How to Recover Any Byte of Plaintext on RC4

Toshihiro Ohigashi[1(✉)], Takanori Isobe[2], Yuhei Watanabe[2], and Masakatu Morii[2]

[1] Hiroshima University, 1-4-2 Kagamiyama, Higashi-Hiroshima, Hiroshima 739-8511, Japan
ohigashi@hiroshima-u.ac.jp
[2] Kobe University, 1-1 Rokkoudai, Nada-ku, Kobe 657-8501, Japan
Takanori.Isobe@jp.sony.com
yuheiwatanabe@stu.kobe-u.ac.jp
mmorii@kobe-u.ac.jp

Abstract. In FSE 2013, Isobe et al. proposed efficient plaintext recovery attacks on RC4 in the broadcast setting where the same plaintext is encrypted with different user keys. Their attack is able to recover first 1000 terabytes of a plaintext with probability of almost one, given 2^{34} ciphertexts encrypted by different keys. Since their attack essentially exploits biases in the initial (1st to 257th) bytes of the keystream, it does not work any more if such initial bytes are disregarded. This paper proposes two advanced plaintext recovery attacks that can recover *any* byte of a plaintext without relying on initial biases, i.e., our attacks are feasible even if initial bytes of the keystream are disregarded. The first attack is the modified Isobe et al.'s attack. Using the partial knowledge of the target plaintext, e.g., only 6 bytes of the plaintext, the other bytes can be recovered with the high probability from 2^{34} ciphertexts. The second attack does not require any previous knowledge of a plaintext. In order to achieve it, we develop a *guess-and-determine* plaintext recovery method based on two strong long-term biases. Given 2^{35} ciphertexts, any byte of a plaintext can be recovered with probability close to one.

Keywords: RC4 · Broadcast setting · Plaintext recovery attack · Bias · Guess-and-determine attack · Multi-session setting · RC4-drop

1 Introduction

RC4, designed by Rivest in 1987, is one of most widely used stream ciphers in the world. It is adopted in many software applications and standard protocols such as SSL/TLS, WEP, Microsoft Lotus and Oracle secure SQL. RC4 consists of a key scheduling algorithm (KSA) and a pseudo-random generation algorithm (PRGA). The KSA converts a user-provided variable-length key (typically, 5–32 bytes) into an initial state S consisting of a permutation of $\{0, 1, 2, \ldots, N-1\}$, where N is typically 256. The PRGA generates a keystream Z_1, Z_2, ..., Z_r, ... from S, where r is a round number of the PRGA. Z_r is XOR-ed with the

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 155–173, 2014.
DOI: 10.1007/978-3-662-43414-7_8,

r-th plaintext byte P_r to obtain the ciphertext byte C_r. The algorithm of RC4 is shown in Algorithm 1, where + denotes arithmetic addition modulo N, ℓ is the key length, and i and j are used to point to the locations of S, respectively. Then, $S[x]$ denotes the value of S indexed x.

In FSE 2001, Mantin and Shamir proposed a plaintext recovery attack on RC4 in the broadcast setting where the same plaintext is encrypted with different user keys [12]. Using a bias of Z_2, a second byte of the plaintext is recovered from $\Omega(N)$ ciphertexts encrypted with randomly-chosen different keys. In FSE 2011, Maitra, Paul and Sen Gupta showed that $Z_3, Z_4, \ldots, Z_{255}$ are also biased to 0 [9]. The bytes 3 to 255 are also obtained in the broadcast setting, from $\Omega(N^3)$ ciphertexts. In FSE 2013, Isobe et al. introduced several new biases in the initial bytes of the RC4 keystream, and constructed a cumulative list of strong biases in the first 257 bytes with theoretical reasons [7]. They demonstrated plaintext recovery attacks using their strong biases set with typical parameters of $N = 256$ and $\ell = 16$ (128-bit key). 2^{32} ciphertexts encrypting the same plaintext enable to extract first 257 bytes of a plaintext with probability more than 0.8. Using these initial biases in conjunction with the digraph repetition bias proposed by Mantin in EUROCRYPT 2005 [11], the consecutive first 1000 terabytes of a plaintext is theoretically recovered with probability of almost one from 2^{34} ciphertexts encrypted by different keys. After that, AlFardan et al. also proposed similar plaintext recovery attack of the first 256 bytes [1] independently of [7], and this attack can recover first 256 bytes of a plaintext with probability more than 0.96 from 2^{32} ciphertexts encrypted by different keys. Note that broadcast attacks [1,7] can be converted into the attacks for the multi-session setting of SSL/TLS where the target plaintext blocks are repeatedly sent in the same position in the plaintexts in multiple sessions [3].

Previous plaintext recovery attacks essentially exploit biases in the 1st to 257th bytes of the keystream. If the initial 256/512/768 bytes of the keystream are disregarded, as recommended in case of RC4 usages, it does not work any more as mentioned in [7]. Thus, RC4 that disregards the first n bytes of a keystream seem to be secure against above attacks for $n > 257$.

This paper proposes two advanced plaintext recovery attacks that can recover *any* byte of a plaintext without relying on initial biases of the keystream, i.e., our attacks are feasible even if initial bytes of the keystream are disregarded, unlike Isobe et al. and AlFardan et al.'s attacks. To begin with, we improve Isobe et al.'s attack so that it works without initial biases of a keystream. In particular, we assume that an attacker knows some bytes of the target plaintext, e.g., fixed header information. Using the digraph repetition biases in forward and backward manners, the other bytes of the plaintext are recovered from the partial knowledge of the plaintext. In our attack, if only consecutive 6 bytes of the target plaintexts are known, 1000 terabytes of the target plaintext can be recovered with probability of about 0.636 from 2^{34} ciphertexts. The number of required ciphertexts of this attack is same as that of Isobe et al.'s attack, while Isobe et al.'s attack needs the initial 257 bytes of the keystream. The second attack does not require any previous knowledge of a plaintext. In order to achieve

Algorithm 1. RC4 Algorithm

KSA($K[0 \ldots \ell - 1]$)**:**
  for $i = 0$ to $N - 1$ **do**
    $S[i] \leftarrow i$
  end for
  $j \leftarrow 0$
  for $i = 0$ to $N - 1$ **do**
    $j \leftarrow j + S[i] + K[i \bmod \ell]$
    Swap $S[i]$ and $S[j]$
  end for

PRGA(K)**:**
  $i \leftarrow 0$
  $j \leftarrow 0$
  $S \leftarrow KSA(K)$
  loop
    $i \leftarrow i + 1$
    $j \leftarrow j + S[i]$
    Swap $S[i]$ and $S[j]$
    Output $Z \leftarrow S[S[i] + S[j]]$
  end loop

it, we develop a novel *guess-and-determine* plaintext recovery method based on two strong long-term biases, i.e., digraph repetition biases [11] and Fluhrer-McGrew biases [4]. The basic idea behind our guess-and-determine attack is that two biases are used for the detection the wrong candidates of plaintext bytes. Given 2^{35} ciphertext encrypted by different keys, any byte of a plaintext can be recovered with probability close to one[1].

We emphasize that our attacks are applicable even if any number of initial bytes of the keystream are disregarded, with almost same amount of ciphertexts as Isobe et al.'s attack. Therefore, our work reveals that the RC4 implementation that disregards the first n bytes of a keystream is also not secure even if n is enough large (e.g. $n = 3072$).

2 Preliminary

In this section, we introduce two known long-term biases, which occur in any keystream bytes, because our attacks are based on them. Then we describe previous plaintext recovery attacks on RC4 in the broadcast setting.

2.1 Long-term Bias

As a long-term bias, following two types of biases were proposed.

Bias of Digraph Probabilities (FM00 Bias). Fluhrer and McGrew showed a long-term bias of digraph probabilities in the RC4 keystream, called the *FM00 bias*. It is a bias of 2-byte word of the keystream with the condition of index i ($= r \bmod N$) [4], and consists of 12 positive or negative events. The detail of the FM00 bias is shown in Table 1.

[1] Independently of our work, other plaintext recovery attacks on RC4 implementation which disregards the first n bytes of a keystream, was recently reported in [1,2]. The attack uses only the Fluhrer-McGrew biases with the sophisticated count-up method, and obtains experimental results similar to that of our attack.

Table 1. Events of the FM00 bias with the condition of index i ($= r \bmod N$)

Condition of event	Digraph (Z_r, Z_{r+1})	$\Pr(Z_r \wedge Z_{r+1})$
$i = 1$	$(0, 0)$	$N^{-2} \cdot (1 + 2 \cdot N^{-1})$
$i \neq 1, N-1$	$(0, 0)$	$N^{-2} \cdot (1 + N^{-1})$
$i \neq 0, 1$	$(0, 1)$	$N^{-2} \cdot (1 + N^{-1})$
$i \neq N-2$	$(i+1, N-1)$	$N^{-2} \cdot (1 + N^{-1})$
$i \neq 1, N-2$	$(N-1, i+1)$	$N^{-2} \cdot (1 + N^{-1})$
$i \neq 0, N-3, N-2, N-1$	$(N-1, i+2)$	$N^{-2} \cdot (1 + N^{-1})$
$i = N-2$	$(N-1, 0)$	$N^{-2} \cdot (1 + N^{-1})$
$i = N-1$	$(N-1, 1)$	$N^{-2} \cdot (1 + N^{-1})$
$i = 0, 1$	$(N-1, 2)$	$N^{-2} \cdot (1 + N^{-1})$
$i = 2$	$(N/2+1, N/2+1)$	$N^{-2} \cdot (1 + N^{-1})$
$i \neq N-2$	$(N-1, N-1)$	$N^{-2} \cdot (1 - N^{-1})$
$i \neq 0, N-1$	$(0, i+1)$	$N^{-2} \cdot (1 - N^{-1})$

The Digraph Repetition Bias (*ABSAB* Bias). Mantin found another long-term bias of digraph distribution in the RC4 keystream [11], called the *ABSAB bias*. Assuming A and B are two words of the keystream, the digraph AB tends to repeat with short gaps S between them, e.g., $ABAB$, $ABCAB$ and $ABCDAB$, where gap S is defined as zero, C, and CD, respectively. The detail of the $ABSAB$ bias is as follows,

$$Z_r \,||\, Z_{r+1} = Z_{r+2+G} \,||\, Z_{r+3+G} \quad \text{for } G \geq 0, \tag{1}$$

where $||$ is a concatenation. The probability that Eq. (1) holds is given as Theorem 1.

Theorem 1 [11]**.** *For small values of G the probability of the pattern $ABSAB$ in RC4 keystream, where S is a G-byte string, is* $(1 + e^{(-4-8G)/N}/N) \cdot 1/N^2$.

2.2 Previous Works

This section briefly reviews known attacks on RC4 in the broadcast setting where the same plaintext is encrypted with different randomly-chosen keys.

Mantin-Shamir (MS) Attack. Mantin and Shamir first presented broadcast RC4 attacks. Their attacks exploit a bias of second byte of keystream, Z_2 [12] as follows.

Theorem 2 [12]**.** *Assume that the initial permutation S is randomly chosen from the set of all the possible permutations of $\{0, 1, 2, \ldots, N-1\}$. Then the probability that the second output byte of RC4 is 0 is approximately $\frac{2}{N}$.*

This probability is estimated as $\frac{2}{256}$ when $N = 256$. Based on this bias, a distinguishing attack and a plaintext recovery attack on RC4 in the broadcast setting are demonstrated by Theorems 3 and 4, respectively.

Theorem 3 [12]. *Let X and Y be two distributions, and suppose that the event e happens in X with probability p and in Y with probability $p \cdot (1+q)$. Then for small p and q, $\mathrm{O}(\frac{1}{p \cdot q^2})$ samples suffice to distinguish X from Y with a constant probability of success.*

In this case, p and q are given as $p = 1/N$ and $q = 1$. The number of samples is about $\frac{1}{p \cdot q^2} = N$.

Theorem 4 [12]. *Let P be a plaintext, and let $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$ be the RC4 encryptions of P under k uniformly distributed keys. Then, if $k = \Omega(N)$, the second byte of P can be reliably extracted from $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$.*

According to the relation $C_2^{(i)} = P_2^{(i)} \oplus Z_2^{(i)}$, if $Z_2^{(i)} = 0$ holds, then $C_2^{(i)}$ is same as $P_2^{(i)}$. From Theorem 2, $Z_2 = 0$ occurs with twice the expected probability of a random one. Thus, most frequent byte in amongst $C_2^{(1)}, C_2^{(2)}, \ldots, C_2^{(k)}$ is likely to be P_2 itself. When $N = 256$, it requires more than 2^8 ciphertexts encrypted with randomly-chosen keys.

Maitra-Paul-Sen Gupta (MPS) Attack. Maitra, Paul and Sen Gupta showed that $Z_3, Z_4, \ldots, Z_{255}$ are also biased to 0 [6,9]. Although Mantin and Shamir assume that an initial permutation S is random, Maitra et al. exploit biases of S after the KSA [10]. Then the 3rd to 255th bytes of a plaintext are obtained from $\Omega(N^3)$ ciphertexts encrypted with different keys.

Isobe-Ohigashi-Watanabe-Morii (IOWM) Attack. Isobe et al. proposed a full plaintext recovery attack, which is able to extract the full bytes of a plaintext on RC4 from ciphertexts in the broadcast setting [7]. Their attack consists of two phases: an initial byte recovery phase and a sequential recovery phase for finding later bytes of a plaintext.

In the initial byte recovery phase, the first 257 bytes of a plaintext are recovered by using the cumulative bias set of $Z_1, Z_2, \ldots, Z_{257}$. Their cumulative bias set includes a conditional bias $Z_1 = 0 | Z_2 = 0$ [7] and single byte biases $Z_2 = 0$ [12], $Z_3 = 131$ [7], $Z_r = 0$ for $3 \le r \le 255$ [6,9], $Z_r = r$ for $3 \le r \le 255$ [7], $Z_{16} = 240$ [5], $Z_r = (256 - r)$ for $r = 32, 48, 64, 80, 96, 112$ [7], and $Z_{256} \neq 0$ [7], $Z_{257} = 0$ [7] (when $N = 256$ and $\ell = 16$). Given 2^{32} ciphertexts encrypted by randomly-chosen keys, the first 257 bytes of a plaintext are extracted with probability more than 0.8.

In the sequential recovery phase, the later bytes (after P_{258}) are sequentially recovered with the first 257 bytes of the plaintext, which were already obtained in the initial byte recovery phase. The sequential algorithm effectively uses a long-term bias, the $ABSAB$ bias [11]. In particular, $ABSAB$ biases with different G are simultaneously used for enhancing the attack, using the following lemmas for the discrimination.

Lemma 1 [11]. *Let X and Y be two distributions and suppose that the independent events $\{e_i : 1 \le i \le k\}$ occur with probabilities $\Pr_X(e_i) = p_i$ in X and*

$\Pr_Y(e_i) = (1 + q_i) \cdot p_i$ *in* Y. *Then the discrimination* D *of the distributions is* $\sum_i p_i \cdot q_i^2$.

The number of required samples for distinguishing the biased distribution from the random distribution with probability of $1-\alpha$ is given as the following lemma.

Lemma 2 [11]. *The number of samples that is required for distinguishing two distributions that have discrimination* D *with success rate* $1 - \alpha$ *(for both directions) is* $(1/D) \cdot (1 - 2\alpha) \cdot log_2 \frac{1-\alpha}{\alpha}$.

This lemma shows that in the broadcast RC4 attack, once the discrimination D and the number of samples k are given, the success probability $\Pr_{distinguish}$ for distinguishing the distribution of correct candidate plaintext byte (the biased distribution) from the distribution of one wrong candidate of plaintext byte (a random distribution) always becomes constant. The success probability for recovering plaintext bytes depends on $\Pr_{distinguish}$. Thus if k is fixed, the success probability only depends on D.

In their attack, the following equation regarding the $ABSAB$ bias is used.

$$\begin{aligned}
&(C_r \mid\mid C_{r+1}) \oplus (C_{r+2+G} \mid\mid C_{r+3+G}) \\
&= (P_r \oplus Z_r \mid\mid P_{r+1} \oplus Z_{r+1}) \oplus (P_{r+2+G} \oplus Z_{r+2+G} \mid\mid P_{r+3+G} \oplus Z_{r+3+G}) \\
&= (P_r \oplus P_{r+2+G} \oplus Z_r \oplus Z_{r+2+G} \mid\mid P_{r+1} \oplus P_{r+3+G} \oplus Z_{r+1} \oplus Z_{r+3+G}). \quad (2)
\end{aligned}$$

Assuming that Eq. (1) (event of the $ABSAB$ bias) holds, the relation of plaintexts and ciphertexts without keystreams is obtained, i.e., $(C_r \mid\mid C_{r+1}) \oplus (C_{r+2+G} \mid\mid C_{r+3+G}) = (P_r \oplus P_{r+2+G} \mid\mid P_{r+1} \oplus P_{r+3+G}) = (P_r \mid\mid P_{r+1}) \oplus (P_{r+2+G} \mid\mid P_{r+3+G})$. For combining these relations with different G to enhance the biases, the algorithm uses the knowledge of pre-guessed plaintext bytes. For example, in the cases of ($r = r'$ and $G = 1$) and ($r = r' + 1$ and $G = 0$), right parts of equations are given as $(P_{r'} \mid\mid P_{r'+1}) \oplus (P_{r'+3} \mid\mid P_{r'+4})$ and $(P_{r'+1} \mid\mid P_{r'+2}) \oplus (P_{r'+3} \mid\mid P_{r'+4})$, respectively. Then, if $P_{r'}$, $P_{r'+1}$, and $P_{r'+2}$ are already known, the two equations with respected to $(P_{r'+3} \mid\mid P_{r'+4})$ is obtained by transposing $P_{r'}$, $P_{r'+1}$, and $P_{r'+2}$ to the left part of the equation. Then, these equations with different G can be merged.

Suppose that $P_1, P_2, \ldots, P_{257}$ are guessed by the cumulative bias set. Then, the sequential algorithm for recovering P_r for $r = 258, 259, \ldots, P_{MAX}$, from k ciphertexts $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$ encrypted by different keys, by using $ABSAB$ biases of $G = 0, 1, \ldots, G_{MAX}$ is given as follows.

Step 1 Obtain $C_{258-3-G_{MAX}}, C_{258-2-G_{MAX}}, \ldots, C_{P_{MAX}}$ in each ciphertext, and make frequency tables $T_{count}[r][G]$ of $(C_{r-3-G} \mid\mid C_{r-2-G}) \oplus (C_{r-1} \mid\mid C_r)$ for all $r = 258, 259, \ldots, P_{MAX}$ and $G = 0, 1, \ldots, G_{MAX}$, where $(C_{r-3-G} \mid\mid C_{r-2-G}) \oplus (C_{r-1} \mid\mid C_r) = (P_{r-3-G} \mid\mid P_{r-2-G}) \oplus (P_{r-1} \mid\mid P_r)$ only if Eq. (1) holds.

Step 2 Set $r = 258$.

Step 3 Guess the value of P_r.

Step 3.1 For $G = 0, 1, \ldots, G_{MAX}$, convert $T_{count}[r][G]$ into a frequency table $T_{marge}[r]$ of $(P_{r-1} \,||\, P_r)$ by using pre-guessed values of $P_{r-3-G_{MAX}}$, $\ldots, P_{r-2}$, and merge counter values of all tables.

Step 3.2 Make a frequency table $T_{guess}[r]$ indexed by only P_r from $T_{marge}[r]$ with knowledge of the P_{r-1}. To put it more precisely, using a pre-guessed value of P_{r-1}, only tables $T_{marge}[r]$ corresponding to the value of P_{r-1} is taken into consideration. Finally, regard most frequency one in table $T_{guess}[r]$ as the correct P_r.

Step 4 Increment r. If $r = P_{MAX} + 1$, terminate this algorithm. Otherwise, go to Step 3.

Isobe et al. theoretically showed that this algorithm can recover consecutive 1000 terabytes of a plaintext from 2^{34} ciphertexts when $G_{MAX} = 63$ ($D = 2^{-28.03}$) is adopted.

Countermeasure. These attacks essentially exploit biases in the initial (1st to 257th) bytes of the RC4 keystream. If initial bytes of the keystream are disregarded, it does not work any more as mentioned in [7]. Thus, the RC4-drop(257) is considered as a countermeasure against previous plaintext recovery attacks, where RC4-drop(n) is an RC4 implementation that disregards the first n bytes of a keystream[2].

In addition, Mironov also recommended $n = 512$ or 768, and gave a conservative recommended parameter $n = 3072$ based on the experimental data for avoiding initial bytes biases [13].

3 Plaintext Recovery Attack Using Known Partial Plaintext Bytes

In this section, we propose a plaintext recovery attack that is feasible even if initial bytes of a keystream are disregarded, unlike previous attacks. We improve Isobe et al.'s attack so that it works without initial biases of the keystream. In particular, we suppose that an attacker has the partial knowledge of a target plaintext, e.g., fixed header information. This assumption is reasonable in the practical usage on RC4. Then, with partial knowledge of the target plaintext, the other bytes of the plaintext can be recovered by using $ABSAB$ biases.

For the simplification, the encryption on RC4-drop(n) denotes $C_r = P_r \oplus Z_{r+n} = P_r \oplus Z_{r^*}$ where $r^* = r + n$.

3.1 Attack Functions

We give two functions based on $ABSAB$ biases for recovering an unknown byte of the plaintext.

[2] RC4-drop(n) is a generalized implementation of the countermeasure written by [13], and this is defined at http://www.users.zetnet.co.uk/hopwood/crypto/scan/cs.html.

Algorithm 2. $f_{ABSAB\_F}()$

Require: r, /* round number of a plaintext to be guessed */
G_{MAX}, /* parameter of the ABSAB bias*/
$P_{r-G_{MAX}-3}, \ldots, P_{r-1}$, /* known plaintext bytes */
$(C_{r-G_{MAX}-3}, \ldots, C_r)$s of $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$ /* bytes of k ciphertexts encrypted by different keys */
Ensure: P_r

1: **for** $G = 0$ **to** G_{MAX} **do**
2: Make frequency tables $T_{count}[r][G]$ of $(C_{r-3-G} \,||\, C_{r-2-G}) \oplus (C_{r-1} \,||\, C_r)$ from all ciphertexts $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$.
3: Convert $T_{count}[r][G]$ into a frequency table $T_{marge}[r]$ of $(P_{r-1} \,||\, P_r)$ by $P_{r-3-G_{MAX}}, \ldots, P_{r-2}$, and merge counter values of all tables.
4: **end for**
5: Make a frequency table $T_{guess}[r]$ indexed by only P_r from $T_{marge}[r]$ with knowledge of P_{r-1}. To put it more precisely, using a pre-guessed value of P_{r-1}, only tables $T_{marge}[r]$ corresponding to the value of P_{r-1} is taken into consideration.
6: Regard most frequency one in table $T_{guess}[r]$ as the correct P_r.
7: Output P_r

$f_{ABSAB\_F}()$ **:** Find an unknown byte P_r from pre-known consecutive $(G_{MAX} + 3)$ bytes of a plaintext $P_{r-G_{MAX}-3}, \ldots, P_{r-1}$ (See Algorithm 2).
$f_{ABSAB\_B}()$ **:** Find an unknown byte P_r from pre-known consecutive $(G_{MAX} + 3)$ bytes of a plaintext $P_{r+1}, \ldots, P_{r+G_{MAX}+3}$.

The algorithm of $f_{ABSAB\_B}()$ is given by replacing "−" of subscripts of variables in Algorithm 2 to "+". These functions can be obtained from Step 1 and Step 3 of the IOWM attack. Figure 1 (Fig. 3) illustrates the procedures of $f_{ABSAB\_F}()$ and $f_{ABSAB\_B}()$. By using above two functions, all plaintext bytes can be recovered from the partial knowledge of the plaintext and the corresponding ciphertexts.

3.2 Attack Procedure

Suppose that x bytes of a target plaintext, $P_r, \ldots P_{r+x-1}$, are given. An attacker aims to recover the next byte (P_{r+x}) or the previous byte (P_{r-1}) of the known plaintext bytes by using $f_{ABSAB\_F}()$ or $f_{ABSAB\_B}()$, respectively. If (P_{r+x}) or (P_{r-1}) is successfully recovered, the attacker recovers (P_{r+x+1}) or (P_{r-2}) with knowledge of (P_{r+x}) or (P_{r-1}). Since G_{MAX} increases, the probability for recovering a plaintext byte also increases in the next step.

Our attack repeats above procedure until the all plaintext bytes are found. After G_{MAX} reaches 63, G_{MAX} is fixed since the increase of D is converged around $G_{MAX} = 63$ as mentioned in [7]. Figure 2 shows that our plaintext recovery attack using known partial plaintext bytes when consecutive 6 bytes of a target plaintext are given.

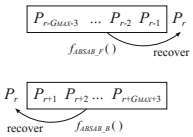

Fig. 1. Forward and backward functions for recovering one byte of a target plaintext using the partial knowledge of the plaintext and $ABSAB$ biases

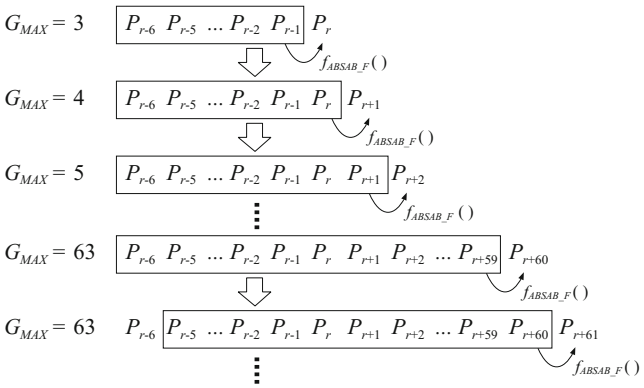

Fig. 2. A plaintext recovery attack using the known partial plaintext bytes when consecutive 6 bytes of a target plaintext are known

3.3 Experimental Results

We evaluate our plaintext recovery attack on RC4-drop(n) in the broadcast setting by the computer experiment when $N = 256$ and $n = 3072$, which is a conservative recommended parameter given in [13]. Then, ciphertext C is expressed as $(C_1, C_2, \ldots, C_r, \ldots) = (P_1 \oplus Z_{1+3072}, P_2 \oplus Z_{2+3072}, \ldots, P_r \oplus Z_{r+3072}, \ldots)$.

In order to estimate the success probability of our attack, we evaluate the probabilities for recovering the one byte of the target plaintext by $f_{ABSAB\_F}()$ and $f_{ABSAB\_B}()$. The probabilities dependent on G_{MAX} and the number of obtained ciphertexts, but does not depend on the round number r. Thus, our experiment uses parameters such that $G_{MAX} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 31, 63$ and $2^{31}, 2^{32}, \ldots, 2^{36}$ ciphertexts.

Table 2 shows the experimental result for 128 different plaintexts when $r = 128$, $n = 3072$, and the number of known plaintext bytes and the discrimination D and corresponding G_{MAX}. The success probability for recovering P_r increases with the increasing the value of G_{MAX} and D.

For the estimation of the impact in the realistic environment, let us consider the situation of Fig. 2 where an attacker knows consecutive *only 6 bytes* of a target plaintext. Suppose that 2^{34} ciphertexts encrypted by randomly-chosen keys are obtained, the probability for recovering P_r by $f_{ABSAB\_F}()$ with $G_{MAX} = 3$ is estimated as 0.8125. Similarly, $P_{r+1}, P_{r+2}, \ldots P_{r+5}$ are recovered by $f_{ABSAB\_F}()$

Table 2. The probabilities for recovering P_{128} by using $f_{ABSAB\_F}()$ and $P_{128-G_{MAX}-3}$, $\ldots, P_{127}$ when $n = 3072$

# of known plaintext bytes	G_{MAX}	D	# of ciphertexts 2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}
3	0	$2^{-32.05}$	0.0078	0.0547	0.0625	0.1250	0.4609	0.8750
4	1	$2^{-31.09}$	0.0156	0.0469	0.1797	0.4141	0.8516	0.9766
5	2	$2^{-31.55}$	0.0625	0.1484	0.3516	0.6641	0.9688	1.0000
6	3	$2^{-30.18}$	0.0703	0.1875	0.4297	0.8125	0.9922	1.0000
7	4	$2^{-29.90}$	0.1172	0.2266	0.5156	0.8750	0.9922	1.0000
8	5	$2^{-29.68}$	0.0938	0.2656	0.6250	0.9375	1.0000	1.0000
9	6	$2^{-29.50}$	0.1563	0.3438	0.7344	0.9688	1.0000	n/a
10	7	$2^{-29.35}$	0.1484	0.3594	0.7656	0.9922	1.0000	n/a
11	8	$2^{-29.22}$	0.1484	0.4063	0.7578	0.9922	1.0000	n/a
12	9	$2^{-29.11}$	0.1484	0.4922	0.8203	1.0000	1.0000	n/a
18	15	$2^{-28.66}$	0.2969	0.6172	0.9453	1.0000	1.0000	n/a
34	31	$2^{-28.21}$	0.3359	0.7656	0.9766	1.0000	n/a	n/a
66	63	$2^{-28.03}$	0.3672	0.7656	0.9766	1.0000	n/a	n/a

with probabilities of 0.8750, 0.9375, 0.9688, 0.9922, 0.9922, where these parameters are $G_{MAX} = 4, 5, 6, 7, 8$, respectively. Then, the attacker obtains the consecutive $12(= 6+6)$ bytes with probability of $(0.8125) \cdot (0.8750) \cdot (0.9375) \cdot (0.9688) \cdot (0.9922) \cdot (0.9922) \sim 0.636$. $P_{r+6}, P_{r+7}, \ldots, P_{r+59}$ are expected to be recovered by $f_{ABSAB\_F}()$ with probability of one from Table 2. After that, other bytes of the target plaintexts can be recovered with probability of one similar to the IOWM attack because the parameter becomes $G_{MAX} = 63$. Therefore, in our attack, the only knowledge of consecutive 6 bytes of the target plaintexts enables to recover 1000 terabytes of the target plaintext with probability of about 0.636 from 2^{34} ciphertexts. The number of required ciphertexts of this attack is same as that of IOWM attack, while IOWM attack uses the initial 257 bytes of the keystream.

4 Guess-and-Determine Plaintext Recovery Attack (GD Attack)

This section gives a plaintext recovery attack which does not require any previous knowledge of the plaintext unlike the attack in Sect. 3. In order to achieve it, we develop a *guess-and-determine* (GD) plaintext recovery method based on two strong long-term biases, the FM00 bias and the *ABSAB* bias. Generally, in stream ciphers, the guess-and-determine method is considered as a technique for internal state recovery attacks such that a part of an internal state is determined from the other parts by exploiting the relations between the state and keystream. Our method seems to be a new class of the guess-and-determine methods for the plaintext recovery attack.

Assuming P_r is the target byte, the overview of our GD attack is given as follows.

1. Guess the value of P_r.
2. Recover x bytes of the plaintext, $P_{r-x}, \ldots, P_{r-1}$, from P_r (guessed in Step 1) by using the FM00 bias.
3. Recover P'_r from $P_{r-x}, \ldots, P_{r-1}$ (guessed in Step 2) by using the $ABSAB$ bias.
4. If P'_r is not equal to P_r guessed in Step 1, the value is wrong. Otherwise the value is regarded as a candidate of correct P_r.

For each candidate of P_r, Step 1–4 are performed. The basic idea behind our GD attack is that if the value of P_r guessed in Step 1 is correct, P'_r is surely same as P_r guessed in Step 1. Two biases are used for the detection the wrong candidates of plaintext bytes.

In this section, we firstly give attack functions based on the FM00 bias for guess-and-determine methods. Then, we explain the detailed algorithm of guess-and-determine methods. Finally we evaluate this attack.

4.1 FM00 Bias for GD Attack

The FM00 bias is a two-word bias (See Table 1), and is relatively weaker than the $ABSAB$ bias. If the simple count-up method for guessing correct plaintext byte is used, the FM00 bias is not directly used for efficient plaintext recovery attacks, because some events indexed by same r^* are dependent each other.

For example, let us consider two events of $(Z_{r^*}, Z_{r^*+1}) = (0, 0)$ and $(Z_{r^*}, Z_{r^*+1}) = (0, 1)$, whose probabilities are same. Here, the relation of plaintext and ciphertext is given as $(P_r, P_{r+1}) = (C_r \oplus Z_{r^*}, C_{r+1} \oplus Z_{r^*+1})$. If the event of $(Z_{r^*}, Z_{r^*+1}) = (0, 0)$ occurs, the relation of $(P_r, P_{r+1}) = (C_r, C_{r+1})$ hold. On the other hand, if the event of $(Z_{r^*}, Z_{r^*+1}) = (0, 1)$ occurs, $(P_r, P_{r+1} \oplus 1) = (C_r, C_{r+1})$ holds. Since these probabilities are same, we can not determine whether most frequency (C_r, C_{r+1}) is equal to (P_r, P_{r+1}) or $(P_r, P_{r+1} \oplus 1)$ in the plaintext recovery attack[3].

Conditional Bias Regarding the FM00 Bias. So that FM00 biases can be independently used for the plaintext recovery attack, we convert the FM00 bias into conditional bias such that $\Pr(Z_{r^*+1}|Z_{r^*})$ (the forward) or $\Pr(Z_{r^*}|Z_{r^*+1})$ (the backward), assuming that one byte of the plaintext can be known. Here, we consider the forward and backward conditional biases for previous two events $(Z_{r^*}, Z_{r^*+1}) = (0, 0)$ and $(Z_{r^*}, Z_{r^*+1}) = (0, 1)$.

The backward conditional biases, $(Z_{r^*} = 0|Z_{r^*+1} = 0)$ and $(Z_{r^*} = 0|Z_{r^*+1} = 1)$, are independently used for the plaintext recovery attack. Suppose that P_{r+1}

[3] Yarrkov showed a plaintext recovery attack using the FM00 bias on his web page [15] before our results. However, the detailed description of attacks and estimations are not given, and only the source code is given.

Algorithm 3. $f_{FM00\_B}()$

Require: r, /* round number of plaintext to be guessed */
P_{r+1}, /* known plaintext bytes */
(C_r, C_{r+1})s of $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$ /* bytes of k ciphertexts encrypted by different keys */
Ensure: P_r
1: Make frequency tables of $T_{count}[bias]$ of P_r and P_{r+1} for all FM00 biases regarding P_{r+1} from all ciphertexts $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$.
2: Convert $T_{count}[bias]$ into a frequency table $T_{guess}[r]$ indexed by only P_r with knowledge of P_{r+1}. Here, we only deal with the bias independent of other biases.
3: Regard most frequency one in table $T_{guess}[r]$ as the correct P_r.
4: Output P_r

is obtained, the values of $Z_{r^*+1} = 0$ is computed by $Z_{r^*+1} = C_{r+1} \oplus P_{r+1}$. Then, two tables of $Z_{r^*} = 0$ are obtained from two backward conditional biases $(Z_{r^*} = 0|Z_{r^*+1} = 0)$ and $(Z_{r^*} = 0|Z_{r^*+1} = 1)$. In these tables, it is expected that most frequency values of these tables indicate same value of the plaintext, because source event Z_{r^*+1} are different. Thus, these frequency tables are efficiently merged to recover the plaintext byte P_r.

On the other hand, the forward conditional biases $(Z_{r^*+1} = 0|Z_{r^*} = 0)$ and $(Z_{r^*+1} = 1|Z_{r^*} = 0)$ are not independently used. Even if two tables of $Z_{r^*} = 0$ obtained from $(Z_{r^*+1} = 0|Z_{r^*} = 0)$ and $(Z_{r^*+1} = 1|Z_{r^*} = 0)$ are merged, it is expected that two peaks of $Z_{r^*+1} = 0$ and $Z_{r^*+1} = 1$ are observed due to same source event $Z_{r^*} = 0$.

Therefore, if the source events of conditional bias (the forward is Z_{r^*} and the backward is Z_{r^*+1}) are different, these events can be independently used. Note that events for positive bias and negative bias are not dependent even if the source events of conditional bias are same.

Attack Functions Based on the FM00 Bias. By using all the independent conditional biases, we construct the forward and backward functions for the guess-and-determine attack based on the FM00 bias as follows:

$f_{FM00\_F}()$ **:** Find an unknown byte P_r from pre-known a byte of a plaintext P_{r-1}.
$f_{FM00\_B}()$ **:** Find an unknown byte P_r from pre-known a byte of a plaintext P_{r+1} (See Algorithm 3).

The algorithm of $f_{FM00\_F}()$ is given by replacing "+" of subscripts of variables in Algorithm 3 to "−". Figure 3 illustrates the procedures of $f_{FM00\_F}()$ and $f_{FM00\_B}()$.

The number of independent events of the forward conditional bias N_f and that of the backward conditional bias N_b in each index i are shown in Table 3. When index $i = 0$, the all events of backward conditional bias are independent, and $N_b = 5$. On the other hand, two events of forward conditional bias $Z_{r^*+1} = 1|Z_{r^*} = N - 1$ and $Z_{r^*+1} = 2|Z_{r^*} = N - 1$ are not independent, and $N_f =$

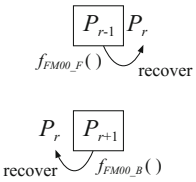

Fig. 3. The guess-and-determine methods based on the conditional bias of the FM00 bias

$5 - 2 = 3$. For all index i, N_b is larger than N_f. Hence, the success probability of $f_{FM00\_B}()$ is larger than that of $f_{FM00\_F}()$.

4.2 Plaintext Recovery Method for Recovering Any Plaintext Byte

Our GD attack utilizes the backward conditional bias of the FM00 bias and the forward $ABSAB$ bias.

To begin with, a plaintext byte P_r is guessed from N candidates. Then, our attack sequentially recovers $P_{r-1} \rightarrow P_{r-2} \rightarrow \dots$ from P_r and the ciphertexts by using $f_{FM00\_B}()$ in the backward manner. Since the number of the candidates of P_x is N, the number of candidates of $(P_r, P_{r-1}, P_{r-2}, \dots)$ is also N. In order to detect the wrong candidates of P_r, we use $f_{ABSAB\_F}()$, which is based on the other bias. In particular, P'_r is obtained from ciphertexts and the candidate of plaintext bytes $(P_{r-1}, P_{r-2}, \dots)$ by using $f_{ABSAB\_F}()$ with G_{MAX}. If the number of ciphertexts is enough larger and P_r is correctly guessed, the relation of $P_r = P'_r$ surely holds. Otherwise the probability that $P_r = P'_r$ holds is $1/N$. After this method, about two candidates of P_r are expected to be left. If the number of the candidates of P_r is not one, the same method is repeated for $P'_{r-1}, P'_{r-2}, \dots$, which are obtained by P_r. If P_r is correct, these method correctly works. In most cases, the number of repeating this method N_{repeat} is less than three. Figure 4 shows the procedure of our plaintext recovery attack for recovering any plaintext byte. The detail of our attack for recovering any plaintext byte P_r is given in Algorithm 4.

We consider the parameter G_{MAX} for the $ABSAB$ bias. It should be chosen so that $ABSAB$ bias is stronger than the FM00 bias to efficiently detect wrong candidates. From Lemma 2, given $1/D$ samples, $\Pr_{distinguish}$ become constant. Since the probability of the plaintext recovery attack depends on $\Pr_{distinguish}$, we evaluate our attack by the number of required ciphertexts for obtaining $1/D$ samples. For example, D of the backward conditional bias of the FM00 bias is estimated as $D = N^{-3} = 2^{-24}$ for $N = 256$ and $i = 3, 4, \dots, N - 4$. From Table 3, there are seven independent events of the FM00 conditional biases in this case. As mentioned before, these biases are independently used. Thus, the probability that a ciphertext matches one of these source events is $7/N$. The number of the required ciphertexts is $(N/7) \cdot (1/D) = 2^{29.19}$ for $1/D$ samples. On the other hand, discriminations D of the $ABSAB$ bias and these number of

Table 3. Events of the conditional bias of the FM00 bias in each index i ($= r^* \bmod N$)

Index i	Z_{r^*}	Z_{r^*+1}	Conditional probability	N_f	N_b
0	0	0	$N^{-1} \cdot (1 + N^{-1})$	3	5
	1	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	1	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	2	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
1	0	0	$N^{-1} \cdot (1 + 2 \cdot N^{-1})$	4	6
	2	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	3	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	2	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
	0	2	$N^{-1} \cdot (1 - N^{-1})$		
2	0	0	$N^{-1} \cdot (1 + N^{-1})$	4	8
	0	1	$N^{-1} \cdot (1 + N^{-1})$		
	3	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	3	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	4	$N^{-1} \cdot (1 + N^{-1})$		
	$N/2+1$	$N/2+1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
	0	3	$N^{-1} \cdot (1 - N^{-1})$		
$3, 4, \ldots, N-4$	0	0	$N^{-1} \cdot (1 + N^{-1})$	3	7
	0	1	$N^{-1} \cdot (1 + N^{-1})$		
	$i+1$	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$i+1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$i+2$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
	0	$i+1$	$N^{-1} \cdot (1 - N^{-1})$		
$N-3$	0	0	$N^{-1} \cdot (1 + N^{-1})$	4	6
	0	1	$N^{-1} \cdot (1 + N^{-1})$		
	$N-2$	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-2$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
	0	$N-2$	$N^{-1} \cdot (1 - N^{-1})$		
$N-2$	0	0	$N^{-1} \cdot (1 + N^{-1})$	2	2
	0	1	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	0	$N^{-1} \cdot (1 + N^{-1})$		
	0	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		
$N-1$	0	1	$N^{-1} \cdot (1 + N^{-1})$	1	3
	0	$N-1$	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	0	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	1	$N^{-1} \cdot (1 + N^{-1})$		
	$N-1$	$N-1$	$N^{-1} \cdot (1 - N^{-1})$		

Algorithm 4. Plaintext Recovery Attack for Recovering Any Plaintext Byte

Require: r, /* round number of plaintext to be guessed */
G_{MAX}, /* parameter of the ABSAB bias*/
$C^{(1)}, C^{(2)}, \ldots, C^{(k)}$ /* k ciphertexts encrypted by different keys */
Ensure: P_r

1: Set all N candidates of a plaintext byte P_r into table T_{cand}.
2: Set $N_{repeat} = 0$.
3: **for all** $P_r \in T_{cand}$ **do**
4: Recover $G_{MAX} + 3 + N_{repeat}$ bytes of the plaintext, $P_{r-G_{MAX}-3-N_{repeat}}, \ldots, P_{r-1}$, from a candidate P_r by using $f_{FM00\_B}()$ and k ciphertexts $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$.
5: Recover $P'_{r-N_{repeat}}$ from $P_{r-G_{MAX}-3-N_{repeat}}, \ldots, P_{r-1-N_{repeat}}$ (guessed in Step 4) by using $f_{ABSAB\_F}()$ and k ciphertexts $C^{(1)}, C^{(2)}, \ldots, C^{(k)}$.
6: **if** $P_{r-N_{repeat}} \neq P'_{r-N_{repeat}}$ **then**
7: The candidate of P_r is removed from T_{cand}.
8: **end if**
9: **end for**
10: **if** the number of candidates in T_{cand} is one **then**
11: Output P_r, and the algorithm stops.
12: **else if** the number of candidates in T_{cand} is zero **then**
13: Our attack fails, and the algorithm stops.
14: **else**
15: Increment N_{repeat}, and go back to Step 3.
16: **end if**

required ciphertexts for recovering a plaintext byte are shown as ($D = 2^{-29.22}$, $1/D = 2^{29.22}$ ciphertexts) for $G_{MAX} = 8$ and ($D = 2^{-29.11}$, $1/D = 2^{29.11}$ ciphertexts) for $G_{MAX} = 9$. Therefore $G_{MAX} = 9$ is chosen for $N = 256$ and $i = 3, 4, \ldots, N - 4$.

4.3 Experimental Results

We perform the computer experiment for demonstrating the effectiveness of our attack with G_{MAX} on RC4-drop(n) in the broadcast setting when $N = 256$ and $n = 3072$. In this experiment, P_{128} is recovered from ciphertexts without the knowledge of the target plaintext. The parameters of the backward conditional bias of the FM00 bias, index i, satisfy $r^* \bmod 256 = i \in \{3, 4, \ldots, N-4\}$. Hence, $G_{MAX} = 9$ is used as the parameter of the $ABSAB$ bias.

First, in order to evaluate $f_{FM00\_B}()$, we obtain the success probability for recovering $P_{114}, \ldots, P_{127}$ under the condition that the correct P_{128} is given. The success probabilities when 2^{29} to 2^{35} ciphertexts are given is shown in Table 4, where the number of tests is 256. The experimental result shows that all bytes of $P_{114}, \ldots, P_{127}$ are recovered from 2^{35} ciphertexts encrypted by randomly-chosen different keys with probability of one by using $f_{FM00\_B}()$. This results also shows that if one byte of the plaintext is known, 2^{33} ciphertexts enable to

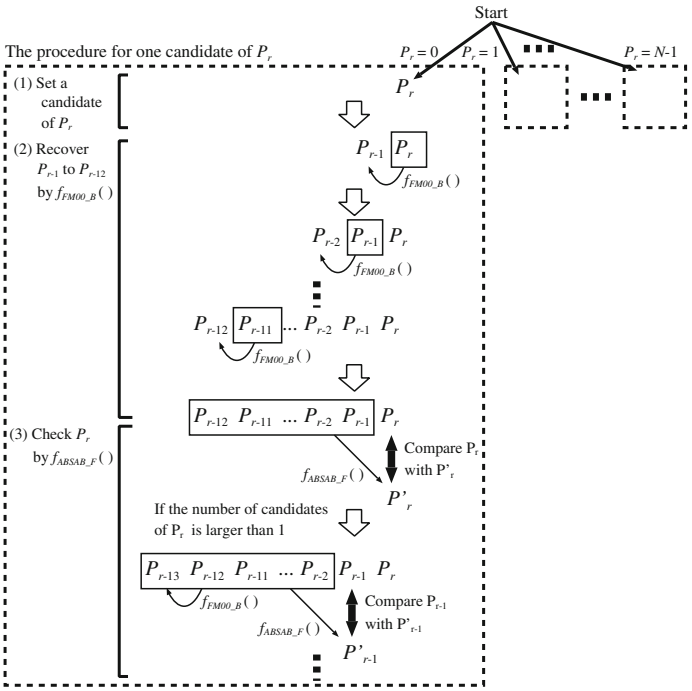

Fig. 4. The procedure of our plaintext recovery attack for recovering any plaintext byte ($G_{MAX} = 9$)

recover the other one byte with probability of about 0.8. Interestingly, it is more efficient than the attack in Sect. 3 when G_{MAX} is small. The attack in Sect. 3 are improved by using $f_{FM00\_B}()$. If consecutive 6 bytes of the plaintext are known, the other bytes can be recovered from 2^{34} with probability of about 0.984, while that of the attack in Sect. 3 is about 0.636.

Then, we estimate the success probability for recovering P_{128} from only ciphertexts by our plaintext recovery attack. The success probability when 2^{32} to 2^{35} ciphertexts are given is shown in Table 5, where the number of tests is 256. This experiment requires about one week with one CPU core (Intel(R) Core(TM) i7 CPU 920@ 2.67 GHz) to obtain the result of one plaintext. The experimental result shows that our attack can recover the target plaintext byte P_{128} with probability of one from 2^{35} ciphertexts encrypted by randomly-chosen different keys. Remaining plaintext bytes, namely $P_r (r \neq 128)$ can be recovered by repeating our attack or using our attack functions $f_{ABSAB\_F}()$, $f_{ABSAB\_B}()$, $f_{FM00\_F}()$, and $f_{FM00\_B}()$. Especially, in the cases of $i = N-2, N-1$, the success probabilities of conditional bias of the FM00 bias are relatively small than that of others cases. These bytes should be recovered by using $f_{ABSAB\_F}()$, $f_{ABSAB\_B}()$ for an efficient recovery attack after other bytes are recovered by using our GD attack.

Table 4. Success probabilities of $f_{FM00\_B}()$ for recovering $(P_{114}, \ldots, P_{127})$ under the condition that the correct P_{128} is given when $n = 3072$

	# of ciphertexts						
	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}
P_{114}	0.0039	0.0039	0.0039	0.0078	0.0781	0.8750	1.0000
P_{115}	0.0039	0.0039	0.0078	0.0117	0.1055	0.8828	1.0000
P_{116}	0.0078	0.0000	0.0078	0.0117	0.1133	0.8828	1.0000
P_{117}	0.0039	0.0039	0.0078	0.0078	0.1328	0.8945	1.0000
P_{118}	0.0078	0.0195	0.0078	0.0078	0.1758	0.9023	1.0000
P_{119}	0.0078	0.0000	0.0078	0.0117	0.1992	0.9180	1.0000
P_{120}	0.0039	0.0039	0.0078	0.0156	0.2422	0.9258	1.0000
P_{121}	0.0039	0.0039	0.0117	0.0078	0.2773	0.9492	1.0000
P_{122}	0.0039	0.0078	0.0039	0.0117	0.3203	0.9570	1.0000
P_{123}	0.0000	0.0117	0.0117	0.0195	0.3672	0.9688	1.0000
P_{124}	0.0078	0.0039	0.0195	0.0391	0.4727	0.9844	1.0000
P_{125}	0.0078	0.0039	0.0078	0.0742	0.5820	0.9883	1.0000
P_{126}	0.0039	0.0078	0.0391	0.1602	0.6680	0.9922	1.0000
P_{127}	0.0430	0.0898	0.1719	0.3984	0.8008	0.9922	1.0000

Table 5. Success probabilities of our attack for recovering P_{128} when $n = 3072$

	# of ciphertexts			
	2^{32}	2^{33}	2^{34}	2^{35}
P_{128}	0.0039	0.1133	0.9102	1.0000

Given 2^{34} ciphertexts encrypted by randomly-chosen different keys, our attack can recover any plaintext byte with probability of about 0.91. The number of required ciphertexts are same as that of IOWM attack on original RC4, which does not discard initial keystream bytes. In addition, even if 2^{33} ciphertexts are given, our attack is more efficient than a random guess.

Also, this attack is applicable to original RC4 with constant success probability regardless of the position of plaintext bytes, while that of the IOWM attack decrease for the later plaintext byte. It is an advantage of our attack from the IOWM attack on the original RC4.

5 Conclusion

In this paper, we have evaluated the security of relatively secure RC4 implementation called RC4-drop(n), which discards the first n bytes of the keystream. We proposed two advanced plaintext recovery attacks that can recover *any* byte of a plaintext on RC4-drop(n) in the broadcast setting or the multi-session setting. The first attack is the modified IOWM attack. Using partial knowledge of the target plaintext, the other bytes can be recovered from ciphertexts encrypted by different keys. The attack can recover 1000 terabytes of a target plaintext

with the high probability from 2^{34} ciphertexts encrypted by different keys if the knowledge of only consecutive 6 bytes of the target plaintext is given. The second attack does not rely on any previous knowledge of a plaintext. In order to achieve it, we developed a *guess-and-determine* plaintext recovery method based two strong long-term biases. Given 2^{35} ciphertext encrypted by different keys, any byte of a plaintext can be recovered with probability close to one from only ciphertexts. The amount of ciphertext is almost same as the IOWM attack on original RC4. Therefore, RC4 is not secure even if the enough initial keystream bytes are disregarded.

We recommend to replace RC4 with other stream ciphers [8] or the algorithms of authenticated encryption in the practical protocols and software applications.

A future work is to compare our attack with the method of [1,2] in the same conditions. In addition, we will combine our attack with the method of [1,2] for obtaining more efficient attacks.

Acknowledgments. This work was supported in part by Grant-in-Aid for Scientific Research (C) (KAKENHI 23560455) and Grant-in-Aid for Young Scientists (B) (KAKENHI 25730085) for Japan Society for the Promotion of Science.

References

1. AlFardan, N.J., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.N.: On the security of RC4 in TLS. In: USENIX Security 2013 (2013) (to appear)
2. AlFardan, N.J., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.N.: On the security of RC4 in TLS and WPA. http://www.isg.rhul.ac.uk/tls/RC4biases.pdf (2013)
3. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599. Springer, Heidelberg (2003)
4. Fluhrer, S.R., McGrew, D.A.: Statistical analysis of the alleged RC4 keystream generator. In: Schneier [14], pp. 19–30
5. Sen Gupta, S., Maitra, S., Paul, G., Sarkar, S.: Proof of empirical RC4 biases and new key correlations. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 151–168. Springer, Heidelberg (2012)
6. Sen Gupta, S., Maitra, S., Paul, G., Sarkar, S.: (Non-)random sequences from (Non-) random permutations - analysis of RC4 stream cipher. J. Cryptol. 1–42 (2012). http://dblp.uni-trier.de/rec/bibtex/journals/joc/GuptaMPS14
7. Isobe, T., Ohigashi, T., Watanabe, Y., Morii, M.: Full plaintext recovery attack on broadcast RC4. Preproceeding of Fast Software Encryption (FSE) (2013)
8. Josefsson, S., Strombergson, J., Mavrogiannopoulos, N.: The salsa20 stream cipher for transport layer security (TLS) and datagram transport layer security (DTLS). Network Working Group Internet-Draft, March 2013. http://tools.ietf.org/html/draft-josefsson-salsa20-tls-01 (2013)
9. Maitra, S., Paul, G., Sen Gupta, S.: Attack on broadcast RC4 revisited. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 199–217. Springer, Heidelberg (2011)
10. Mantin, I.: Analysis of the stream cipher RC4. Master's Thesis, The Weizmann Institute of Science, Israel. http://www.wisdom.weizmann.ac.il/itsik/RC4/rc4.html (2001)

11. Mantin, I.: Predicting and distinguishing attacks on RC4 keystream generator. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 491–506. Springer, Heidelberg (2005)
12. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)
13. Mironov, I.: (not so) random shuffles of RC4. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 304–319. Springer, Heidelberg (2002)
14. Schneier, B. (ed.): FSE 2000. LNCS, vol. 1978. Springer, Heidelberg (2001)
15. Yarrkov, E.: Why the recent RC4 attack doesn't surprise me. https://cipherdev.org/rc4_2013-03-13.html (2013)