Journal of Information Processing Vol.22 No.2 357–365 (Apr. 2014)

[DOI: 10.2197/ipsjjip.22.357]

Recommended Paper

Expanding Weak-key Space of RC4

Atsushi Nagao[1,a)] Toshihiro Ohigashi[2] Takanori Isobe[1] Masakatu Morii[1]

Received: July 7, 2013, Accepted: November 1, 2013

Abstract: RC4 is a stream cipher designed by Rivest in 1987. It is the most famous stream cipher and widely used e.g., SSL/TLS, WEP and WPA. Although RC4 in particular implementations and settings such as the WEP implementation and the broadcast setting, was already broken, RC4 itself is not completely broken yet. In 2011, Teramura et al. generalized classes of weak keys of RC4 by using the predictive state, which are special classes of the internal state of RC4. The total number of Teramura et al.'s weak keys is approximately $2^{117.29}$. Their weak-key attack can recover a 128-bit secret key with *efficiency* of $2^{95.10}$, where *efficiency* is defined as time complexity per success probability of the attack. This attack works only if particular patterns of the keystream are observed. In this paper, we further expand weak-key space of RC4. By thoroughly analyzing the relation between the key and the initial state of the pseudo-random generation algorithm, we can find new classes of predictive state which are utilized for key recovery attacks. As a result, $2^{118.58}$ keys can be defined as new weak keys, whose number is more than twice the number of Teramura et al.'s weak keys. Moreover, our attack is applicable to *any* keystream, while Teramura et al.'s attack is feasible only in particular patterns of the keystream. Given *any* keystream, our weak-key attack can recover a 128-bit secret key with efficiency of $2^{115.11}$. Our attack is the best-known single-key key recovery attack on RC4 with respect to efficiency. In addition, if we focus on specific keystreams similar to Teramura et al.'s attack, the 128-bit secret key can be recovered with efficiency of $2^{76.32}$, which is more efficient than Teramura et al.'s attack.

Keywords: cryptanalysis, stream cipher, RC4, weak key, predictive state, key recovery attack

1. Introduction

RC4 is a stream cipher designed by Rivest in 1987 [1]. It is widely used in security protocols such as Secure Sockets Layer (SSL), Transport Layer Security (TLS) [2], Wired Equivalent Privacy (WEP) [3] and Wi-Fi Protected Access (WPA) [4]. The key length L and the array size of an internal state N are variable. Typical parameters are $L = 16$ (128-bit key) and $N = 256$. RC4 consists of two algorithms: Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA). An internal state of the PRGA is initialized by a secret key in the KSA, and the arbitrary length keystream is generated from the internal state in the PRGA. The plaintext/ciphertext is encrypted/decrypted by XORed with the keystream.

A number of attacks for the WEP implementation were suggested since the weakness of WEP was pointed out by Fluhrer et al. [5]. Among them, the VX attack, which was suggested by Vaudenay et al. in 2007 [6], is a powerful attack. In 2013, Sepehrdad et al. improved the VX attack [7]. Their attack is feasible with 22,500 packets and probability 0.5 and worked with an inexpensive PC. On the other hand, in 2001, Mantin et al. presented a practical plaintext recovery attack for the second plaintext byte on RC4 in the broadcast setting where the same plaintext is encrypted with different user keys [8]. The attack requires only N ciphertexts. In 2011, Maitra et al. showed biases of the 3rd to 255th of the RC4 keystream, and proposed an attack based on these biases for recovering the 3rd to 255th of the plaintext bytes [9]. They estimated the number of required ciphertexts for the attack as N^3. In 2013, Isobe et al. proposed the full plaintext attack on RC4 in the broadcast setting. Their attack can recover first 257 bytes and 2^{50} bytes of a plaintext from 2^{32} and 2^{34} ciphertexts with probability almost one [10]. Since Isobe et al.'s attack and Sepehrdad et al.'s attack require specific conditions such as the WEP implementation and the broadcast setting, it does not mean that RC4 itself was completely broken.

Over past 20 years, a number of security analyses on RC4 itself have been proposed. Single key distinguisher [11], [12], [13] is the attack for distinguishing a keystream from a true random stream. Equivalent key [14], [15] is a pair of keys which generates same keystream. A state recovery attack, which recovers the internal state from the keystream, was proposed by Knudsen et al. in 1998 [16]. In 2008, Maximov et al. improved this attack [17]. Their attack requires 2^{241} time complexity. Herewith, RC4 which has the key length longer than 241-bit was broken, however, 128-bit key RC4 did not have the influence from this attack. The key recovery attack was proposed by Sepehrdad et al. in 2010 [18]. This attack recovers a secret key from a keystream with efficiency of $2^{122.06}$ without any condition of the keystream. The efficiency is defined as time complexity per success probability of the attack. Generally speaking, key recovery attacks relying on weak keys or biases do not succeed with probability one. Therefore,

[1] Graduate School of Engineering, Kobe University, Kobe, Hyogo 657–8501, Japan
[2] Information Media Center, Hiroshima University, Higashihiroshima, Hiroshima 739–8511, Japan
[a)] a.nagao@stu.kobe-u.ac.jp

The initial version of this paper was presented at Computer Security Symposium 2012 (CSS 2012) in October 2012. This paper was recommended to be submitted to Journal of Information Processing (JIP) by Program Chair of CSS 2012.

it is necessary to use a criterion in consideration of the success probability. The efficiency means the average time complexity to get the one key with many trials. By the efficiency, a variety of key recovery attacks are able to be compared with the same criterion. When the x-bit key is used, the efficiency lower than 2^x means that the attack is more effective than a random key search. Sepehrdad et al.'s key recovery attack was the best single-key attack of 128-bit key RC4.

Weak key is the class of special keys which can be recovered from the keystream more effective than a random key search. In 1995, Roos showed the existence of weak key in RC4 [19]. The key recovery attack using Roos' weak keys has efficiency of $2^{122.9}$. Teramura et al. expanded Roos' weak keys in 2011 [20]. They exploit a part of the predictive state for a key recovery attack, where the predictive state is the special states of the PRGA on RC4, and predicts a few bytes keystream from a few bytes of the internal state. The total number of Teramura et al.'s weak keys is roughly estimated as $2^{117.29}$. By using these weak keys, a 128-bit key is recovered with efficiency of $2^{95.10}$, only if particular patterns of keystream are observed. This attack seems not to be the standard key recovery setting, because it is applicable to particular patterns of the keystream. Generally, key recovery attacks of stream ciphers are estimated for arbitrary keystream even if weak key setting [18].

In this paper, we aim to further expand the weak-key space of RC4, and estimate the accurate number of weak keys and efficiency of the key recovery attack in a standard setting. By thoroughly analyzing the relation between the key and the initial state of the PRGA, we can find new classes of predictive state which are utilized for key recovery attacks. Specifically, we exploit the discrete predictive states in the range of $S[0], \ldots, S[15]$, while Teramura et al.'s weak key use only continuous predictive states as $S[0], S[1], \ldots, S[a-1]$. Also, we succeed in relaxing the condition for leading a part of the secret key from predictive state by utilizing probabilistic events in the KSA. As a result, $2^{118.58}$ keys can be defined as new weak keys, whose number is more than twice the number of Teramura et al.'s weak keys. Then, we propose key recovery attacks based on our new weak keys. Our attack is applicable to *any* keystream unlike Teramura et al.'s attack. Given a keystream, our attack enables to recover a 128-bit secret key with efficiency of $2^{115.11}$. **Table 1** is the comparison of previous and our key recovery attacks. Our attack is the known best single-key key recovery attack on RC4 with respect to the efficiency. In addition, if we focus on specific keystreams similar to Teramura et al.'s attack, the 128-bit secret key can be recovered with efficiency of $2^{76.32}$, which is more efficient than Teramura et al.'s attack.

The paper is organized in the following way: in Section 2, we describe RC4 and predictive state. Furthermore, the number of predictive states is measured in this section. In Section 3, we introduce the previous weak keys, and Section 4 is the proposal of new weak keys. Two key recovery attacks are proposed in Section 5. We measured efficiencies of those attacks in the same section. Finally, we conclude this paper in Section 6.

2. RC4

In this section, we briefly review the algorithm of the stream cipher RC4.

2.1 Description of RC4

The stream cipher RC4 consists of two parameters and two algorithms. Parameters are n and L, and Algorithms are the KSA and the PRGA. n is the word length and L is the key length. 1 byte is n bits. The variable N, which is frequently used in RC4, is decided by n, $N = 2^n$. The KSA initializes an N-byte internal state by using an L-byte secret key K. In general, n is equal to 8 and L is equal to 16. In other words, N is equal to 256 and the key length is 128-bits (=16-bytes). In this paper, these parameters are used unless otherwise noted. Although, our theory is able to adapt for other parameters. The KSA initializes the internal state S using K, and the PRGA generates a pseudo-random numbers (called a keystream). Details of the KSA and the PRGA are described in Algorithm 1 and Algorithm 2.

The array S is permutations of $\{0, 1, 2, \ldots, 255\}$, the values don't overlap in S. In this paper, the KSA and the PRGA are distinguished by a star (*), which is attached to variables at the KSA. The internal state of RC4 consists of an N-byte permutation array S and two indices i and j. Variable t is used to clarify other variable's value at each round. Let $S_t[x]$ be the value of the array S at the index x and $S_t^{-1}[y]$ be the index of the value y in the array S after the t-th round in the PRGA.

2.2 Predictive State

Fluhrer and McGrew have observed stronger correlations between keystreams and the initial states of the PRGA, and introduced the notion as fortuitous state [12]. This is a special class of internal states defined by the values of i_t, j_t, and several state elements of array S at the t-th round, which predicts the keystream. Mantin and Shamir expanded and generalized the notion of fortuitous state as predictive state [8]. The predictive state was defined as follows.

Table 1 Comparison of the key recovery attacks. Roos' and Sepehrdad et al.'s attacks use one keystream for one attack (single-key setting), but Teramura et al.'s attack must continue throwing away keystreams until an objective keystream appears (multiple-key setting).

		Time complexity	Probability	Efficiency
Single-key setting	Roos [19]	2^{112}	$2^{-10.9}$	$2^{122.9}$
	Sepehrdad et al. [18] (probability optimized)	$2^{38.09}$	$2^{-87.90}$	$2^{125.99}$
	Sepehrdad et al. [18] (efficiency optimized)	1	$2^{-122.06}$	$2^{122.06}$
	Ours	$2^{96.36}$	$2^{-18.75}$	$2^{115.11}$
Multiple-key setting	Teramura et al. [20]	$2^{95.1}$	$2^{-7.1}$	2^{88}
	Ours	2^{65}	$2^{-11.32}$	$2^{76.32}$

Algorithm 1 Key Scheduling Algorithm

```
for x = 0 to N − 1 do
    S*_{−1}[x] ← x
end for
j*_{−1} ← 0
for t = 0 to N − 1 do
    i*_t ← t
    j*_t ← j*_{t−1} + S*_{t−1}[i*_t] + K[i*_t mod L] mod N
    S*_t ← Swapped S*_{t−1}(S*_{t−1}[i*_t] and S*_{t−1}[j*_t])
end for
```

Algorithm 2 Pseudo-Random Generation Algorithm

```
i_0 ← 0
j_0 ← 0
t ← 0
loop
    t ← t + 1
    i_t ← t mod N
    j_t ← j_{t−1} + S_{t−1}[i_t] mod N
    S_t ← Swapped S_{t−1}(S_{t−1}[i_t] and S_{t−1}[j_t])
    Output Z_t ← S_t[S_t[i_t] + S_t[j_t] mod N]
end loop
```

Definition 1 [8, Definition 1] *An a-state is a partially specified RC4 state, that includes i, j, and a* (*not necessarily consecutive*) *elements of S.*

Definition 2 [8, Definition 2] *Let A be an a-state for some a. Suppose that all the RC4 states that are compatible with A produce the same output word after r rounds. Then A is said to predict its r-th output.*

Definition 3 [8, Definition 3] *Let A be an a-state, and suppose that for some* $r_1, \ldots, r_b \leq 2N$*, A predicts the outputs of rounds* $r_1, \ldots, r_b$*. Then A is said to be b-predictive.*

In the PRGA, each Z_i is calculated from 3-states, $S[i], S[j]$ and $S[S[i] + S[j]]$. Their pointers i, j and $S[i] + S[j]$ are overlapped frequently. Therefore, as 2-predictive 2-state, there are cases that many keystreams can be predicted from less internal state. There is the condition $b \leq a$ between a and b, it was proven by Paul and Preneel [21]. In this paper, unless otherwise specifically noted, a predictive state has $i = 0$, $j = 0$.

In this study, we search thoroughly the b-predictive a-state under conditions of $b, a \leq 4$ and using only $S[0], S[1], \ldots, S[15]$. Those predictive states are guaranteed to be correlated weak keys. Incidentally, we use the searching algorithm shown by Teramura et al. [20]. Because of the much quantity of search, the total number of high parameters is not calculated. Nevertheless, we confirm the existence of the predictive state which has high parameters by limiting a search range. It is empirically known that the b-predictive b-state tend to have the straight positions such as $S_0[1], S_0[2], \ldots, S_0[b]$, the range $[-b, +2b]$ and the value of $S_0[1]$ is limited in $[1, b]$.

Now, there are derivation relations between predictive states which have other a. For example, there is the following 1-predictive 2-state:

Table 2 Number of predictive states which is lead from weak keys. "Total" includes the subset of other predictive state, and "w/o extension" is omitted them.

a	b	Total	w/o extension
2	1	540	540
2	2	1	1
3	1	72,386	68,105
3	2	593	4
3	3	2	2
4	1	20,221,906	18,769,291
4	2	335,941	107,365
4	3	5,558	682
4	4	18	2

$$\begin{cases} S_0[1] = 1 \\ S_0[2] = 2 \Rightarrow Z_1 = 2. \\ i_0 = 0,\ j_0 = 0 \end{cases} \tag{1}$$

The following 5-predictive 5-state is an expansion of the above predictive state.

$$\begin{cases} S_0[1] = 1 \\ S_0[2] = 2 \\ S_0[3] = 0 \\ S_0[4] = 254 \\ S_0[5] = 3 \\ i_0 = 0,\ j_0 = 0. \end{cases} \Rightarrow \begin{cases} Z_1 = 2 \\ Z_2 = 0 \\ Z_3 = 2 \\ Z_4 = 1 \\ Z_5 = 3. \end{cases} \tag{2}$$

In this case, the predictive state of the latter is a special case of the former. We checked each predictive state's relation and derived the number of primitive predictive states. **Table 2** is the result of it.

3. Previous Key Recovery Attack

Weak key is a class of special keys that can be attacked with higher efficiency than other keys. In this section, we introduce existing weak keys in RC4. Our weak key proposed in the next section is an extension of these.

3.1 Roos' Weak Key [19]

In 1995, Roos discovered that there was a strong association between Z_1 and $K[0], K[1], K[2]$ in high probability. Roos' weak key exists with $2^{-10.9}$ of probability in all keys, and can obtain the fragment 2 byte information of the key. Therefore, Roos' weak key can be decoded with time complexity of $2^{128-16} = 2^{112}$. Efficiency of Roos' weak key is

$$\frac{2^{112}}{2^{-10.9}} = 2^{122.9}. \tag{3}$$

Incidentally, the total number of Roos' weak key is $2^{117.1}$.

3.2 Teramura et al.'s Weak Key [20]

In 2011, Teramura et al. generalized the classes of weak keys using predictive state. Teramura et al. derived the condition of keys which leads predictive state with high probability, and defined those keys as Teramura et al.'s weak keys. Their weak keys can obtain the fragment a-byte information of the key from b-predictive a-state using $\{S[0], S[1], \ldots, S[a-1]\}$. And they discover that Roos' weak keys are subsets of Teramura et al.'s weak

keys using 1-predictive 2-states. Specifically, Roos' weak keys use the following predictive state:

$$\begin{cases} S_0[1] = 1 \\ S_0[2] = X \end{cases} \Rightarrow Z_1 = X. \tag{4}$$

Their weak key's efficiency depends on parameter a and b strongly. Among weak keys which Teramura et al. discovered, the most effective parameter is 5-predictive 5-state. When using only one weak key, Teramura et al.'s attack must continue throwing away keystreams until an objective keystream appears. Those weak keys exists with $2^{-7.1}$ in specific keystreams, which have particular patterns, and can obtain the 40-bit (= 5-byte) information of the key. Therefore, the time complexity is $2^{128-40} = 2^{88}$ and the probability is $2^{-7.1}$ in the most effective parameter. Efficiency of the most effective parameter is

$$\frac{2^{88}}{2^{-7.1}} = 2^{95.1}. \tag{5}$$

This efficiency is $2^{32.9}$ times greater than Roos' weak key.

3.3 Sepehrdad et al.'s Key Recovery Attack

In 2010, Sepehrdad et al. proposed the attack under this condition [18]. Sepehrdad et al.'s key recovery attack calculates the most frequent key from the plural biases using $Z_1, \ldots, Z_{47}$. Sepehrdad et al. gather up old biases and discover the new biases. For example, there is the following bias:

$$K[0] + K[1] - z_1 = 255 \tag{6}$$

This bias occurs with $1.04237/N$ of probability. Their attack bundles such biases and calculates the probable keys. When their attack be optimized to efficiency, the key can be calculated with the probability $2^{-122.06}$ without any key search, therefore the efficiency is $2^{122.06}$. Optimized to probability, the probability reduces to $2^{-87.90}$, however, the time complexity increases to $2^{38.09}$ and the efficiency is $2^{125.99}$.

4. New Weak Key

We further extend Teramura et al.'s weak keys by thoroughly analyzing the relation between the key and the initial state of the PRGA. Teramura et al.'s weak keys used only the b-predictive a-state with $S[0], \ldots, S[a-1]$, whereas our weak keys can use discrete predictive states, which have free positions in the range of $S[0], \ldots, S[15]$. In addition, we succeed in relaxing the condition in the KSA. Teramura et al.'s weak key was necessary to prepare a value of $S[x]$ at x-round of the KSA. In contrast, our weak key allows that some values are not prepared in x-round, and set those values by stochastic means at initial round of the PRGA. As a result, we define $2^{118.58}$ keys as weak keys.

4.1 Condition of Weak Key

$K[x]$ and $S_0[x]$ ($x = \{0, \ldots, L-1\}$) correlate to each other. Therefore, the keystreams which generated by $S_0[0], \ldots, S_0[L-1]$ contain more key informations than other keystreams. $S_{pre}[x_1], \ldots, S_{pre}[x_a]$, which defined by b-predictive a-state, conjecture c-byte key, $K[y_1], \ldots, K[y_c]$ ($a - b < c \leq a, \forall y_i \in \{x_1, \ldots, x_a\}$). The conditions of weak key are

$$\begin{aligned} & S^*_{k-1}[j^*_k] = S_{pre}[k] \\ \Leftrightarrow\ & j^*_k = S^{*-1}_{k-1}[S_{pre}[k]]\ (k \in \boldsymbol{y}) \\ \Leftrightarrow\ & K[k] = S^{*-1}_{k-1}[S_{pre}[k]] - S^*_{k-1}[k] - j^*_{k-1} \bmod N. \end{aligned} \tag{7}$$

In addition, $K[\bar{k}]$ does not satisfy the above equation ($\bar{k} \in \boldsymbol{x} \backslash \boldsymbol{y}$).

We introduce a simple example. When you choose the predictive state which is written in Eq. (2) and parameters as $c = 3$ and $\boldsymbol{y} = (1, 3, 4)$ which are selected at random, the weak key is written as follows:

$$\begin{cases} K[1] = S^{*-1}_0[1] - S^*_0[1] - j^*_0 \\ K[2] \neq S^{*-1}_1[2] - S^*_1[2] - j^*_1 \\ K[3] = S^{*-1}_2[0] - S^*_2[3] - j^*_2 \\ K[4] = S^{*-1}_3[254] - S^*_3[4] - j^*_3 \\ K[5] \neq S^{*-1}_4[3] - S^*_4[5] - j^*_4 \end{cases} \pmod N \tag{8}$$

S^*, j^* change by other K, so $K[1], \ldots, K[5]$ are not the same value at all times. These values are determined while operating the KSA.

4.2 Probability for Weak Keys

In this section, we explain how a weak key leads a b-predictive a-state. A predictive state is defined by S, i, and j. Please note that for the sake of ease, we limit $i = j = 0$, and consider only the S. In this paper, we define $P[E_{out}|E_{key}]$ as the probability that predictive state is generated from weak keys.

At first, we derive the probability that each $S_{y_i}[y_i]$ stored to the $S_{pre}[y_i]$ at round y_i in the KSA ($y_i \in \boldsymbol{y}$). Because $i^*_{y_i} = y_i$, the requirement is that the value pointed by $j^*_{y_i}$ is $S_{pre}[y_i]$. This condition is written as Eq. (7), and the probability is 1. Furthermore, $S_{pre}[y_i]$ should not move in $(N - y_i - 1)$-rounds until 0-th round in the PRGA that the predictive state demands. To put another way, the pointer j^* should not point to the y_i. This probability is

$$\left(\frac{N-1}{N}\right)^{N-y_i-1}. \tag{9}$$

Thus, c-values, $S_{pre}[y_1], \ldots, S_{pre}[y_c]$, are stored until the 0-th round in the PRGA. The remaining $(a-c)$-values are not fixed in the middle of the KSA, fit in the correct position by coincidence. Assuming that the value is moved at random, the probability that one value fits correctly is

$$\left(\frac{1}{N}\right). \tag{10}$$

Therefore, $P[E_{out}|E_{key}]$ is derived as the following equation:

$$\begin{aligned} P[E_{out}|E_{key}] &= \prod_{i=1}^{c} \left(\frac{N-1}{N}\right)^{N-y_i-1} \cdot \left(\frac{1}{N}\right)^{a-c} \\ &= \left(\frac{N-1}{N}\right)^{c(N-1)-\sum_{i=1}^{c} y_i} \left(\frac{1}{N}\right)^{a-c}. \end{aligned} \tag{11}$$

4.3 Probability from Keystream

In this section, it is defined $P[E_{key}|E_{out}]$ that the probability of the keystreams was extracted from the weak keys. $P[E_{key}|E_{out}]$ is calculated from $P[E_{out}|E_{key}]$ using Bayes' theorem:

$$P[E_{key}|E_{out}] = \frac{P[E_{key}]}{P[E_{out}]} \cdot P[E_{out}|E_{key}]. \tag{12}$$

If the generation probability of keys and keystream have no bias,

$$\begin{cases} P[E_{out}] = \left(\frac{1}{N}\right)^b, \\ P[E_{key}] = \left(\frac{1}{N}\right)^c \cdot \left(\frac{N-1}{N}\right)^{a-c}. \end{cases} \tag{13}$$

Therefore, using the $P[E_{out}|E_{key}]$ in Section 4.1,

$$P[E_{key}|E_{out}] = \left(\frac{N-1}{N}\right)^{a+c(N-2)-\sum_{i=1}^{c} y_i} \left(\frac{1}{N}\right)^{a-b}. \tag{14}$$

$P[E_{key}|E_{out}]$ is a value to be decided by a, b, c and $\sum_{i=1}^{c} y_i$ (rather than y_i itself). As a representative parameter, we show the case that $\boldsymbol{y} = (1, 2, \ldots, c)$ in **Table 3**. In our weak key, c byte partial keys are able to be predicted and other keys are recovered by only N^{L-c} times searching with probability of $P[E_{key}|E_{out}]$. Hence, the efficiency of our weak key, which is defined as the time complexity / probability, is calculated as

$$\frac{N^{L-c}}{P[E_{key}|E_{out}]}. \tag{15}$$

Table 4 is the representative example, it has same parameter of Table 3.

4.4 Number of Weak Keys

In this section, we derive the total number of our weak keys. Our weak keys exist for each predictive state a lot, and a lot of predictive states themselves exist, too.

At first, we derive the number of weak keys corresponding to each predictive state. Weak keys that belong to each predictive state are subdivided by parameter c and $\boldsymbol{y}$. Our weak key decides c byte keys, $K[y_1], \ldots, K[y_c]$, and we can choose c-positions, $y_1, \ldots, y_c$, freely from $\{0, 1, \ldots, a\}$. Therefore, it is $N^{L-c} \cdot \binom{a}{c}$ that the number of candidates of weak keys corresponding each parameter. The true weak keys, which lead a predictive state, exist in these candidates with $P[E_{out}|E_{key}]$. Hence, the number of weak keys when the parameter c is fixed is $N^{L-c} \cdot \binom{a}{c} \cdot P[E_{out}|E_{key}]$. The total number of weak keys from each predictive state can approximate as

$$\sum_{c=a-b+1}^{a} \left(N^{L-c} \cdot \binom{a}{c} \cdot P[E_{out}|E_{key}] \right). \tag{16}$$

Table 5 is the list of typical parameters which has same parameter as other tables, $\boldsymbol{y} = (1, 2, \ldots, c)$.

Next, we derive the total number of weak keys from all known predictive states using the number of each predictive state. Weak keys have inclusive relationship as well as predictive state. The weak keys which based extended predictive state are only a subset of weak keys based on the original predictive state. Therefore, when the total number of weak keys is calculated, extended weak keys must be omitted. **Table 6** is the total number of weak keys calculated from the number of original predictive states, which is written as "w/o extension" in Table 2. In Table 6, we calculate the exact number in consideration of the $\sum_{i=1}^{c} y_i$ of each predictive state. 1-predictive 2-states occupy the majority of weak keys, the total number does not change significantly even considering the parameters not in this table. The number of our weak keys is more than twice Teramura et al.'s weak keys.

5. Key Recovery Attack

In this section, we propose two key recovery attacks using our

Table 3 Typical $P[E_{key}|E_{out}]$. The cells with the parenthesis have a negative gain.

a	b	$c=1$	$c=2$	$c=3$	$c=4$
2	1	$(2^{-9.43})$	$2^{-10.86}$		
2	2	$2^{-1.43}$	$2^{-2.86}$		
3	1	$(2^{-17.43})$	$(2^{-18.86})$	$2^{-20.29}$	
3	2	$(2^{-9.43})$	$2^{-10.86}$	$2^{-12.29}$	
3	3	$2^{-1.43}$	$2^{-2.86}$	$2^{-4.29}$	
4	1	$(2^{-25.43})$	$(2^{-26.86})$	$(2^{-28.29})$	$2^{-29.70}$
4	2	$(2^{-17.43})$	$(2^{-18.86})$	$2^{-20.29}$	$2^{-21.70}$
4	3	$(2^{-9.43})$	$2^{-10.86}$	$2^{-12.29}$	$2^{-13.70}$
4	4	$2^{-1.43}$	$2^{-2.86}$	$2^{-4.29}$	$2^{-5.70}$

Table 4 Typical efficiency of our weak keys. Teramura et al.'s weak keys are used only $a = c$.

a	b	$c=1$	$c=2$	$c=3$	$c=4$
2	1	$(2^{129.43})$	$2^{122.86}$		
2	2	$2^{121.43}$	$2^{114.86}$		
3	1	$(2^{137.43})$	$(2^{130.86})$	$2^{124.29}$	
3	2	$(2^{129.43})$	$2^{122.86}$	$2^{116.29}$	
3	3	$2^{121.43}$	$2^{114.86}$	$2^{108.29}$	
4	1	$(2^{145.43})$	$(2^{138.86})$	$(2^{132.29})$	$2^{125.70}$
4	2	$(2^{137.43})$	$(2^{130.86})$	$2^{124.29}$	$2^{117.70}$
4	3	$(2^{129.43})$	$2^{122.86}$	$2^{116.29}$	$2^{109.70}$
4	4	$2^{121.43}$	$2^{114.86}$	$2^{108.29}$	$2^{101.70}$

Table 5 The number of weak keys from each predictive state. The number of Teramura et al.'s weak keys was decided only by a, but our weak keys increase even by b because of the flexibility of the c.

a	b	Ours	Teramura et al.'s ($c = a$)
2	1	$2^{109.14}$	$2^{109.14}$
2	2	$2^{111.81}$	$2^{109.14}$
3	1	$2^{99.71}$	$2^{99.71}$
3	2	$2^{102.89}$	$2^{99.71}$
3	3	$2^{104.65}$	$2^{99.71}$
4	1	$2^{90.30}$	$2^{90.30}$
4	2	$2^{93.84}$	$2^{90.30}$
4	3	$2^{96.07}$	$2^{90.30}$
4	4	$2^{97.34}$	$2^{90.30}$

Table 6 The strict total number of weak keys. The total number of Teramura et al.'s weak keys is derived by the number of predictive states in [20] and Table 5 in this paper. Because Teramura et al. didn't search the predictive states which have low parameter, some cells are not clear.

a	b	w/o extended	All	Teramura et al.'s
2	1	$2^{118.22}$	$2^{118.22}$	$2^{117.28}$
2	2	$2^{111.81}$	$2^{111.81}$	$2^{109.14}$
3	1	$2^{115.82}$	$2^{115.91}$	–
3	2	$2^{104.90}$	$2^{112.11}$	$2^{105.82}$
3	3	$2^{106.43}$	$2^{106.43}$	0
4	1	$2^{114.50}$	$2^{114.61}$	–
4	2	$2^{110.59}$	$2^{112.22}$	$2^{107.45}$
4	3	$2^{106.33}$	$2^{109.36}$	$2^{100.54}$
4	4	$2^{100.44}$	$2^{103.61}$	$2^{93.30}$
Total		$2^{118.58}$		$2^{117.29}$

weak keys. One is a single-key attack, which is a case of performing an attack on any keystream. This is the standard key recovery setting. In this attack, efficiency changes for the predictive state corresponding by the keystream. It is possible to recover the key with efficiency of $2^{115.11}$. The other attack uses only most effective predictive state, the setting is used by Teramura et al. This attack focus on specific keystreams which have same output as the predictive state. It is necessary to observe a large amount of keystream group, but it is possible to recover the key with efficiency of $2^{76.32}$.

5.1 Single-key Attack

Our attack uses the list of predictive states which predict two bytes keystream, namely, Z_1 and Z_2. All keystream match any one of listed predictive states. The making method of the list is shown in Appendix A.1. Because each pattern has plural predictive states, the list adopts the only the most efficient predictive state. As a result of exploring the patterns of all, all keystreams corresponding to the 2-predictive 5-state shown below even worst:

$$\begin{cases} S_0[1] = 2 \\ S_0[2] = 13 \\ S_0[4] = 12 \\ S_0[14] = Z_2 \\ S_0[15] = Z_1 \\ i_0 = 0,\ j_0 = 0. \end{cases} \tag{17}$$

Efficiency of this predictive state is $2^{119.00}$, this is the worst efficiency in our attack. Also, when it is $Z_1 = Z_2$ can be applied following 2-predictive 4-state:

$$\begin{cases} S_0[0] = 4 \\ S_0[1] = 0 \\ S_0[2] = 2 \\ S_0[4] = Z_1 = Z_2 \\ i_0 = 0,\ j_0 = 0. \end{cases} \tag{18}$$

Others, more effective predictive states exist in $0 \le Z_1, Z_2 \le 15$. Under the condition of $b = 2$, 2-predictive 2-state in case of $Z_1 = Z_2 = 0$ is the best, its efficiency is $2^{114.86}$. **Figure 1** is the graph that shows efficiencies of every Z_1 and Z_2. To unify these, the list of predictive states is made for every Z_1 and Z_2 pattern. The attacker chooses the predictive state corresponding to the keystream which be observed among the list. The average efficiency of this attack is calculated by the average time complexity / the average success probability. The average time complexity and the average success probability are the arithmetical means of each Z_1, Z_2. We explain the details about the time complexity. Let $a_{(Z_1,Z_2)}$ be defined the parameter a for each Z_1, Z_2, which is changed by each predictive state. In each Z_1, Z_2, the time complexity is written as $256^{16-a_{(Z_1,Z_2)}}$. Therefore, the time complexity as the arithmetical mean is

$$\frac{1}{256^2} \sum_{Z_1=0}^{255} \sum_{Z_2=0}^{255} 256^{16-a_{(Z_1,Z_2)}}. \tag{19}$$

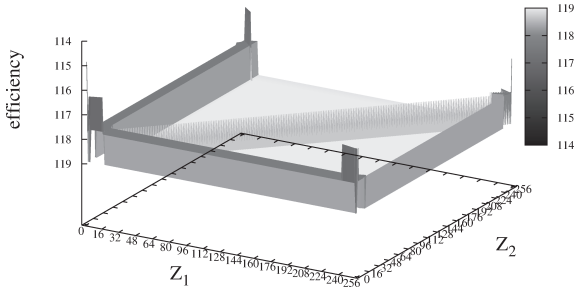

Fig. 1 The efficiency map of the predictive state. The vertical axis shows efficiency with a logarithm. As for z-axis being high, the efficiency is little. Because the efficiency means average time complexity, the less efficiency the better. $Z_1 = Z_2 = 0$ is the most powerful pattern, and the worst efficiency is 2^{119}.

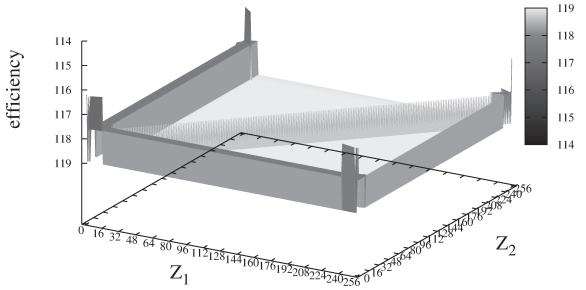

Fig. 2 Efficiency map without Teramura et al.'s weak keys. Compared with Fig. 1, $Z_1 = Z_2 = 0$ decrease in the efficiency. However, most of the list does not change. RC4 is not available safely by measures to only Teramura et al.'s weak keys.

The probability is calculated equally. The average time complexity of all patterns is $2^{96.36}$. The average of the attack success probability, namely the success probability for every keystream is $2^{-18.75}$. Therefore, the efficiency for the whole is $2^{96.36}/2^{-18.75} = 2^{115.11}$.

We consider if all Teramura et al.'s weak keys are excluded. In other words, the predictive states using $\boldsymbol{y} = (0, 1, \ldots, a)$ are excluded. **Figure 2** is the result. Because the 2-predictive 2-state, which is most efficient parameter, corresponds to Teramura et al.'s weak keys, the overall efficiency declines a little. Although, it is not affected by most of predictive states in the list, because Teramura et al.'s weak keys have the lowest $\sum_{i=1}^{c} y_i$ and not be adapted to the list. As a result, even if Teramura et al.'s weak keys are excluded, the average time complexity maintains $2^{94.13}$, the success probability is $2^{-22.53}$, and the average efficiency is $2^{116.67}$.

Our attack's average efficiency ($2^{115.11}$) is smaller than Sepehrdad et al.'s efficiency ($2^{122.06}$). Hence, we propose a useful attack in key recovery attack on RC4.

5.2 Multiple-key Attack

In this section, we consider the case that relieved a condition than Section 5.1. In Section 5.1, we propose the attack to every keystream, but the most effective attack using our weak keys is not so. It is effective to use only the most excellent predictive state without using the plural predictive states. Namely, the attack searches a key only to the keystream which conform to predict. To this attack, it is necessary to observe a large quantity keystream as a premise.

This attack is classified into the keystream choice stage and the key search stage. In the keystream choice stage, it is de-

termined whether the observed keystream accords with the prediction of the most effective predictive state. When we assume time complexity 1 per one comparison, the time complexity necessary to discover the keystream according in the prediction of the b-predictive a-state is N^b. In the key search stage, the c-bytes during 16 bytes key are fixed and the remaining keys are searched. $c = a$ is the most effective, and it is not necessary to search for $c < a$. The time complexity of the key search stage is N^{L-c}, and the success probability is $P[E_{key}|E_{out}]$, which is shown in Eq. (14). Therefore, the efficiency that adds up two stages is

$$\frac{N^b + N^{L-c}}{P[E_{key}|E_{out}]}. \tag{20}$$

In 128-bit key RC4, the most powerful parameter is 8-predictive 8-state ($c = 8$). We confirmed that the number of 8-predictive 8-states is at least 10,000 by a restrictive search. Although, it is not necessary to grasp all and is enough any one for the attack. Show an example below:

$$\begin{cases} S_0[1] = 2 \\ S_0[2] = 3 \\ S_0[3] = 1 \\ S_0[4] = 255 \\ S_0[5] = 6 \\ S_0[6] = 251 \\ S_0[7] = 0 \\ S_0[8] = 4 \\ i_0 = 0,\ j_0 = 0. \end{cases} \Rightarrow \begin{cases} Z_1 = 6 \\ Z_2 = 3 \\ Z_3 = 0 \\ Z_4 = 255 \\ Z_5 = 4 \\ Z_6 = 3 \\ Z_7 = 6 \\ Z_8 = 4. \end{cases} \tag{21}$$

In this example, the attacker can discover one keystream agreeing with the 8-predictive 8-state by comparing 2^{64} kinds keystream. In addition, time complexity of the key search stage is 2^{64} and success probability is $2^{-11.32}$. Therefore, efficiency is

$$\frac{2^{64} + 2^{64}}{2^{-11.32}} = 2^{76.32}. \tag{22}$$

This efficiency is the best value of our attack.

Time-memory-data trade-off attack is a general attack model under the conditions of this section. When the attacker observes 2^{64} kinds of keystream and hold all in memory, one of 2^{64} random keys produces a keystream which was observed. The efficiency of time-memory-data trade-off attack is $2^{64} + 2^{64} = 2^{65}$. However, the memory space that 2^{64} kinds of keystreams can keep is essential to this attack. In contrast, our attack does not store any keystream, works only with the data of the predictive state. In other words, our attack dispensed with memory by decreasing efficiency to $2^{76.32}$.

6. Conclusion

In this paper, we expanded weak-key space of RC4. By thoroughly analyzing the relation between the key and the initial state of the PRGA, we found new classes of predictive states which are utilized for key recovery attacks. In addition, we succeed in relaxing the condition for leading a part of the secret key from predictive state by utilizing probabilistic events in the KSA. As a result, we showed the total number of our weak keys is $2^{118.58}$ at least, whose number is more than twice the number of Teramura et al.'s weak keys. Given *any* keystream, our weak-key attack can recover a 128-bit secret key with efficiency of $2^{115.11}$. Note that our attack is the known best *single-key* key recovery attack on RC4 with respect to the efficiency. In addition, even if previous weak keys are removed from RC4, $2^{117.82}$ our weak keys remains behind and the attack efficiency is still $2^{116.67}$. Those mean RC4 has potential vulnerabilities, unless otherwise our weak keys are removed. If we focus on specific keystreams similar to Teramura et al.'s attack, the 128-bit secret key can be recover with efficiency of $2^{76.32}$. Therefore, the evaluation of our weak keys in this paper is useful as the security assessment of RC4.

Acknowledgments This work was supported in part by Grant-in-Aid for Scientific Research (C) (KAKENHI 23560455) for Japan Society for the Promotion of Science, and Cryptography Research and Evaluation Committee (CRYPTREC).

References

[1] Schneier, B. and Sutherland, P.: *Applied cryptography: Protocols, algorithms, and source code in C*, John Wiley & Sons, Inc. (1995).
[2] Frier, A., Karlton, P. and Kocher, P.: The SSL 3.0 protocol, *Netscape Communications Corp*, Vol.18, p.2780 (1996).
[3] IEEE Std 802.11-1997: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications (1997).
[4] Alliance, W.: Wi-Fi protected access (2003), available from ⟨http://www.weca.net/opensection/protected_access.asp⟩.
[5] Fluhrer, S., Mantin, I. and Shamir, A.: Weaknesses in the key scheduling algorithm of RC4, *Selected Areas in Cryptography*, pp.1–24, Springer (2001).
[6] Vaudenay, S. and Vuagnoux, M.: Passive–only key recovery attacks on RC4, *Selected Areas in Cryptography*, pp.344–359, Springer (2007).
[7] Sepehrdad, P., Sušil, P., Vaudenay, S. and Vuagnoux, M.: Smashing WEP in A Passive Attack, *Fast Software Encryption-FSE* (2013).
[8] Mantin, I. and Shamir, A.: A practical attack on broadcast RC4, *Fast Software Encryption*, pp.87–104, Springer (2002).
[9] Maitra, S., Paul, G. and Gupta, S.S.: Attack on broadcast RC4 revisited, *Fast Software Encryption*, pp.199–217, Springer (2011).
[10] Isobe, T., Ohigashi, T., Watanabe, Y. and Morii, M.: Full Plaintext Recovery Attack on Broadcast RC4, *Fast Software Encryption-FSE* (2013).
[11] Golić, J.D.: Linear statistical weakness of alleged RC4 keystream generator, *Advances in Cryptology-EUROCRYPT'97*, pp.226–238, Springer (1997).
[12] Fluhrer, S. and McGrew, D.: Statistical analysis of the alleged RC4 keystream generator, *Fast Software Encryption*, pp.66–71, Springer (2001).
[13] Mantin, I.: Predicting and distinguishing attacks on RC4 keystream generator, *Advances in Cryptology–Eurocrypt 2005*, p.551 (2005).
[14] Matsui, M.: Key collisions of the RC4 stream cipher, *Fast Software Encryption*, pp.38–50, Springer (2009).
[15] Chen, J. and Miyaji, A.: Generalized RC4 key collisions and hash collisions, *Security and Cryptography for Networks*, pp.73–87, Springer (2010).
[16] Knudsen, L., Meier, W., Preneel, B., Rijmen, V. and Verdoolaege, S.: Analysis methods for (alleged) RC4, *Advances in Cryptology-ASIACRYPT'98*, pp.327–341, Springer (1998).
[17] Maximov, A. and Khovratovich, D.: New state recovery attack on RC4, *Advances in Cryptology–CRYPTO 2008*, pp.297–316, Springer (2008).
[18] Sepehrdad, P., Vaudenay, S. and Vuagnoux, M.: Discovery and Exploitation of New Biases in RC4, *Selected Areas in Cryptography*, pp.74–91, Springer (2011).
[19] Roos, A.: A class of weak keys in the RC4 stream cipher (1995).
[20] Teramura, R., Ohigashi, T., Kuwakado, H. and Morii, M.: Generalized Classes of Weak Keys on RC4 Using Predictive State, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.94, No.1, pp.10–18 (2011).
[21] Paul, S. and Preneel, B.: Analysis of non-fortuitous predictive states of the RC4 keystream generator, *Progress in Cryptology-INDOCRYPT 2003*, pp.67–70 (2003).

Appendix

A.1 Predictive State Searching Algorithm from Keystream

In this appendix, we show the predictive state search algorithm, which correspond with given keystream as much as possible. The basic idea is simple, the PRGA is operated without determining the value of S, and values are not fixed unless they are pressed by necessity. Namely, the necessities are the track of pointer j and output Z_i. Because this algorithm target discoveries the predictive state which is led by weak key, the search range is limited to $S_0[0], S_0[1], \ldots, S_0[L-1]$. S_p is the pointer array, which use to make the value of S_0 clear. $V[x]$ is the array which manage the value x as used or unused in S. This algorithm is the tree search which has two kinds of leaves, SUCCESS or FAILURE. SUCCESS means no room for the search or the further search is difficult. FAILURE means that the search is in conflict. Incidentally, this algorithm cannot detect the predictive state which has discrete keystream, e.g., Z_1, Z_2 and Z_4 without Z_3.

Editor's Recommendation

Mantin et al. pointed out vulnerability of WEP arisen from the initialization of RC4, which tries to guess a part of secret key from the output sequences. The size of such weak keys in RC4 has been extended by many subsequent researches. This paper revisited some equations in the previous literatures, and showed that the space of the weak key becomes larger than that of the previously investigated ones. Indeed the authors extended the size of the weak key in RC4 by 8 times, namely from $2^{100.91}$ to $2^{103.78}$. This attack is not a real threat in practice, but the theoretical contribution is worth recommending for publication in JIP.

(Program Chair of Computer Security Symposium 2012,
Tsuyoshi Takagi)

Algorithm 3 Predictive state searching algorithm

Require: $\{Z_1, Z_2, \ldots, Z_{max}\}$ are the target keystream.
1: $S_0[x]$ $(0 \le x \le N-1) \leftarrow$ "undefined"
2: $S_0[x]$ $(0 \le x \le L-1)$ are tagged "available"
3: $S_0[x]$ $(L \le x \le N-1)$ are tagged "unavailable"
4: $S_p[x] \leftarrow x$ $(0 \le x \le N-1)$
5: $V[x]$ $(0 \le x \le N-1) \leftarrow$"unused"
6: $a = 0, b = 0$
7: $i = 0, j = 0$
8: Set label: "Main Loop"
9: $i = i + 1 \bmod N$
10: **if** $i > max$ **then**
11: return(SUCCESS)
12: **else if** $S_0[S_p[i]] \neq$"available" **then**
13: return(SUCCESS)
14: **end if**
15: **if** $S_0[S_p[i]] =$"undefined" **then**
16: **fork the search** as many as the number of "unused" in V.
Copy all variables from parent.
$n \leftarrow$ a "unused" number
$S_0[S_p[i]] \leftarrow n$
$V[n] \leftarrow$"used"
$a++$.
17: **end if**
18: $j \leftarrow j + S_0[S_p[i]] \bmod N$
19: Swap$(S_p[i], S_p[j])$
20: **if** $S_0[S_p[i]]$ tagged "unavailable" **then**
21: return(SUCCESS)
22: **end if**
23: **if** $S_0[S_p[i]] \neq$"undefined" **then**
24: Go to "Search Value"
25: **end if**
26: **if** $V[Z_i] =$"unused" **then**
27: **fork the search** same as Line 16
28: Go to "Search Value"
29: **end if**
30: **if** $V[S^{-1}[Z_i] - S_0[S_p[j]]] =$"used" **then**
31: return(FAILURE)
32: **end if**
33: $S_0[S_p[i]] \leftarrow S^{-1}[Z_i] - S_0[S_p[j]]$
34: $V[S_0[S_p[i]]] \leftarrow$"used"
35: $a++, b++$
36: Go to "Main Loop"
37: Set label: "Search Value"
38: $k \leftarrow S_0[S_p[i]] + S_0[S_p[j]]$
39: **if** $S_0[S_p[k]]$ tagged "unavailable" **then**
40: return(SUCCESS)
41: **else if** $S_0[S_p[k]] =$"undefined" **then**
42: **if** $V[Z_i] =$"used" **then**
43: return(FAILURE)
44: **end if**
45: $S_0[S_p[k]] \leftarrow Z_i$
46: $a++, b++$
47: Go to "Main Loop"
48: **else if** $S_0[S_p[k]] = Z_i$ **then**
49: $b++$
50: Go to "Main Loop"
51: **end if**
52: return(FAILURE)

Atsushi Nagao received his B.E. degree from Kobe University, Japan in 2012. Since 2012, he has been a master's student in Graduate School of Engineering, Kobe University. His current research interests are in cryptography and information security. He received the CSS2012 Student Paper Award from CSEC group of IPSJ in 2012.

Toshihiro Ohigashi received his B.E. and M.E. degrees from University of Tokushima, Japan, and Ph.D. degree from Kobe University in 2002, 2004, and 2008, respectively. Since 2008, he has been an Assistant Professor in Information Media Center, Hiroshima University. His current research interests include cryptography, information security, and network protocol. He received the SCIS 20th Anniversary Award from ISEC group of IEICE in 2003. He is a member of IEICE.

Takanori Isobe received his B.E., M.E., and Ph.D. degrees from Kobe University, Japan, in 2006 and 2008, and 2013 respectively. He joined the Sony Corporation in 2008. His current research interests include information security and cryptography. He received the SCIS Paper Award from ISEC group of IEICE in 2008. He received the FSE 2011 Best Paper Award from IACR in 2011.

Masakatu Morii received his B.E. degree in electrical engineering and M.E. degree in electronics engineering from Saga University, Saga, Japan, and D.E. degree in communication engineering from Osaka University, Osaka, Japan, in 1983, 1985, and 1989, respectively. From 1989 to 1990 he was an Instructor in the Department of Electronics and Information Science, Kyoto Institute of Technology, Japan. From 1990 to 1995 he was an Associate Professor at the Department of Computer Science, Faculty of Engineering at Ehime University, Japan. From 1995 to 2005 he was a Professor at the Department of Intelligent Systems and Information Science, Faculty of Engineering at the University of Tokushima, Japan. Since 2005, he has been a Professor at the Department of Electrical and Electronics Engineering, Faculty of Engineering at Kobe University, Japan. His research interests are in error correcting codes, cryptography, discrete mathematics and computer networks and information security. He is a member of IEEE.