooses plaintext candidate $P^* = \mu_1||\dots||\mu_L$ which has the maximum estimated likelihood λ_{P^*}. For the likelihood λ_{P^*}, we have optimality preserving property that is for all prefixes $\mu_1||\dots||\mu_{\ell-1}$ of P^* where $\ell \le L$, $\lambda_{\mu_1||\dots||\mu_{\ell-1}}$ have the largest value through all $(\ell-1)$-length plaintext candidates which have $\mu_{\ell-1}$ as their last byte.

The plaintext P^* is consructed iteratively by evaluating the prefixes of P^* with increasing length. At each step until the last byte, we have a set of candidates with size N. For example, let we have N candidates $\mu_1||\dots||\mu_{\ell-1}$ with $\ell-1$ length. Then for all possible values of μ_ℓ, the likelihood estimates of $\mu_1||\dots||\mu_{\ell-1}||\mu_\ell$ are computed and the maximum one is chosen and added to the set of candidates with length ℓ. When we reach the length of P^*, we get P^* itself because we have only one candidate for the last byte. The computation of $\lambda_{\mu_{i+1}|\mu_i}$ is similar to maximum likelihood computation in Algorithm 7. First of all, we compute the

vector $(I_{i,0\times00,0\times00},\ldots,I_{i,0\times FF,0\times FF})$, where

$$I_{i,k_1,k_2} = |\{1 \leq j \leq S | (C_{j,i}, C_{j,i+1}) = (k_1 \oplus \mu_i, k_2 \oplus \mu_{i+1})\}|$$

Then, the plaintext byte pairs (μ_i, μ_{i+1}) are encrypted to ciphertext byte pairs $(C_{j,i}, C_{j,i+1})$ follows a multinomial distribution and computed as

$$Pr(P_i = \mu_i \wedge P_{i+1} = \mu_{i+1}|C) = \frac{S!}{I_{i,0\times00,0\times00}! \ldots I_{i,0\times FF,0\times FF}!} \prod_{k_1,k_2 \in \{0\times00,\ldots,0\times FF\}} p_{i,k_1,k_2}^{I_{i,k_1,k_2}}$$

Then, $\delta_{\mu_{i+1}|\mu_i}$ computed as

$$\begin{aligned} \delta_{\mu_{i+1}|\mu_i} &= Pr(P_{i+1} = \mu_{i+1}|P_i = \mu_i \wedge C) \\ &= \frac{Pr(P_i = \mu_i \wedge P_{i+1} = \mu_{i+1}|C)}{Pr(P_i = \mu_i|C)} \end{aligned}$$

Since we produce long keystream by using single key, we can assume that there is no significant single byte bias. Thus the term $Pr(P_i = \mu_i|C)$ can be ignored. Besides, the terms $S!/I_{i,0\times00,0\times00}! \ldots I_{i,0\times FF,0\times FF}!$ will be ignored due to the similar reasons as in Algorithm 7.

Because of high complexity, we apply this algorithm against RC4-4. We encrypted $S = 1 \cdot 2^{17}, \ldots, 16 \cdot 2^{17}$ copies of the plaintext and assume that first and last plaintext positions are known and try to get the positions between them. We apply algorithm 100 times for each session and success rate of algorithm seen in Figure 3.4.

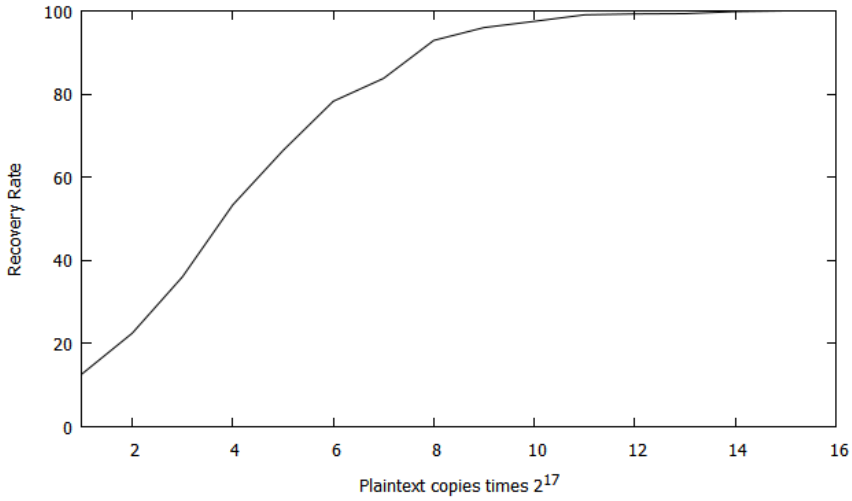


Figure 3.4: Success rate for recovering all positions for RC4-4.

Algorithm 8 Double-byte bias attack

input: C - encryption of S copies of fixed plaintext P
L - length of P (multiple of 256)
$(p_{r,k_1,k_2})_{1\leq L-1,0\times 00\leq k_1,k_2\leq 0\times FF}$- keystream distribution
output: P^* - estimate for plaintext P
notation: $max_2(Q)$ denote $(P,\lambda)\in Q$ such that $\lambda\geq\lambda'\forall(P',\lambda')\in Q$
$I_{(r,k_1,k_2)}\leftarrow 0$ for all $1\leq r<L$, $0\times 00\leq k_1,k_2\leq 0\times FF$
for $j=1$ to S **do**
  for $r=1$ to $L-1$ **do**
    $I_{(r,C_{j,r},C_{j,r+1})}\leftarrow I_{(r,C_{j,r},C_{j,r+1})}+1$
  end for
end for
$Q\leftarrow\{(\mu_1,0)\}$
for $r=1$ to $L-2$ **do**
  $Q_{ext}\leftarrow\{\}$ //List of plaintext candidates of length $r+1$
  for $\mu_{r+1}=0\times 00$ to $0\times FF$ **do**
    $Q_{\mu_{r+1}}\leftarrow\{\}$ //List of plaintext candidates ending with μ_{r+1}
    for each $(P',\lambda_{P'})\in Q$ **do**
      $P'\rightarrow\mu_1||\ldots||\mu_r$
      $\lambda_{P'||\mu_{r+1}}\leftarrow\lambda_{P'}+\sum_{k_1=0\times 00}^{0\times FF}\sum_{k_2=0\times 00}^{0\times FF}I_{(r,k_1\oplus\mu_r,k_2\oplus\mu_{r+1})}\cdot\log p_{(r,k_1,k_2)}$
      $Q_{\mu_{r+1}}\leftarrow Q_{\mu_{r+1}}\cup\{(P'||\mu_{r+1},\lambda_{P'||\mu_{r+1}})\}$
    end for
    $Q_{ext}\leftarrow Q_{ext}\cup\{max_2(Q_{\mu_{r+1}})\}$
  end for
  $Q\leftarrow Q_{ext}$
end for
$Q_{\mu_L}\leftarrow\{\}$ //List of plaintext candidates ending with μ_r
for each $(P',\lambda_{P'})\in Q$ **do**
  $P'\rightarrow\mu_1||\ldots||\mu_{L-1}$
  $\lambda_{P'||\mu_L}\leftarrow\lambda_{P'}+\sum_{k_1=0\times 00}^{0\times FF}\sum_{k_2=0\times 00}^{0\times FF}I_{(r,k_1\oplus\mu_{L-1},k_2\oplus\mu_L)}\cdot\log p_{(r,k_1,k_2)}$
  $Q_{\mu_L}\leftarrow Q_{\mu_L}\cup\{(P'||\mu_L,\lambda_{P'||\mu_L})\}$
end for
$(P^*,\lambda_{P^*})\leftarrow max_2(Q_{\mu_L})$
return P^*

Table 3.3: Double Byte Bias Attack (3).

3.2 Linear Correlation Attack

In 2010, Sepehrdad, Vaudenay and Vuagnoux (9) presented a new technique which reveals linear correlations in PRGA of RC4. By applying this technique, they confirmed all known biases and exploited 48 new ones in PRGA.

3.2.1 Known Correlations in the PRGA of RC4

Notation: In this work, let j_r be the value of variable j during the round r of PRGA and S_r be the value of array S after round r of PRGA.

- **Jenkins Correlations:**((29)) The relations at rth round of PRGA $S_r[j_r] = i_r - Z_r$ and $S_r[i_r] = j_r - Z_r$ occurs with the probability $2/N$.

- **Paul, Rathi and Maitra Correlation:**((53)) The probability distribution of the output index that selects the first keystream output is;

$$Pr(S_1[1] + S_1[j_1] = x) = \begin{cases} \frac{1}{N} & \text{for odd } x, \\ \frac{1}{N} - \frac{2}{N(N-1)} & \text{for even } x \neq 2, \\ \frac{2}{N} - \frac{1}{N(N-1)} & \text{for even } x = 2. \end{cases}$$

- The bias in second keystream output which is stated in Theorem 1 is another known correlation in PRGA.

3.2.2 Attack

There is no specific method to discover the biased correlations mentioned in previous section. Sepehrdad et al. (9) gave a method to rediscover all known biases and to find new ones as well. In this method, they define linear equations

which consist of internal values of a round of the PRGA, $\{j_i, S_i[i], S_i[j_i]\}$ and keystream output of that round, z_i as;

$$(c_0 \cdot j_i + c_1 \cdot S_i[i] + c_2 \cdot S_i[j_i] + c_3 \cdot z_i) \bmod N = C$$

The equation has four coefficients and to evaluate all possible linear correlations they need to take values of c_i from the set $\{0, 1, \ldots, 255\}$. This defines 2^{40} linear equations which is too large to evaluate. Thus, they take values of c_i from the set $\{-1, 0, 1\}$. By using 10^9 randomly chosen keys with length 16 bytes, they verify all linear equations for first 256 rounds. In each round, when an equation holds, a counter for that one is incremented. In this way, they rediscovered all known biased correlations in PRGA, and also they discover 48 new biased correlations.

j_i	$S_i[i]$	$S_i[j_i]$	z_i	C	Probability	Discovered	Proved
1	-1	0	-1	0	$2/N$	(29)	(30)
0	0	1	1	i	$2/N$	(29)	(30)
0	1	1	-1	0	$1.9/N$	(9)	(36)
0	1	1	-1	1	$0.89/N$	(9)	(36)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	1	1	-1	255	$1.25/N$	(9)	(36)
0	1	1	1	i	$0.95/N$	(9)	—
1	1	0	0	i	$0.95/N$	(9)	—
1	1	-1	0	i	$2/N$	(9)	(36)
1	-1	1	0	i	$2/N$	(9)	(36)
1	-1	0	0	1	$0.9/N$	(9)	—
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	-1	0	0	255	$1.25/N$	(9)	—
1	-1	0	0	0	$1.9/N$	(9)	(36)
0	0	1	0	$i+1$	$1.36/N$	(9)	—
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	1	0	255	$0.9/N$	(9)	—
0	0	1	0	i	$2.34/N$	(9)	(36)

Table 3.4: Biased linear correlations observed in all rounds of PRGA in RC4. Note that the probability of some biases increase or decrease according to i.

The important biased correlations observed in all rounds that was found by this technique are seen in Table 3.4. Especially, for the first rounds, there exists some additional biased correlations, they are listed in Table 3.5, and Table 3.6. The strong biased correlations are proved by Gupta et al. (36).

j_i	$S_i[i]$	$S_i[j_i]$	z_i	C	Probability	Discovered	Proved
0	1	0	-1	0	$0.95/N$	(9)	
0	1	1	0	2	$1.95/N$	(53)	(53)
1	1	0	0	2	$1.94/N$	(9)	(36)

Table 3.5: Additional biased linear correlations observed in the first round of PRGA in RC4.

j_i	$S_i[i]$	$S_i[j_i]$	z_i	C	Probability	Discovered	Proved
0	0	0	1	0	$2/N$	(32)	(32)
1	-1	1	-1	0	$2/N$	(9)	(36)
1	1	0	-1	even	$1.0183/N$	(9)	—
1	1	0	-1	odd	$1.0316/N$	(9)	—
1	0	1	0	6	$2.37/N$	(9)	(36)
1	0	-1	0	255	$0.75/N$	(9)	—
1	-1	1	0	0	$2/N$	(9)	(36)
0	-1	1	0	0	$0.95/N$	(9)	—

Table 3.6: Additional biased linear correlations observed in the second round of PRGA in RC4.

Chapter 4

Security Analysis of RC4A

We start with previous attacks against RC4A and give a brief summary of works of Maximov (49) , Tsunoo (50), and Sarkar (51) on RC4A. Then, we apply plaintext recovery attacks and linear correlation attack against RC4A.

4.1 Previous Attacks Against RC4A

4.1.1 The Attack of Maximov

In 2005, Maximov (49) showed that RC4A is a strong cipher against consecutive digraph biases which is the general weakness of RC4 family of stream cipher. Then, he made additional investigation and checked the values of $Pr(Z_t = x, Z_{t+2} = y)$ for small number n, for RC4A-n. He noticed that $Pr(Z_t = Z_{t+2}|t \text{ is odd})$ is not equal to random association. He focused on that anomaly and got the following theoretical bias value;

Theorem 4 ((49))**.** *Assume KSA of RC4A performs throughly random permutation which means internal variables are from the uniform distribution. Then;*

$$Pr(Z_t = Z_{t+2}|t \text{ is odd}) \approx 2^{-8}(1 - 2^{-30.05})$$

He also checked the assumptions of theorem and found that KSA of RC4A is not perfect and internal values are not throughly random. Because of this reason, his experimental results were more consistent than theoretical ones.

4.1.2 The Attack of Tsunoo et al.

When Paul and Preneel proposed RC4A (1), they have showed a new bias in RC4 about equality between first and second output bytes. In 2005, Tsunoo et al. (50) discovered the biased correlation between first and third keystream outputs of RC4A by using the idea of Paul and Preneel. In this attack, they give three assumptions;

1. $S_1[1] = 2$, (equal assumption to paper [pp]) ($P_1 = \frac{1}{256}$)
2. $A \neq 0xff$,(A is entry of $S_1[2]$) ($P_2 = \frac{255}{256}$)
3. $B \neq A + 2$, (B is entry of $S_2[1]$) ($P_3 = \frac{255}{256}$)

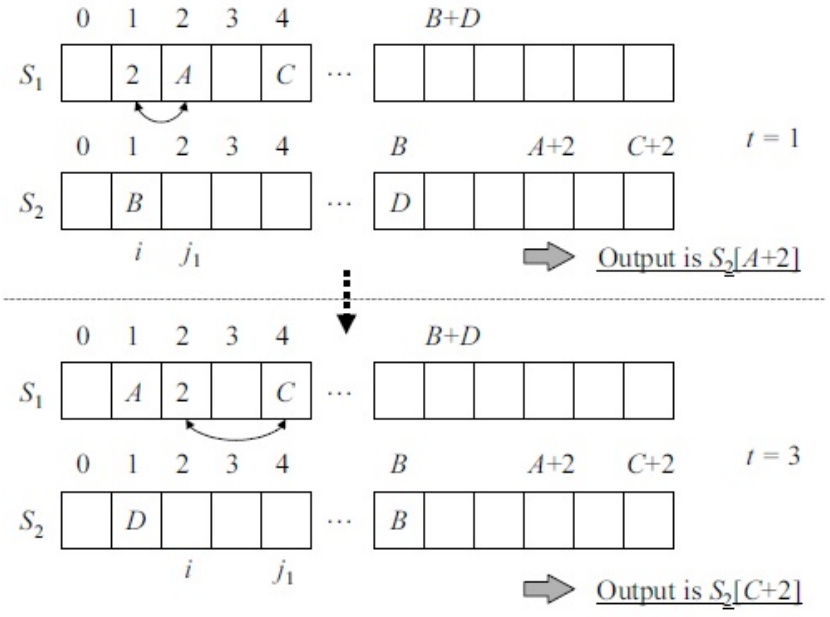


Figure 4.1: Transition of arrays S_1 and S_2.

At time $t = 1$; index j_1 updated by the following equation;

$$j_1 = j_1 + S_1[i] = 0 + S_1[1] = 2$$

Then, first and second indexes are swapped, and $S_2[A + 2]$ is output as first keystream. At time $t = 2$; index j_2 updated to value of first index of S_2. Then $S_2[1]$ and $S_2[B]$ are swapped and second keystream is output.
At time $t = 3$; index j_1 updated by the following equation;

$$j_1 = j_1 + S_1[i] = 2 + S_1[2] = 2 + 2 = 4$$

Let say that $S_1[4] = C$. Then second and fourth indexes are swapped, and $S_2[C + 2]$ is output as third keystream.
It is obvious that $A \neq C$, since all values in state tables are different. So, if $S_2[A + 2]$ is not swapped at time $t = 2$, then, first and third keystreams are different. Assumptions 2 and 3 assures that $S_2[A + 2]$ is not swapped at time $t = 2$.
Then the probability that first and third output bytes are equal calculated as follows;

$$\begin{aligned}
Pr[Z_1 = Z_3] = Pr[Z_1 = Z_3 | S_1[1] = 2 \cap A \neq 0xff \cap B \neq A + 2] \cdot \\
Pr[S_1[1] = 2 \cap A \neq 0xff \cap B \neq A + 2] \\
+ Pr[Z_1 = Z_3 | S_1[1] \neq 2 \cap A = 0xff \cap B = A + 2] \cdot \\
Pr[S_1[1] \neq 2 \cap A = 0xff \cap B = A + 2] \\
= 0 \cdot P_1 \cdot P_2 \cdot P_3 + 2^{-8} \cdot (1 - P_1 \cdot P_2 \cdot P_3) \\
\simeq 2^{-8} \cdot (1 - 2^{-8.01})
\end{aligned}$$

This probability is considerably smaller than the ideal probability 2^{-8}.

4.1.3 The Attack of Sarkar

In 2013, Sarkar (51) investigated that the second byte of RC4A has a positive bias towards 2. To prove this bias, he uses the following result for RC4;

Theorem 5 ((54)). *Assume initial permutation S is taken uniformly from the set of all possible permutation of the set $0, 1, \ldots, N-1$. Then for the first index toucher t where $t = S[i] + S[j]$, we have*

$$Pr(t = 2) = \frac{2}{N} - \frac{1}{N(N-1)}.$$

Then, he proves the following lemma;

Lemma 1 ((51)). $Pr(Z_1^{(2)} = t_2) \simeq \frac{2}{N}$.

The results of this lemma also discovered in the section 4.3 in Table 4.1. Then, he proves the bias in the second output byte.

Theorem 6 ((51)). *The probability that the second output byte is equal to two is*

$$Pr(Z_1^{(2)} = 2) \simeq \frac{1}{N-1}.$$

Proof.

$$Pr(Z_1^{(2)} = 2) = Pr(Z_1^{(2)} = 2 \wedge t_2 = 2) + \sum_{\substack{x=0 \\ x \neq 2}}^{N-1} Pr(Z_1^{(2)} = 2 \wedge t_2 = x)$$

Then by using Lemma 1 and Theorem 5;

$$\begin{aligned} Pr(Z_1^{(2)} = 2 \wedge t_2 = 2) =& Pr(Z_1^{(2)} = t_2 | t_2 = 2) Pr(t_2 = 2) \\ &\simeq \frac{2}{N} \cdot \frac{2}{N} \\ &= \frac{4}{N^2} \end{aligned}$$

Also for $x \neq 2$,

$$\begin{aligned} Pr(Z_1^{(2)} = 2 \wedge t_2 = x) =& Pr(t_2 = x) \cdot Pr(Z_1^{(2)} = 2 | t_2 = x) \\ &\simeq \frac{1 - \frac{2}{N}}{N-1} \cdot Pr(S_1^{(1)}[x] = 2) \\ &= (\frac{1 - \frac{2}{N}}{N-1})^2 \end{aligned}$$

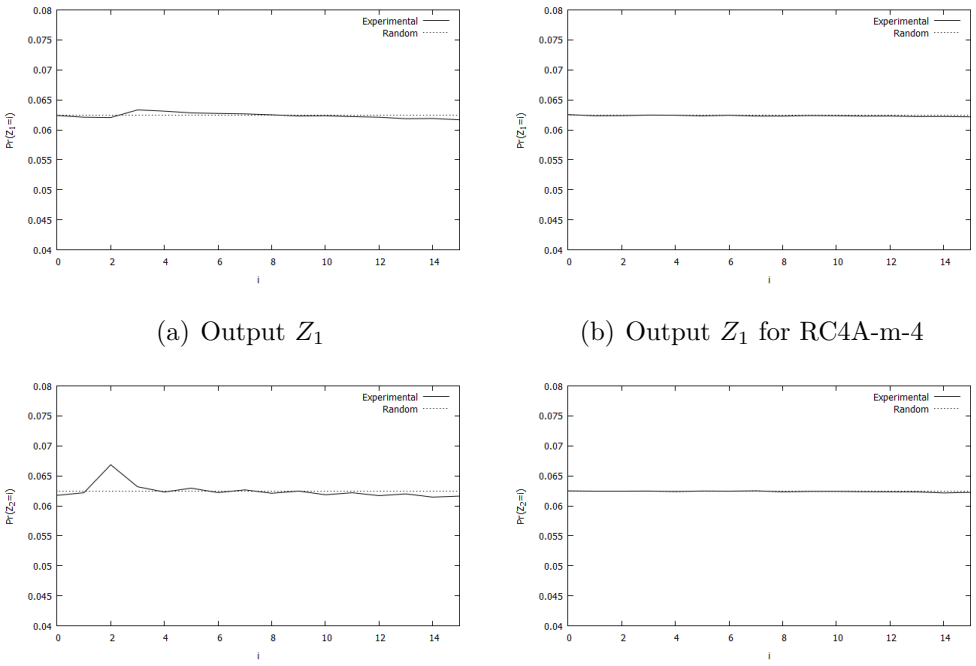
Hence, $Pr(Z_1^{(2)} = 2) \simeq \frac{4}{N^2} + \frac{(1-\frac{2}{N})^2}{N-1} = \frac{1}{N-1}$ □

4.2 Plaintext Recovery Attack Against RC4A

4.2.1 Single Byte Bias Plaintext Recovery Attack

We apply single-byte plaintext recovery attack (3) against RC4A-4 and RC4A-m-4. First of all, by using 2^{28} independent keys, we estimate the probabilities $\{p_{r,k}\}_{1\leq r\leq 16, 0\leq k\leq 16}$.

The bias $Pr(Z_1^{(2)} = 2)$ is experimentally observed in Figure 4.3(c). Keystream output distribution for RC4A-m-4 is more closer to random than RC4A-4 as it is seen in Figure 5.3.



(a) Output Z_1

(b) Output Z_1 for RC4A-m-4

(c) Output Z_2

(d) Output Z_2 for RC4A-m-4

Figure 4.2: Measured distributions of RC4A-4 and RC4A-m-4 keystream outputs Z_1 and Z_2.

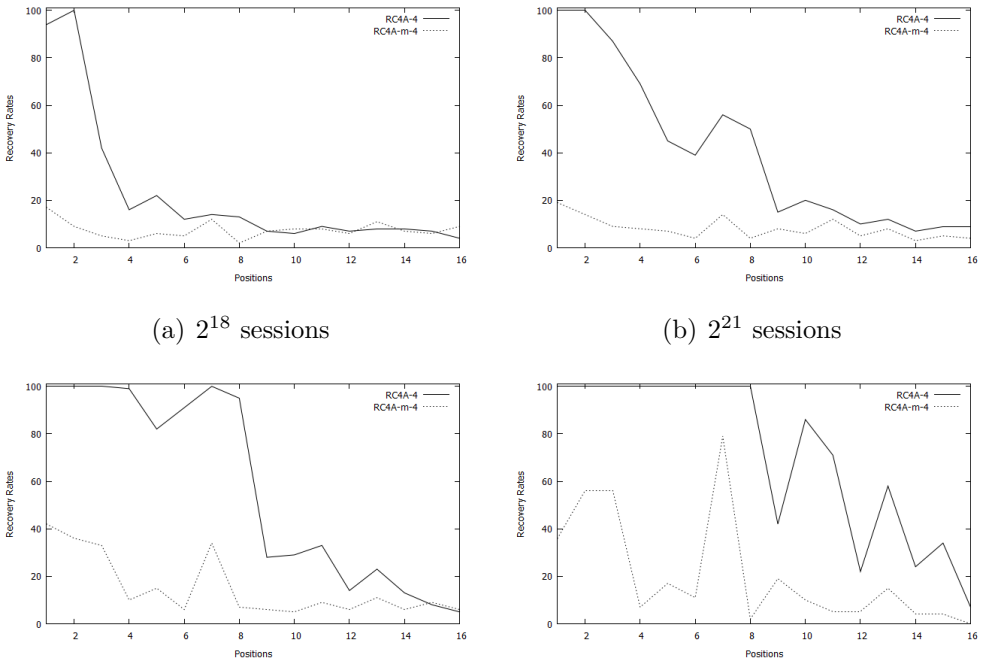
By using keystream distribution, we analyze single-byte plaintext recovery attack under four different session size, $2^{18}, 2^{21}, 2^{24}, 2^{27}$. We run the attack 100 times for RC4A-4 and RC4A-m-4 for each sessions.

- With $S = 2^{18}$ sessions, for RC4A-4, first position is recovered with rate 94% and second position with rate 100%. The bias $Pr(Z_1^{(2)} = 2)$ is the main reason for the high recovery rate of the second position.

- With $S = 2^{21}$ sessions, recovery rates indicates a big increment for RC4A-4 according to first sample. Besides, RC4A-m-4, there is no important change at rates.

- With $S = 2^{24}$ sessions, the first 8 positions are recovered with rates more than 82% for RC4A-4.

- With $S = 2^{27}$ sessions, the first 8 positions are recovered with rates 100% for RC4A-4. There exist a remarkable increasing at recovery rates of the last 8 positions. On the other hand, for RC4A-m-4, increment at recovery rates is observed only for some specific positions.



(a) 2^{18} sessions

(b) 2^{21} sessions

(c) 2^{24} sessions

(d) 2^{27} sessions

Figure 4.3: Recovery rates for RC4A-4 and RC4A-m-4.

4.2.2 Double Byte Bias Plaintext Recovery Attack

From the previous work of Maximov (49),we know that RC4A does not have known weaknesses in the distribution of consecutive keystream outputs. Thus, the double byte bias plaintext recovery attack (3) fails against RC4A. On the other hand, we know that RC4A has some weaknesses in the distribution of consecutive odd keystream outputs from the previous attacks (49; 50). So, to use these weaknesses, we modify attack (3) for RC4A-4 and apply this modified version against RC4A-4 . First of all, by using 2^{28} independent keys, we estimate the probabilities $\{p_{r,k_1,k_2}\}_{1\leq r\leq 16,0\leq k_1\leq 16,0\leq k_2\leq 16}$;

$$p_{r,k_1,k_2} = Pr[(Z_{16q+r}, Z_{16q+r+2}) = (k_1, k_2)]$$

We encrypted $S = 1 \cdot 2^{28}, \ldots, 16 \cdot 2^{28}$ copies of the plaintext with length 16 and assume that first and fifteenth plaintext positions are known and try to get 6 odd positions between them. We apply algorithm 16 times for each session and success rate of algorithm seen in Figure 4.4.

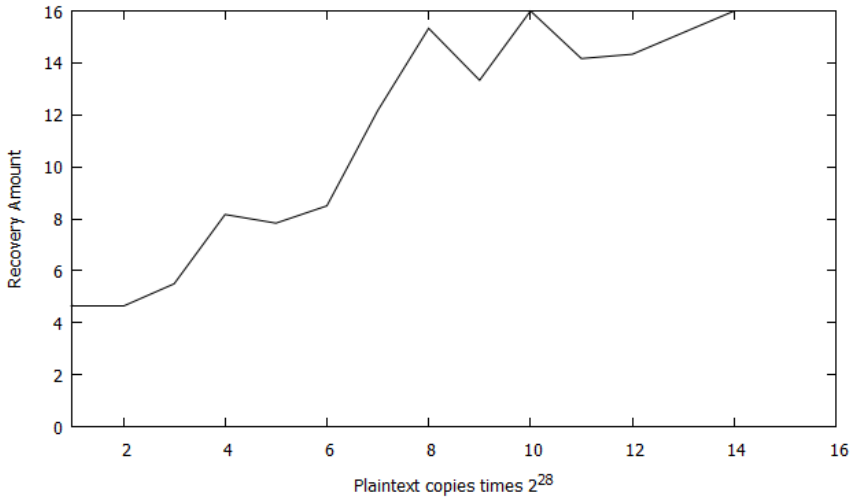


Figure 4.4: Success amount for recovering odd positions for RC4A-4.

4.3 Linear Correlations in PRGA of RC4A

In each round, two keystream ouputs are produced in RC4A differently from RC4. Thus, we analyze the PRGA of RC4A in two parts. In the first part, j_1 and S_1 are updated and first keystream output generated from S_2 by using them. In the second part, j_2 and S_2 are updated and second keystream output generated from S_1 by using them. Thus, we apply linear correlation attack (9) against RC4A by using two equations for these two parts;

$$(c_0^1 \cdot j_i^1 + c_1^1 \cdot S_i^1[i] + c_2^1 \cdot S_i^1[j_i^1] + c_3^1 \cdot z_i^1) \bmod N = C^1$$

$$(c_0^2 \cdot j_i^2 + c_1^2 \cdot S_i^2[i] + c_2^2 \cdot S_i^2[j_i^2] + c_3^2 \cdot z_i^2) \bmod N = C^2$$

We have two equations for two separate parts but generally biased correlations occur in both of them except first round. Thus, we will only list values for the first equation. From the Table 4.1, we confirms that most of the biased correlations that are observed in RC4 are also found in RC4A. However, the bias which is observed by Mantin and Shamir (32) and Jenkins correlations (29) which are observed in RC4 are not observed in RC4A.

j_{1_i}	$S_{1_i}[i]$	$S_{1_i}[j_i]$	$z_i^{(1)}$	C^1	Probability
0	0	1	0	0	$2/N$
⋮	⋮	⋮	⋮	⋮	⋮
0	0	1	0	255	$1.344/N$
0	1	1	-1	0	$2/N$
⋮	⋮	⋮	⋮	⋮	⋮
0	1	1	-1	0	$1.024/N$
1	-1	0	0	0	$1.84/N$
⋮	⋮	⋮	⋮	⋮	⋮
1	-1	0	0	0	$0.98/N$
1	-1	1	0	i	$2/N$
1	1	-1	0	i	$2/N$

Table 4.1: Correlations observed for all rounds.

The Table 4.2 depicts the additional correlations observed in the second round. The number of biased correlations is decreasing according to round number i. We also apply linear correlation attack to RC4A-m. However, we observe that there is no important difference in both number of biases and their degree.

j_{1_i}	$S_{1_i}[i]$	$S_{1_i}[j_i]$	$z_i^{(1)}$	C^1	Probability
1	0	1	0	6	$2.2/N$

Table 4.2: Additional correlation observed in second round.

4.4 Discussion of the Results

The main design principle of RC4A is to weaken the correlations between variables in PRGA by making each keystream output dependent on more random variables. According to the result of linear correlation attack, they could not achieve this objective. Besides, the result of the single byte plaintext recovery attack against RC4A-m indicates that when we initilize internal variable j_1 and j_2 from the values in KSA, algorithm becomes more stronger against single byte biases.

The failure of double byte bias plaintext recovery attack shows that RC4A does not have important long term biases in consecutive keystream outputs. In this attack, we modify attack to use long term biases in consecutive odd keystream outputs for RC4A and succeed to recover odd positions of plaintext.

Chapter 5

Security Analysis of VMPC

We start with previous attacks against VMPC and give a brief summary of works of Maximov (49), Tsunoo (50), Li (52), and Sarkar (51) on VMPC. Then, we apply plaintext recovery attacks and linear correlation attack against VMPC.

5.1 Previous Attacks Against VMPC

5.1.1 The Attack of Maximov

In (49), in 2005, Maximov analyzed the biases in consecutive pairs of bytes, called digraphs on VMPC, and he gave first linear distinguishing attack on VMPC which based on long term digraph biases. He used the approach of Fluhrer and McGrew (4) who investigated first digraph biases in RC4 and used these biases in an efficient distinguishing attack on RC4.

For RC4-3, RC4-4, and RC4-5, Fluhrer and McGrew calculated the theoretical probabilities $Pr\{(Z_t = x, Z_{t+1} = y, i)\}$ for all possible values of triple (x, y, i). Due to the high complexity, for RC4-8, they could only approximate the theoretical probabilities.

In his paper, Maximov showed that the digraph biases is a general problem for

RC4 family of stream cipher, and this problem is need to be considered when designing a new cipher of this kind.
He showed that VMPC has also this kind of vulnerability. Due to the high complexity, he first tried to find any anomalies for VMPC-n like $Pr(Z_t = x, Z_{t+1} = y, i)$ for small number of n. After finding an anomaly for the probability $Pr(Z_t = Z_{t+1} = 0|i = 0)$, he focus on that anomaly and get the following theoretical bias value;

Theorem 7 ((49))**.** *Assume KSA of VMPC performs throughly random permutation which means internal variables S and j are from the uniform distribution. Then;*

$$Pr(Z_t = Z_{t+1} = 0|i = 0) \approx 2^{-16}(1 - 2^{-7.98322})$$

5.1.2 The Attack of Tsunoo et al.

In 2005, Tsunoo et al. (50) investigated the new biased correlation between first and second bytes of VMPC. In this attack, they give two assumptions;

1. $j = 0$, $(P_1 = \frac{1}{256})$

2. $S[A] = 0$ where A denotes the entry of $S[0]$, $(P_2 = \frac{1}{256})$

At time $t = 1$, index j is updated by the following equation;

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

Then first keystream is output by the following equation;

$$Z = S[S[S[j]] + 1] = S[S[S[0]] + 1] = S[S[A] + 1] = S[0 + 1] = S[1] = B$$

Since both index j and i are equal to 0, there will be no swap in this stage.
At time $t = 2$, index j is updated by the following equation;

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

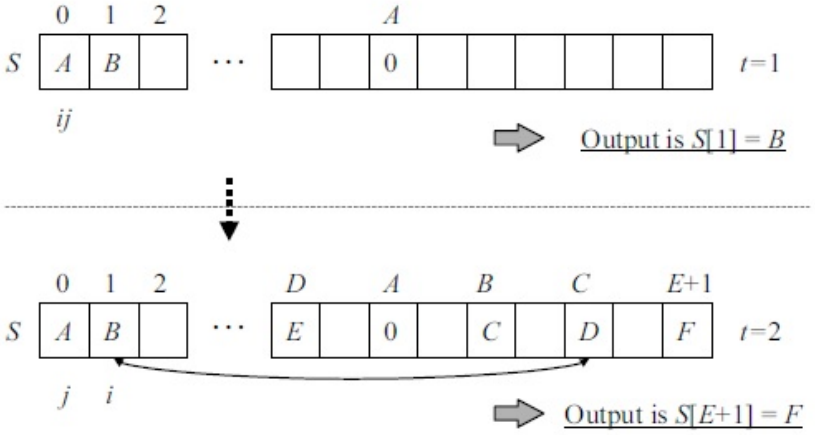


Figure 5.1: PRGA of VMPC and transition of array S.

Then second keystream is output by the following equation;

$$Z = S[S[S[j]] + 1] = S[S[S[C]] + 1] = S[S[D] + 1] = S[E + 1] = F$$

Then index $i = 1$ and index $j = C$ are swapped.
From the construction of array S, we know that the values $A, B, C, D, E,$ and F are all different from each other. We also know that $E \neq 0$ because $E = S[D]$, $D \neq A$, and $S[A] = 0$. Then, B can not be equal to F. Therefore, if two assumptions stated above are satisfied, first two keystream outputs will never be equal.
The probability that first two keystream outputs are equal calculated as follows;

$$\begin{aligned} Pr[Z_1 = Z_2] &= Pr[Z_1 = Z_2 | j = 0 \cap S[A] = 0] \cdot Pr[j = 0 \cap S[A] = 0] \\ &\quad + Pr[Z_1 = Z_2 | j \neq 0 \cap S[A] \neq 0] \cdot Pr[j \neq 0 \cap S[A] \neq 0] \\ &= 0 \cdot P_1 \cdot P_2 + 2^{-8} \cdot (1 - P_1 \cdot P_2) \\ &\simeq 2^{-8} \cdot (1 - 2^{-16}) \end{aligned}$$

This probability is smaller than the ideal probability 2^{-8}.

5.1.3 The Attack of Li et al.

In 2012, Li et al. (52) improved the result observed by Tsunoo et al (50) which is the biased correlation between first and second keystream outputs of VMPC. They give two assumptions;

1. $S[j + S[0]] = 0$, $(P_1 = \frac{1}{256})$
2. $S[S[1]] \neq 0$ $(P_2 = \frac{255}{256})$

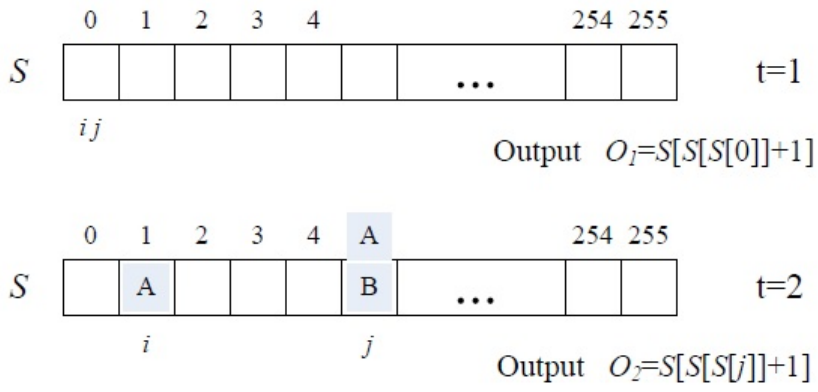


Figure 5.2: The first two keystream outputs of VMPC.

At time $t = 1$, index j is updated by the following equation;

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

Then, the first keystream is output as $S[S[S[0]] + 1]$. There will be no swap, since both index i and j are equal to 0.
At time $t = 2$, index i is updated to 1 and index j is updated by the following equation;

$$j = S[j + S[i]] = S[0 + S[1]] = S[S[1]] = B \neq 0$$

Then, the second keystream is output as $S[S[S[B]] + 1]$. Since entries of array S are not swapped at time $t = 1$ and $B \neq 0$, we can say that first two keystream

outputs are not equal under these two assumptions.
The probability that first two keystream outputs are equal calculated as follows;

$$\begin{aligned}
Pr[Z_1 = Z_2] &= Pr[Z_1 = Z_2 | S[j + S[0]] = 0 \cap S[S[1]] \neq 0] \cdot \\
&\qquad Pr[S[j + S[0]] = 0 \cap S[S[1]] \neq 0] \\
&+ Pr[Z_1 = Z_2 | S[j + S[0]] \neq 0 \cap S[S[1]] = 0] \cdot \\
&\qquad Pr[S[j + S[0]] \neq 0 \cap S[S[1]] = 0] \\
&= 0 \cdot P_1 \cdot P_2 + 2^{-8} \cdot (1 - P_1 \cdot P_2) \\
&\simeq 2^{-8} \cdot (1 - 2^{-8.01})
\end{aligned}$$

This probability is considerably smaller than the ideal probability 2^{-8}.

5.1.4 The Attack of Sarkar

In 2013, Sarkar (51) investigated a new bias in the correlation between second and fourth bytes. First of all, he proves following lemma;

Lemma 2 ((51))**.** *Let j_2, j_3 be the values of j when $i = 2, 3$ respectively. Then*

$$Pr(j_2 = 1 | j_3 = 2) \simeq \frac{2}{N} - \frac{1}{N^2}$$

Proof. j_3 is updated by the equation $j_3 = S_3[j_2 + S_3[3]]$. If $j_2 = 1$ and $S_3[3] = 2$, then j_3 will always be 2. Thus,

$$Pr(j_2 = 1 \wedge j_3 = 2 | S_3[3] = 2) = Pr(j_2 = 1 | S_3[3] = 2) = \frac{1}{N}.$$

If $j_2 = 1$ and $S_3[3] \neq 2$, j_3 can be 2 due to random association. Hence,

$$\begin{aligned}
Pr(j_2 = 1 \wedge j_3 = 2) &= Pr(j_2 = 1 \wedge j_3 = 2 | S_3[3] = 2) \cdot Pr(S_3[3] = 2) \\
&\quad + Pr(j_2 = 1 \wedge j_3 = 2 | S_3[3] \neq 2) \cdot Pr(S_3[3] \neq 2) \\
&= \frac{1}{N} \cdot \frac{1}{N} + \frac{1}{N} \cdot \frac{1}{N} \cdot (1 - \frac{1}{N}) \\
&= \frac{2}{N^2} - \frac{1}{N^3}
\end{aligned}$$

Then $Pr(j_2 = 1 | j_3 = 2) = \frac{Pr(j_2 = 1 \wedge j_3 = 2)}{Pr(j_3 = 2)} = \frac{2}{N} - \frac{1}{N^2}$ □

Then he proves the bias in the correlation between second and fourth output sequences of VMPC.

Theorem 8 ((51))**.** *Assume initial permutation S are taken uniformly from the set of all possible permutation of the set $0, 1, \ldots, N-1$. Also j is taken uniform at random from the set $0, 1, \ldots, N-1$. Then the probability $Pr(Z_1 = Z_3) \simeq \frac{1}{N} + \frac{1}{N^2}$, where Z_1, Z_3 are keystreams when $i = 1$ and 3 respectively.*

Proof. He considers three cases;
Case 1. $Pr(Z_1 = Z_3 | j_3 = 1)$: Consider the event

$$E = \Big\{ S_1[S_1[j_1]]+1 \notin \{j_1, j_2\}, S_1[S_1[j_1]]+1 \notin \{1, 2\}, S_1[j_1] \notin \{j_1, j_2\}, S_1[j_1] \notin \{1, 2\} \Big\}$$

There are 8 conditions in the event E. Now consider the event $E' = E \wedge \{j_2 \neq 1\}$. If $j_3 = 1$ holds with the event E', then Z_1 will always be equal to Z_3. Since, we have 9 conditions in the event E', $Pr(E') = 1 - \frac{1}{9}$. When E' does not hold together with $j_3 = 1$, Z_3 can be equal to Z_1 due to random association. Hence;

$$\begin{aligned} Pr(Z_1 = Z_3 | j_3 = 1) &= Pr(Z_1 = Z_3 | j_3 = 1, E') \cdot Pr(E') \\ &\quad + Pr(Z_1 = Z_3 | j_3 = 1, E'^c) \cdot Pr(E'^c) \\ &\approx 1.\Big(1 - \frac{9}{N}\Big) + \frac{1}{N} \cdot \frac{9}{N} \\ &= 1 - \frac{9}{N} + \frac{9}{N^2} = p_1. \end{aligned}$$

Case 2. $Pr(Z_1 = Z_3 | j_3 = 2)$: We have

$$\begin{aligned} Pr(Z_1 = Z_3 \wedge j_3 = 2) &= Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge E \wedge j_2 = 1) \\ &\quad + Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge (E^c \cup j_2 \neq 1)) \end{aligned}$$

and

$$\begin{aligned} Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge E \wedge j_2 = 1) &= Pr(Z_1 = Z_3 | j_3 = 2 \wedge E \wedge j_2 = 1) \cdot \\ &\qquad Pr(j_3 = 2 \wedge E \wedge j_2 = 1) \\ &\approx Pr(Z_1 = Z_3 | j_3 = 2 \wedge E \wedge j_2 = 1) \cdot \\ &\qquad Pr(E) \cdot Pr(j_3 = 2 \wedge j_2 = 1) \\ &= Pr(Z_1 = Z_3 | j_3 = 2 \wedge E \wedge j_2 = 1) \cdot \\ &\qquad Pr(E) \cdot Pr(j_2 = 1 | j_3 = 2) \cdot Pr(j_3 = 2) \end{aligned}$$

Z_3 will be equal to Z_1 when the event E holds with the conditions $j_3 = 2$ and $j_2 = 1$. Since we have 8 conditions in event E, E holds with probability $1 - \frac{8}{N}$. By using lemma above, we have;

$$Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge E \wedge j_2 = 1) = 1 \cdot \left(1 - \frac{8}{N}\right) \cdot \left(\frac{2}{N} - \frac{1}{N^2}\right) \cdot \frac{1}{N}$$

Z_3 will be always different to Z_1 when the event E holds with the conditions $j_3 = 2$ and $j_2 \neq 1$. As a consequence, $Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge (E^c \cup j_2 \neq 1)) = Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge E^c)$. When E does not hold, Z_3 can be equal to Z_1 due to random association. Thus,

$$\begin{aligned} Pr(Z_1 = Z_3 \wedge j_3 = 2 \wedge E^c) &= Pr(Z_1 = Z_3) \cdot Pr(j_3 = 2) \cdot Pr(E^c) \\ &= \frac{1}{N} \cdot \frac{1}{N} \cdot \frac{8}{N} \end{aligned}$$

So,

$$\begin{aligned} Pr(Z_1 = Z_3 | j_3 = 2) &= Pr(Z_1 = Z_3 \wedge j_3 = 2) \cdot Pr(j_3 = 2) \\ &= \left(\left(1 - \frac{8}{N}\right) \cdot \left(\frac{2}{N} - \frac{1}{N^2}\right) \cdot \frac{1}{N} + \frac{8}{N^3}\right) \cdot \frac{1}{N} \\ &= \left(1 - \frac{8}{N}\right) \cdot \left(\frac{2}{N} - \frac{1}{N^2}\right) + \frac{8}{N^2} = p_2 \end{aligned}$$

Case 3. $Pr(Z_1 = Z_3 | j_3 \neq 1, 2)$:

$$\begin{aligned} Pr(Z_1 = Z_3 | j_3 = x) &= Pr(E^c) \cdot Pr(Z_1 = Z_3 | j_3 = x, E^c) \\ &= Pr(E^c) \cdot \frac{1}{N} = \frac{8}{N} \cdot \frac{1}{N} \\ &= \frac{8}{N^2} = p_3 \end{aligned}$$

hence,

$$\begin{aligned} Pr(Z_1 = Z_3) &= Pr(Z_1 = Z_3 | j_3 = 1) Pr(j_3 = 1) + Pr(Z_1 = Z_3 | j_3 = 2) \cdot Pr(j_3 = 2) \\ &= \sum_{\substack{x=0 \\ x \notin \{1,2\}}}^{N-1} Pr(Z_1 = Z_3 | j_3 = x) \cdot Pr(j_3 = x) = \frac{p_1}{N} + \frac{p_2}{N} + p_3\left(1 - \frac{2}{N}\right) \\ &= \frac{1}{N} + \frac{1}{N^2} - \frac{16}{N^3} + \frac{8}{N^4} \approx \frac{1}{N} + \frac{1}{N^2} \end{aligned}$$

□

By using the same approach, he gets the most significant long term bias on VMPC.

Conjecture 1 ((51))**.** *Assume that the initial permutation S is taken uniformly from the set of all possible permutations of the set $\{0, 1, ..., N-1\}$. Also j is taken uniformly at random from the set $\{0, 1, ..., N-1\}$. Then the probability $Pr(Z_{kN+1} = Z_{kN+3}) \approx \frac{1}{N} + \frac{1}{N^2}$, where Z_ℓ is the $\ell + 1^{th}$ keystream byte and k is a non-negative integer.*

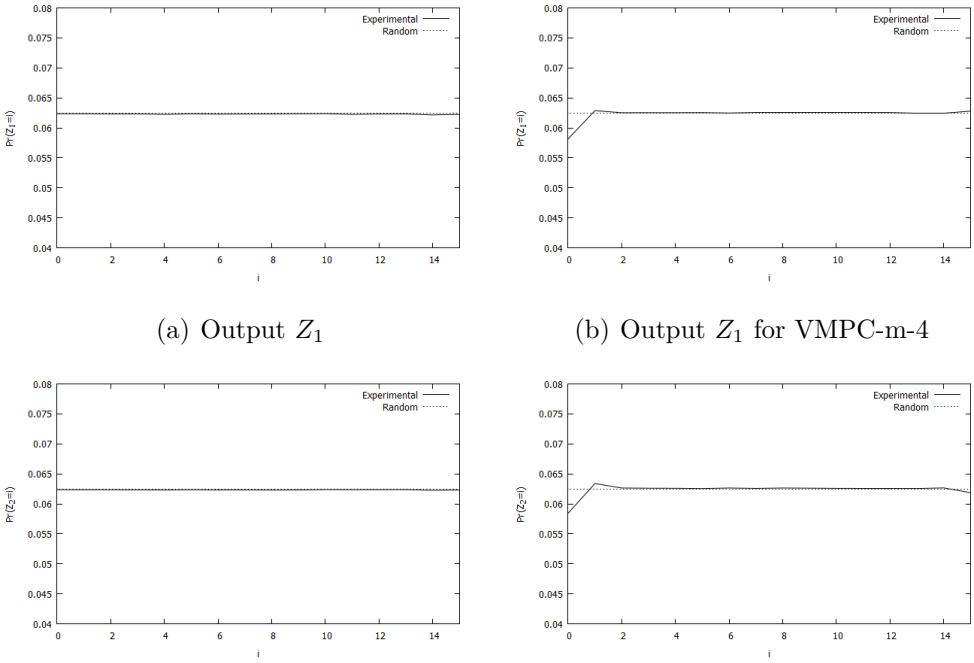
By using 1 billion sample size, he also experimentally shows that $Pr(Z_{kN+1} = Z_{kN+3}) = 0.003921 (\approx \frac{1}{N} + \frac{1}{N^2})$.

5.2 Plaintext Recovery Attack Against VMPC

5.2.1 Single Byte Bias Plaintext Recovery Attack

We apply single-byte plaintext recovery attack (3) against VMPC-4 and VMPC-m-4. First of all, by using 2^{28} independent keys, we estimate the probabilities $\{p_{r,k}\}_{1 \leq r \leq 16, 0 \leq k \leq 16}$.

Keystream output distribution for VMPC-4 is very closed to random as it is seen in Figure 5.3. Thus, we can say that there is no important short term biases for VMPC-4. On the other hand, for VMPC-m-4, some biases observed in keystream outputs. Especially, in first three keystream outputs, there exists strong negative biases towards zero.



(a) Output Z_1

(b) Output Z_1 for VMPC-m-4

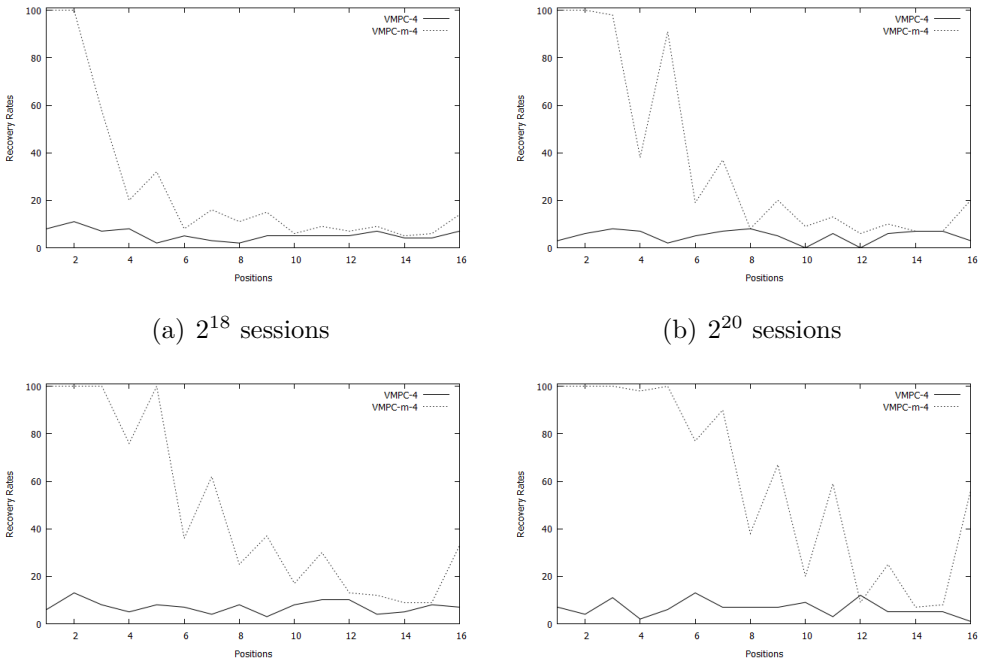
(c) Output Z_2

(d) Output Z_2 for VMPC-m-4

Figure 5.3: Measured distributions of VMPC-4 and VMPC-m-4 keystream outputs Z_1 and Z_2.

By using keystream distribution, we analyze single-byte plaintext recovery attack under four different session size, $2^{18}, 2^{20}, 2^{22}, 2^{24}$. We run the attack 100 times for VMPC-4 and VMPC-m-4 for each sessions.

As it is seen in Figure 5.4, recovery rates for VMPC-4 are not increase with sample size. It is very closed to random estimation. On the other hand, for VMPC-m-4, recovery rate is increase with sample size. Especially, in odd positions, rates are higher than even ones.



(a) 2^{18} sessions

(b) 2^{20} sessions

(c) 2^{22} sessions

(d) 2^{24} sessions

Figure 5.4: Recovery rate of the single-byte bias attack against VMPC-4 and VMPC-m-4 for $S = 2^{18}, 2^{20}, 2^{22}$ and 2^{24} sessions for the first 16 positions of plaintext.

5.2.2 Double Byte Bias Plaintext Recovery Attack

We know that VMPC has some weaknesses in the distribution of consecutive keystream outputs from the previous attacks of Maximov, Tsunoo, and Li (49; 50; 52). By using this weakness, we apply double-byte plaintext recovery attack (3) against VMPC-4 . First of all, by using 2^{28} independent keys, we estimate the probabilities $\{p_{r,k_1,k_2}\}_{1\leq r\leq 16,0\leq k_1\leq 16,0\leq k_2\leq 16}$.

Then, we encrypt $S = 1 \cdot 2^{24}, \ldots, 16 \cdot 2^{24}$ copies of the plaintext with length 16 and assume that first and last plaintext positions are known and try to get 14 positions between them. We apply algorithm 100 times for each session and success rate of algorithm seen in Figure 5.5.

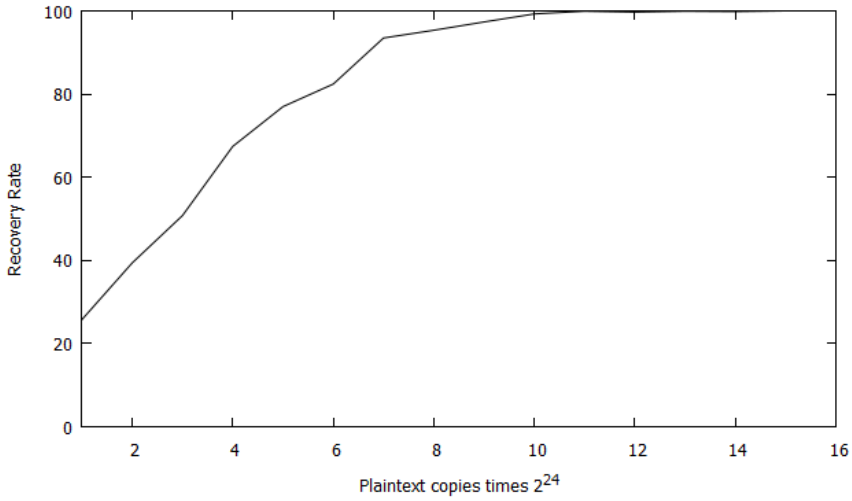


Figure 5.5: Success rate for recovering all positions for VMPC-4.

We also observe following anomaly when we experimentally analyze the probabilities;

Conjecture 2. $Pr(Z_{kN+1} = Z_{kN+2}) \approx 2^{-4} \cdot (1 - 2^{-6.4})$

5.3 Linear Correlations in PRGA of VMPC

In each round of PRGA of VMPC, keystream outputs are generated by using random variables $i, j_i, S_i[i], S_i[j_i]$. Thus, we apply linear correlation attack (9) by using the following equation;

$$(c_o \cdot j_i + c_1 \cdot S_i[i] + c_2 \cdot S_i[j_i] + c_3 \cdot z_i) \bmod N = C$$

We observe four previously undiscovered biased linear correlations in VMPC which occurs in each round. These correlations are listed in Table 5.1.

j_i	$S_i[i]$	$S_i[j_i]$	z_i	C	Probability
1	1	-1	0	i	$2/N$
1	-1	1	0	i	$2/N$
1	-1	0	1	$i-1$	$1.13/N$
0	0	1	-1	1	$2/N$

Table 5.1: Correlations observed for all rounds.

On the other hand, when we apply linear correlation attack to VMPC-m, we observe extra biased correlations in addition to these four biases. Especially, in the first initial rounds, there are a large number of biased correlations. For the first round, additional biased correlations are seen in Table 5.2.

j_i	$S_i[i]$	$S_i[j_i]$	z_i	C	Probability
0	0	1	0	0	$2/N$
0	0	1	1	1	$2/N$
0	1	-1	0	0	$2/N$
0	1	1	0	0	$2/N$
1	-1	-1	0	0	$2/N$
1	0	0	0	0	$2/N$
1	0	1	0	0	$2/N$
1	1	0	0	0	$2/N$
1	1	1	0	0	$2/N$

Table 5.2: Additional correlations observed in the first round for VMPC-m.

5.4 Discussion of the Results

The results of single byte plaintext recovery attack and linear correlation attack indicate that VMPC is a strong cipher against short term biases. On the other hand, the result of double byte plaintext recovery attack shows that the structure of VMPC is weak against biases in consecutive keystream outputs.

Additionally, the results of attacks against VMPC-m show that initialization of internal variable j in PRGA is an important factor for the security of VMPC.

Chapter 6

Conclusion

In this thesis, we explain two kind of attacks; single and double byte plaintext recovery attacks and linear correlation attack on RC4 and we apply these attacks against two variants of RC4; RC4A and VMPC.

Single byte plaintext recovery attack and linear correlation attack mainly depend on short term biases in ciphers. Experimental results indicate that among these three ciphers, VMPC is the strongest one against these attacks. When we apply the single byte plaintext recovery attack against VMPC, we can not recover any position with rate higher than random association. Also, there is no important linear correlation in VMPC except four uniform correlations which exist in all rounds. The main difference of VMPC is the initialization of variable j in PRGA. In VMPC, when we initialize the variable j from 0 as in the RC4 and RC4A, we observe that VMPC becomes vulnerable against these attacks.

By using this observation, we modified RC4 and RC4A by initializing internal variables from the values in KSA and applied these attacks to modified versions. We observe that modified versions of the ciphers are stronger against single byte plaintext recovery attack and linear correlation attack than original ones.

Double byte plaintext recovery attack depends on long term consecutive multi byte biases. We have known that RC4 is vulnerable against double byte plaintext

recovery attack. When we apply this attack on VMPC, we observe that it is also vulnerable against this attack. The structure of RC4A makes it stronger against this attack. However, when we consider multi byte biases which occurs consecutive odd positions in RC4A and modify the double byte plaintext recovery attack according to these biases, we achieve to recover plaintext positions for RC4A.

We discover some new biased correlations in VMPC by applying linear correlation attack against it. The proofs of these correlations are left for future work.

Bibliography

[1] S. Paul and B. Preneel, "A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher," in *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, pp. 245–259, 2004.

[2] B. Zoltak, "VMPC one-way function and stream cipher," in *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, pp. 210–225, 2004.

[3] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt, "On the security of RC4 in TLS," in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pp. 305–320, 2013.

[4] S. R. Fluhrer and D. A. McGrew, "Statistical analysis of the alleged RC4 keystream generator," in *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, pp. 19–30, 2000.

[5] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. CRC Press, Inc., 1st ed., 1996.

[6] C. D. Canniere and B. Preneel, "Trivium specifications," *eSTREAM, ECRYPT Stream Cipher Project*, vol. 2006, 2006.

[7] S. O'Neil, B. Gittins, and H. A. Landman, "Vest hardware-dedicated stream ciphers.," *IACR Cryptology ePrint Archive*, vol. 2005, p. 413, 2005.

[8] P. Rogaway and D. Coppersmith, "A software-optimized encryption algorithm," *J. Cryptology*, vol. 11, no. 4, pp. 273–287, 1998.

[9] P. Sepehrdad, S. Vaudenay, and M. Vuagnoux, "Discovery and exploitation of new biases in RC4," in *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, pp. 74–91, 2010.

[10] S. S. Gupta, S. Maitra, G. Paul, and S. Sarkar, "(Non-)random sequences from (non-)random permutations - analysis of RC4 stream cipher," *J. Cryptology*, vol. 27, no. 1, pp. 67–108, 2014.

[11] A. Grosul and D. S. Wallach, "A related-key cryptanalysis of RC4," Technical Report TR01-380, Department of Computer Science, Rice University, 2000.

[12] E. Biham and O. Dunkelman, "Differential cryptanalysis in stream ciphers," *IACR Cryptology ePrint Archive*, vol. 2007, p. 218, 2007.

[13] M. Matsui, "Key collisions of the RC4 stream cipher," in *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, pp. 38–50, 2009.

[14] J. Chen and A. Miyaji, "How to find short RC4 colliding key pairs," in *Information Security, 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings*, pp. 32–46, 2011.

[15] S. Maitra, G. Paul, S. Sarkar, M. Lehmann, and W. Meier, "New results on generalization of roos-type biases and related keystreams of RC4," in *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings*, pp. 222–239, 2013.

[16] G. Paul and S. Maitra, "Permutation after RC4 key scheduling reveals the secret key," in *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, pp. 360–377, 2007.

[17] E. Biham and Y. Carmeli, "Efficient reconstruction of RC4 keys from internal states," in *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pp. 270–288, 2008.

[18] M. Akgün, P. Kavak, and H. Demirci, "New results on the key scheduling algorithm of RC4," in *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, pp. 40–52, 2008.

[19] S. Khazaei and W. Meier, "On reconstruction of RC4 keys from internal states," in *Mathematical Methods in Computer Science, MMICS 2008, Karlsruhe, Germany, December 17-19, 2008 - Essays in Memory of Thomas Beth*, pp. 179–189, 2008.

[20] R. Basu, S. Maitra, G. Paul, and T. Talukdar, "On some sequences of the secret pseudo-random index j in RC4 key scheduling," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 18th International Symposium, AAECC-18 2009, Tarragona, Catalonia, Spain, June 8-12, 2009. Proceedings*, pp. 137–148, 2009.

[21] A. Roos, "A class of weak keys in the RC4 stream cipher," 1995. Available online at `http://www.impic.org/papers/WeakKeys-report.pdf`.

[22] D. A. Wagner, "My RC4 weak keys," 1995. Available online at `http://www.cs.berkeley.edu/~daw/my-posts/my-rc4-weak-keys`.

[23] L. R. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, "Analysis methods for (alleged) RC4," in *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, pp. 327–341, 1998.

[24] J. D. Golic, "Iterative probabilistic cryptanalysis of RC4 keystream generator," in *Information Security and Privacy, 5th Australasian Conference, ACISP 2000, Brisbane, Australia, July 10-12, 2000, Proceedings*, pp. 220–233, 2000.

[25] T. Ohigashi, Y. Shiraishi, and M. Morii, "An improved internal-state reconstruction method of a stream cipher RC4," *Communication, Network, and Information Security*, 2003.

[26] V. Tomasevic, S. Bojanic, and O. Nieto-Taladriz, "Finding an internal state of RC4 stream cipher," *Inf. Sci.*, vol. 177, no. 7, pp. 1715–1727, 2007.

[27] A. Maximov and D. Khovratovich, "New state recovery attack on RC4," in *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pp. 297–316, 2008.

[28] J. D. Golic and G. Morgari, "Iterative probabilistic reconstruction of RC4 internal states," *IACR Cryptology ePrint Archive*, vol. 2008, p. 348, 2008.

[29] R. J. Jenkins Jr., "ISAAC and RC4," 1996. Published on the Internet at `http://burtleburtle.net/bob/rand/isaac.html`.

[30] Itsik Mantin, "Analysis of the stream cipher RC4," Master's thesis, The Weizmann Institute of Science, 2001.

[31] S. R. Fluhrer and D. A. McGrew, "Statistical analysis of the alleged RC4 keystream generator," in *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, pp. 19–30, 2000.

[32] I. Mantin and A. Shamir, "A practical attack on broadcast RC4," in *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, pp. 152–164, 2001.

[33] I. Mironov, "(Not so) random shuffles of RC4," in *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pp. 304–319, 2002.

[34] I. Mantin, "Predicting and distinguishing attacks on RC4 keystream generator," in *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pp. 491–506, 2005.

[35] R. Basu, S. Ganguly, S. Maitra, and G. Paul, "A complete characterization of the evolution of RC4 pseudo random generation algorithm," *J. Mathematical Cryptology*, vol. 2, no. 3, pp. 257–289, 2008.

[36] S. S. Gupta, S. Maitra, G. Paul, and S. Sarkar, "Proof of empirical RC4 biases and new key correlations," in *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pp. 151–168, 2011.

[37] S. Maitra, G. Paul, and S. Sengupta, "Attack on broadcast RC4 revisited," in *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, pp. 199–217, 2011.

[38] T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii, "Full plaintext recovery attack on broadcast RC4," in *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pp. 179–202, 2013.

[39] S. Sarkar, S. S. Gupta, G. Paul, and S. Maitra, "Proving tls-attack related open biases of RC4," *IACR Cryptology ePrint Archive*, vol. 2013, p. 502, 2013.

[40] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, pp. 1–24, 2001.

[41] S. S. Gupta, *Analysis and Implementation of RC4 Stream Cipher*. PhD thesis, Indian Statistical Institute, 2013. Available online at `http://souravsengupta.com/pub/phd_thesis_2013.pdf`.

[42] A. Klein, "Attacks on the RC4 stream cipher," *Designs, Codes and Cryptography*, vol. 48, no. 3, pp. 269–286, 2008.

[43] P. Sepehrdad, S. Vaudenay, and M. Vuagnoux, "Statistical attack on RC4 - distinguishing WPA," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pp. 343–363, 2011.

[44] E. Tews, R. Weinmann, and A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds," *IACR Cryptology ePrint Archive*, vol. 2007, p. 120, 2007.

[45] S. Vaudenay and M. Vuagnoux, "Passive-only key recovery attacks on RC4," in *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, pp. 344–359, 2007.

[46] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proceedings of the Second ACM Conference on Wireless Network Security, WISEC 2009, Zurich, Switzerland, March 16-19, 2009*, pp. 79–86, 2009.

[47] V. Moen, H. Raddum, and K. J. Hole, "Weaknesses in the temporal key hash of WPA," *Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 76–83, 2004.

[48] P. Sepehrdad, *Statistical and Algebraic Cryptanalysis of Lightweight and Ultra-Lightweight Symmetric Primitives*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2012. Available online at `http://lasecwww.epfl.ch/~sepehrdad/Pouyan_Sepehrdad_PhD_Thesis.pdf`.

[49] A. Maximov, "Two linear distinguishing attacks on VMPC and RC4A and weakness of RC4 family of stream ciphers," in *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, pp. 342–358, 2005.

[50] Y. Tsunoo, T. Saito, H. Kubo, M. Shigeri, T. Suzaki, and T. Kawabata, "The most efficient distinguishing attack on VMPC and RC4A," 2005.

[51] S. Sarkar, "Further non-randomness in rc4, RC4A and VMPC," *Cryptography and Communications*, vol. 7, no. 3, pp. 317–330, 2015.

[52] S. Li, Y. Hu, Y. Zhao, and Y. Wang, "Improved cryptanalysis of the VMPC stream cipher," *Journal of Computational Information Systems*, vol. 8, no. 2, pp. 831– 838, 2012.

[53] G. Paul, S. Rathi, and S. Maitra, "On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key," *Des. Codes Cryptography*, vol. 49, no. 1-3, pp. 123–134, 2008.

[54] S. Maitra and G. Paul, "New form of permutation bias and secret key leakage in keystream bytes of RC4," in *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pp. 253–269, 2008.