PAPER *Special Section on Cryptography and Information Security*

Cryptanalysis of Reduced-Round RC6 without Whitening[**]

Atsuko MIYAJI[†*a)], ***Regular Member and*** **Masao NONAKA**[††], ***Nonmember***

SUMMARY We investigate the cryptanalysis of reduced-round RC6 without whitening. Up to now, key recovery algorithms against the reduced-round RC6 itself, the reduced-round RC6 without whitening, and even the simplified variants have been infeasible on a modern computer. In this paper, we propose an efficient and feasible key recovery algorithm against reduced-round RC6 without whitening. Our algorithm is very useful for analyzing the security of the round-function of RC6. Our attack applies to a rather large number of rounds. RC6 without whitening with r rounds can be broken with a success probability of 90% by using $2^{8.1r-13.8}$ plaintexts. Therefore, our attack can break RC6 without whitening with 17 rounds by using $2^{123.9}$ plaintexts with a probability of 90%.

key words: *block cipher, RC6, cryptanalysis, χ^2 attack*

1. Introduction

RC6 [13] is a block cipher, which is constructed by only simple arithmetic such as multiplication, addition, bit-wise exclusive-or (XOR), and data-dependent rotation. Therefore, RC6 can be implemented efficiently in software with a small amount of memory. RC6 is submitted as a candidate for NESSIE [12], and recently has been selected to proceed to the next stage. RC6-32/r/16 means that four 32-bit-word plaintexts are encrypted by r rounds with 16 byte keys. RC6 is the next version of RC5 [14], which consists of only addition, bit-wise exclusive-or (XOR), and data-dependent rotation. RC5 also includes a data-dependent rotation, which is much improved in RC6 in such a way that it is determined by all 32 bits of both data and subkey, but not 5 bits. Such an efficient improvement makes RC6 more secure because it is difficult to handle the rotation with specific plaintexts. Compared with various attacks against RC5 [1], [2], [4]–[7], [11], any key recovery algorithm against RC6 [2], [3], [8] requires much memory and work even in the case of a low round. Multiple linear cryptanalysis is applied to RC6 with 32-byte keys [16], but it has not been applied to RC6 with 16-byte keys.

Manuscript received March 23, 2002.
Manuscript revised July 6, 2002.
Final manuscript received September 20, 2002.

[†]The author is with Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.
[††]The author is with Matsushita Electric Industrial Co., LTD., Kadoma-shi, 571-8501 Japan. This work was conducted when he was with JAIST.
[*]Presently, with University of California, Davis.
a) E-mail: miyaji@jaist.ac.jp

[**]A preliminary version was presented at SCIS'2002.

Correlation attack makes use of correlations between an input and an output, which is measured by the χ^2 test; the specific rotation in RC6 is considered to cause the correlations between the corresponding two 10-bit integer values. Correlation attack consists of two parts, the distinguishing algorithm and the key recovery algorithm. The distinguishing algorithm has only to handle plaintexts in such a way that the χ^2-value of a part of the ciphertext assumes a significantly higher value. On the other hand, the key recovery algorithm has to rule out all false keys, and single out exactly a correct key by using the χ^2-value. However, only the distinguishing algorithm has been investigated up to now [4], [8]. That is, only a high χ^2-value has been focused, which is experimentally computed on the average of keys.

In [8], correlation attacks are applied to recover subkeys from the 1st subkey to the final subkey by handling a plaintext (A_0, B_0, C_0, D_0) in such a way that the χ^2-value after one round becomes a significantly higher. However, unfortunately, their key recovery algorithm has not been executed yet although their distinguishing algorithm has been implemented. Because their algorithm is forced to recover all 32 bits of the first subkey. It thus requires $2^{62.2}$ works with 2^{42} memory even in the case of RC6 with 5 rounds. In a realistic sense, it would be infeasible to employ such an algorithm on a modern computer. This is why their key recovery algorithm is estimated using only the results of the distinguishing algorithm. Their key recovery algorithm is roughly summarized as follows. 1. Choose a plaintext in such a way that the least significant five bits of A_0 and C_0, $lsb_5(A_0)$ and $lsb_5(C_0)$, are fixed, 2. For a plaintext and the corresponding 2^{27} first subkeys that lead to the zero rotation in the first round, compute the χ^2-value of concatenation of $lsb_5(A_{r+1})$ and $lsb_5(C_{r+1})$ of an output after r rounds, $(A_{r+1}, B_{r+1}, C_{r+1}, D_{r+1})$, 3. Output a subkey with the highest χ^2-value as the first subkey. Their key recovery algorithm is based on the idea that the χ^2-value is significantly high if a plaintext is suitably fixed so that one (or both) of the data-dependent rotations in the first round is zero. It works exactly as a distinguishing algorithm, but, as a key recovery algorithm, it is unlikely to rule out all false keys well for the following reason: the data-dependent rotation depends on all bits of the 32-bit subkey, however, the amount of information on data-dependent rotation is only 5 bits.

Fixing the first round rotation to zero merely fixes the 5-bit information on the first subkey but not to all its 32bits. In fact, for a plaintext, there are 2^{27} first subkeys that lead to zero rotation. This is why the above algorithm is unlikely to rule out all false keys. We also note that the number of available plaintexts for each key in their attack is reduced to 2^{118}.

In [11], a correlation attack against RC5 was proposed based on the same idea of fixing the first round rotation as in [8]. They reported three interesting new results: 1. their algorithm can search every four bits of a subkey in the final round; 2. their algorithm can recover subkeys with a high probability and a rather low χ^2-value; 3. an algorithm, applying the idea in [8] to RC5, cannot recover subkeys with high probability although the χ^2-value is extremely high. Their results indicate that not all bits but a few bits of subkeys can be recovered under the condition of fixing the first round rotation, and that a good distinguishing algorithm is not necessarily a good key recovery algorithm.

RC6 consists of three parts, pre-whitening, r-round iterations of round function, and post-whitening. In this paper, we focus on the round function of RC6, RC6 without whitening. Here we call RC6 without whitening, simply RC6W. We propose a feasible key recovery algorithm for the reduced-round RC6W for the first time. We improve the distinguishing algorithm in such a way that the χ^2-values for output become significantly high with less constraint of plaintexts, and then improve the key recovery algorithm in such a way that the variance of the χ^2-value is reduced. We know that output of RC6 is highly unlikely to be uniformly distributed if B_0 or D_0 of a plaintext (A_0, B_0, C_0, D_0) introduces zero rotation in the 1st round, and $lsb_5(A_0)$ and $lsb_5(C_0)$ are fixed [8]. More generally, we investigate how output after r rounds, both A_{r+1} and C_{r+1}, depends on a chosen plaintext, and experimentally find the following feature of RC6: the χ^2-values for the concatenation of $lsb_5(A_{r+1})$ and $lsb_5(C_{r+1})$ of an output after r rounds become significantly high if both the least significant 5 bits of the first and third words before addition to each 1st-round subkey are merely fixed. This means that we can use any plaintext by classifying them into groups with the same condition, and thus, the number of available plaintext is 2^{128}, which is very useful for distinguishing RC6 without pre-whitening.

We improve the key recovery algorithm by taking full advantage of the above feature, that is, the χ^2-values become significantly high for any group. As mentioned above, only the high average of the χ^2-value has been discussed. However, we also direct our attention to the variance of the χ^2-value. We compute the χ^2-value not flatly for all plaintext but for plaintexts in each group, and then compute the average of these χ^2-values. As a result, the variance of the χ^2-value is reduced, and the key recovery algorithm is expected to rule out all false keys. The main points of our feasible key recovery algorithm are as follows.

1. Use any plaintext by classifying it into groups,
2. Compute the χ^2-value of an output for plaintexts in each group, and then compute the average of these χ^2-values.

We also present three key recovery algorithms, which reflect the effect of computing the χ^2-value on each classified group. By employing our attack, RC6W with 5 rounds can be broken within 20 minutes on PPC 604e/332 MHz using 2^{27} plaintexts and 2^{26} memory. RC6W with r rounds can be broken with a success probability of 90%, by using $2^{8.1r-13.8}$ plaintexts. As a result, our attack can break RC6W with 17 rounds using $2^{123.9}$ plaintexts with a probability of 90%. Our algorithm can work faster than an exhaustive key search for the 128-bit key with a feasible memory size, 2^{26}. In [13], a two-register version for RC6 is also described, called RC6-64 in this paper. RC6-64 is oriented to 64-bit architecture, and plaintexts consist of two 64-bit words. The size of subkeys in RC6-64 is 64 bits. Hense the security level of one round in RC6-64, the size of subkeys, is estimated to be equal to that in RC6-32, which has two 32-bit subkeys in one round. Furthermore the round function of RC6-64 is almost the same structure as that of RC6. Therefore it is very useful for discussing the difference of each security level of each round function. By applying our attack to RC6-64 without whitening with r rounds, it can be broken with a success probability of 90% using $2^{5.0r-8.2}$ plaintexts. As a result, our attack can break RC6-64 without whitening with 27 rounds using $2^{126.8}$ plaintexts with a probability of 90%. The weak point of RC5 is thought to a data dependent rotation, which is defined by only a 5-bit subkey and data, but not the data structure of two words. Although the weakness of data-dependent rotation is improved in both RC6 and RC6-64, RC6-64 is much weaker than RC6. From our results, we see that the data structure of RC6, 4-word plaintext, also makes the security high.

This paper is organized as follows. Section 2 summarizes the notation and definitions used in this paper. Section 3 describes some experimental results including the above features of RC6. Section 4 presents the chosen plaintext algorithm, Algorithms 2 and 3. Section 5 discusses how to extend the chosen plaintext algorithm to the known plaintext algorithm, Algorithm 4. Section 6 applies Algorithm 4 to a two-register version for RC6, and discusses the difference between the original RC6 and a two-register version for RC6 from a security point of view.

2. Preliminary

This section denotes some notations, definitions, and experimental remarks. In this paper, RC6-32, the AES submission version, is simply denoted as RC6. First we

Table 1 χ^2-distributions for various degrees of freedom.

Level	0.50	0.60	0.70	0.80	0.90	0.95	0.99
31 degrees of freedom	30.34	32.35	34.60	37.36	41.42	44.99	52.19
63 degrees of freedom	62.33	65.20	68.37	72.20	77.75	82.53	92.01
255 degrees of freedom	254.33	260.09	266.34	273.79	284.34	293.25	310.46
1023 degrees of freedom	1022.33	1033.83	1046.23	1060.86	1081.38	1098.52	1131.16

describe the RC6 algorithm after defining the following notations.

$+, \boxplus$:	addition mod 2^{32}
$-, \boxminus$:	subtraction mod 2^{32}
$\times$:	multiplication mod 2^{32}
$\oplus$:	bit-wise exclusive OR
r :	number of (full) rounds
$a \lll b$:	cyclic rotation of a to the left by b bits
$a \ggg b$:	cyclic rotation of a to the right by b bits
S_i:	i-th subkey
$lsb_n(X)$:	least significant n bits of X
X^i :	i-th bit of X
$X^{[i,j]}$:	from the i-th bit to the j-th bit of X $(i > j)$
$\overline{X}$:	bit-wise inversion of X
$f(X)$:	$X \times (2X+1)$
$F(X)$:	$f(X) \pmod{2^{32}} \lll 5$

We denote the least significant bit (LSB) to the 1st bit, and the most significant bit (MSB) as the 32-nd bit for any 32-bit element. RC6 encryption is defined as follows.

Algorithm 1 (Encryption with RC6):

```
1.  $A_1 = A_0$; $B_1 = B_0 + S_0$;
    $C_1 = C_0$; $D_1 = D_0 + S_1$
2.  for $i = 1$ to $r$ do:
    $t = F(B_i)$; $u = F(D_i)$;
    $A_{i+1} = B_i$; $B_{i+1} = ((C_i \oplus u) \lll t) + S_{2i+1}$;
    $C_{i+1} = D_i$; $D_{i+1} = ((A_i \oplus t) \lll u) + S_{2i}$
3.  $A_{r+2} = A_{r+1} + S_{2r+2}$; $B_{r+2} = B_{r+1}$;
    $C_{r+2} = C_{r+1} + S_{2r+3}$; $D_{r+2} = D_{r+1}$.
```

Parts 1 and 3 of Algorithm 1 are called pre-whitening and post-whitening, respectively. We call the version of RC6 without pre-whitening or post-whitening as simply RC6W or RC6 without whitening.

We make use of a χ^2-tests for distinguishing a random sequence from a nonrandom sequence [6], [8], [9]. Let $X = X_0, ..., X_{n-1}$ be a sequence with $\forall X_i \in \{a_0, \cdots, a_{m-1}\}$. Let $N_{a_j}(X)$ be the number of X_i which equals a_j. The χ^2-statistic of X, $\chi^2(X)$, gives an estimate of the difference between X and the uniform distribution as follows: $\chi^2(X) = \frac{m}{n}\sum_{i=0}^{m-1}\left(N_{a_i}(X) - \frac{n}{m}\right)^2$. Table 1 presents each threshold for 31, 63, 255 and 1023 degrees of freedom. For example, (level, χ^2)=(0.95, 44.99) for 31 degrees in Table 1 means that the value of the χ^2-statistic exceeds 44.99 with the probability of 5% if the observation X is uniform. In this work, we adopt these four degree of freedom. For precision, we often discuss the χ^2-statistic for any degree based on the level. We set the level to 0.95 in order to distinguish a nonrandom sequence X from a random sequence.

In our experiments, all plaintexts are generated using the m-sequence [10]. For example, Algorithms 2, 3, and 4 uses 108-, 113- and 128-bit random numbers generated by the m-sequence, respectively. The platforms are IBM RS/6000 SP (PPC 604e/332 MHz $\times$ 256) with memory of 32 GB.

3. χ^2-Statistic of RC6

In this section, we investigate how to attain to much stronger biases with less constraint of plaintexts. In [8], if plaintexts (A_0, B_0, C_0, D_0) are chosen in such a way that both $lsb_5(A_0)$ and $lsb_5(C_0)$ are fixed, and both B_0 and D_0 introduce zero rotation in the 1st round, then the ciphertexts lead to much stronger biases. However, such a condition is a rather strict constraint because the number of plaintexts satisfying with such conditions are reduced to 2^{108}. We investigate other conditions that have almost the same effect with less constraint of plaintexts. To observe this, we conduct the following experiments.

Test 1: χ^2-test on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ in the case where both B_0 and D_0 induce zero rotation in the 1st round; $lsb_5(A_0) = 0$; and $lsb_5(C_0) = 0$.
Test 2: χ^2-test on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ in the case where both B_0 and D_0 induce zero rotation in the 1st round; $lsb_5(A_0) = 0, ..., 31$; and $lsb_5(C_0) = 0$.
Test 3: χ^2-test on $lsb_n(A_{r+1})||lsb_n(C_{r+1})$ for $n = 3, 4, 5$ in the case where both $lsb_5(A_0)$ and $lsb_5(C_0)$ are set to 0, and both B_0 and D_0 induce zero rotation in the 1st round.
Test 4: χ^2-test on (any consecutive 5 bits of A_{r+1}) $||lsb_5(C_{r+1})$ in the case where both $lsb_5(A_0)$ and $lsb_5(C_0)$ are set to 0, and both B_0 and D_0 induce zero rotation in the 1st round.

The conditions on plaintexts and ciphertexts in Test 1 are the same as those in [8]. Apparently, the conditions on plaintexts or ciphertexts in other tests are eased. We observe whether or not almost the same effect as obtained in Test 1 is obtained under the eased conditions.

3.1 Tests 1 and 2

The ciphertexts lead to much stronger biases if plaintexts (A_0, B_0, C_0, D_0) are chosen in such a way that

Table 2 The χ^2-value on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ of RC6 in Test 1(the average of 100 keys, the level and the variance).

4 rounds									
# texts	2^{12}			2^{13}			2^{14}		
The χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	1045.450	0.694	1774.828	1076.568	0.881	2177.806	1126.800	0.987	2448.999

6 rounds									
# texts	2^{28}			2^{29}			2^{30}		
The χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	1041.933	0.667	2098.079	1060.985	0.801	2263.724	1095.914	0.944	2942.704

Table 3 The χ^2-value on $lsb_n(A_{r+1})||lsb_n(C_{r+1})$ of RC6 for each # texts, the average of 100 keys, the level, and the variance.

4 rounds									
# texts	2^{12}			2^{13}			2^{14}		
χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
$n = 3$	66.275	0.635	140.251	69.518	0.733	155.518	81.111	0.938	244.195
$n = 4$	268.910	0.737	493.753	277.883	0.845	618.303	301.961	0.977	679.494
$n = 5$	1045.450	0.694	1774.828	1076.568	0.881	2177.806	1126.800	0.987	2448.973

6 rounds									
# texts	2^{29}			2^{30}			2^{31}		
χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
$n = 3$	71.804	0.791	203.645	76.572	0.883	209.564	88.474	0.981	270.062
$n = 4$	273.571	0.797	580.289	290.854	0.939	699.839	323.876	0.998	1049.104
$n = 5$	1060.985	0.801	2263.680	1095.913	0.944	2942.691	1173.418	0.999	3270.362

both $lsb_5(A_0)$ and $lsb_5(C_0)$ are 0; and both B_0 and D_0 induce zero rotation in the 1st round [8]. Test 1 reveals the effect. The implementation results for the case of $r = 4, 6$ are shown in Table 2. We compute the χ^2-value on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$, averaging over 100 keys, and the level and the variance. Especially, the variance will be discussed in the following sections. In the case of Test 1, the number of available plaintexts is 2^{108}. Next we discuss the difference between Tests 1 and 2. In the first round, each of A_1 and C_1 is added to each round key, and thus neither $lsb_5(A_1)$ nor $lsb_5(C_1)$ is zero in the final stage of the first round even if plaintexts are chosen under the conditions of Test 1. Therefore, the same effect as $lsb_5(A_0), lsb_5(C_0) = 0$ is expected if only $lsb_5(A_0)$ and $lsb_5(C_0)$ is merely fixed. Test 2 is used to examine the hypothesis. The experimental results of Test 2 are presented in Fig. 1. In Fig. 1, the horizontal line corresponds to the fixed value of $lsb_5(A_0)$ and the vertical line corresponds to the significance level of the χ^2-value for each number of plaintexts. In Fig. 1, we see that any $lsb_5(A_0)$ can be distinguished from a random sequence in almost the same way as $lsb_5(A_0) = 0$. The same discussion also holds in the case of $lsb_5(C_0)$. In summary, we need not set $lsb_5(A_0) = lsb_5(C_0) = 0$ in order to increase the χ^2-value. We can use plaintexts with any (A_0, C_0) merely by classifying them to $lsb_5(C_0)$ and $lsb_5(A_0)$, and thus the number of available plaintexts is 2^{118}.

3.2 Test 3

The bias for $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ of an output after

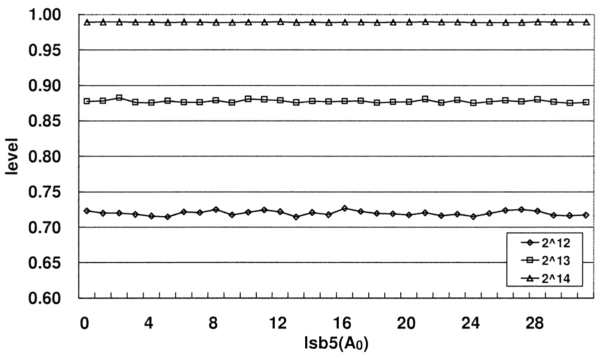

Fig. 1 The χ^2-value for each $lsb_5(A_0)$ of RC6 in Test2 (averaged over 10^4 keys).

r-rounds is confirmed to have highly nonuniform distribution [8]. In Test 3, we examine whether or not output with other bit-size leads also to highly nonuniform distribution. In our key recovery algorithm shown in Sects. 4 and 5, the size of the recovered key can be set flexibly. Therefore if the nonuniform distribution of $lsb_n(A_{r+1})||lsb_n(C_{r+1})$ for $n \neq 5$ also holds, then our algorithm can work according to the memory capacity of the machine. The experimental results of Test 3 in the case of 4 and 6 rounds are presented in Table 3. From the experimental results, we see that the larger n is, the higher the nonuniform distribution of $lsb_n(A_{r+1})||lsb_n(C_{r+1})$ is, and that the nonuniform distribution of $lsb_n(A_{r+1})||lsb_n(C_{r+1})$ for $n = 3, 4$ is also observed in the same way as $n = 5$. Since we use the χ^2-value on $lsb_3(A_{r+1})||lsb_3(C_{r+1})$ in Sect. 4, other experimental results in the case of $lsb_3(A_{r+1})||lsb_3(C_{r+1})$

Table 4 The χ^2-value on $lsb_3(A_5)||lsb_3(C_5)$ of RC6 for each # texts, the average of 10^5 keys, the level, and the variance.

# texts	2^7			2^8			2^9		
χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	63.174	0.530	126.426	63.241	0.532	126.612	63.395	0.538	126.645

# texts	2^{10}			2^{11}		
χ^2-value	Average	Level	Variance	Average	Level	Variance
	63.820	0.553	130.434	64.655	0.581	131.970

Table 5 The χ^2-value on $lsb_5(A_5)||lsb_5(C_5)$ of RC6 without pre-whitening in Test 5 (the average of 100 keys, the level, and the variance).

# texts	2^{12}			2^{13}			2^{14}		
χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	1054.720	0.761	2653.532	1083.073	0.906	2634.250	1137.702	0.993	2504.252

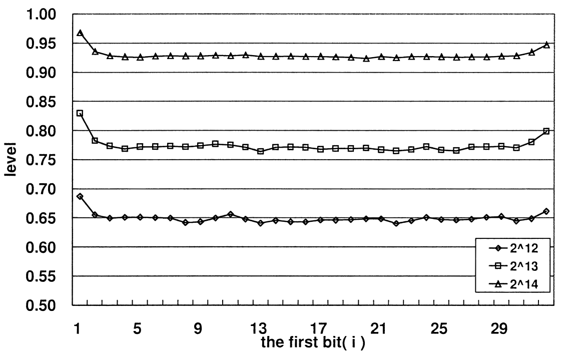

Fig. 2 Level of the χ^2-value in each consecutive 5 bits of $A_5||lsb_5(C_5)$ of RC6 for each # texts (averaged over 10^4 keys).

are shown in Table 4.

3.3 Test 4

In Test 4, we compute the χ^2-value in (any consecutive 5 bits of A_{r+1})$||lsb_5(C_{r+1})$. Figure 2 shows the experimental results in the case of 4 rounds. The horizontal line corresponds to the first bit of consecutive 5 bits of A_5, and each plot presents the level of the χ^2-value in the case of each consecutive 5 bits for each number of plaintexts. For example, the cases of $i = 1$ and $i = 32$ correspond to $A_5^{[5,1]}$ and $\{A_5^{32}, A_5^{[4,1]}\}$. In Fig. 2, we see that (any consecutive five bits of A_5)$||lsb_5(C_5)$ can be distinguished from a random sequence in almost the same way as $lsb_5(A_5)||lsb_5(C_5)$.

3.4 χ^2-Statistic of RC6 without Pre-Whitening

In this section, we focus attention on RC6 without pre-whitening, and investigate how to obtain much stronger biases with less constraint of plaintexts. In Test 2, we saw that the χ^2-value on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ becomes significantly high if both B_0 and D_0 induce zero rotation in the 1st round, and both $lsb_5(A_0)$ and $lsb_5(C_0)$ are fixed. That is, in Test 2, both $lsb_5((A_0 \oplus F(B_0 + S_0)) \lll F(D_0 + S_1))$ and $lsb_5((C_0 \oplus F(D_0 + S_1)) \lll F(B_0 + S_0))$ are fixed. Therefore, in the case of RC6 without pre-whitening, the same effect as Test 2 is expected if only both $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$ and $lsb_5((C_0 \oplus F(D_0)) \lll F(B_0))$ are fixed. To observe this, we carry out the next experiments.

Test 5: χ^2-test on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ of RC6 without pre-whitening with $lsb_5((C_0 \oplus F(D_0)) \lll F(B_0)) = 0$, and $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0)) = 0$.
Test 6: χ^2-test on $lsb_5(A_{r+1})||lsb_5(C_{r+1})$ of RC6 without pre-whitening with $lsb_5((C_0 \oplus F(D_0)) \lll F(B_0)) = 0$, and $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0)) = 0, .., 31$.

Table 5 shows the result of Test 5 in the case of 4 rounds. Compared with Table 2, we see that almost the same effect as Test 1 is obtained from Test 5. More strictly, the effect of Test 5 is better than that of Test 1. The experimental results of Test 6 are presented in Fig. 3. In Fig. 3, the horizontal line corresponds to the fixed value of $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$ and the vertical line corresponds to the χ^2-value for each number of plaintexts. From Fig. 1, we see that any $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$ can be distinguished from a random sequence in almost the same way as $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0)) = 0$. The same discussion also holds in the case of $lsb_5((C_0 \oplus F(D_0)) \lll F(B_0)) = 0$.

More importantly, in the case of the analysis of RC6 without pre-whitening, we can handle plaintexts by controlling $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$. In summary, we can use any plaintext in the analysis for RC6 without pre-whitening merely by classifying it into each of $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$ and $lsb_5((C_0 \oplus F(D_0)) \lll F(B_0))$, and thus the number of available plaintexts is 2^{128}.

3.5 Remarks on Experimental Results

We have seen, from the experimental results, that high correlation between an input and an output of RC6 is observed if both input and output are chosen appropriately. Correlation attack makes use of such correlation; if we choose a correct key, then high correlations between an input and an output of RC6 would be ob-

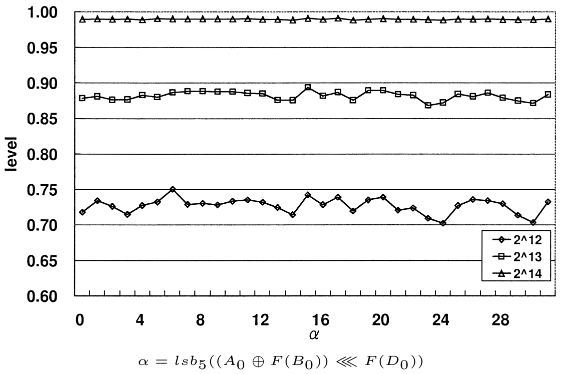

Fig. 3 The χ^2-value for each $lsb_5((A_0 \oplus F(B_0)) \lll F(D_0))$ of RC6 without pre-whitening in Test 6 (averaged over 10^4 keys).

served, but if we choose a false key, then high correlations between an input and an output of RC6 would not be observed. In the distinguishing algorithm, the χ^2-value is computed on the average of keys, and thus only the conditions under which the average χ^2-value is high are discussed. However, results of each experimental show that the variance of distribution of the χ^2-value cannot be negligible in the case of correct keys. Generally, for a normally distributed X with the average μ, and the variance σ^2, the probability that the data exists in $\{\mu - \sigma \leq X \leq \mu + \sigma\}$, $\Pr(\mu - \sigma \leq X \leq \mu + \sigma)$, satisfies

$$\Pr(\mu - \sigma \leq X \leq \mu + \sigma) = 0.68.$$

Therefore, if the variance were not reduced, then we could not rule out all false keys, and single out exactly a correct key. In the following sections, we will design key recovery algorithms in such a way that the variance of χ^2-distribution is reduced.

3.6 Estimation

In the following sections, we will show key recovery algorithms, based on the χ^2-test. We actually implement these key recovery algorithms against RC6W with 5 rounds, and evaluate the χ^2-value necessary for key recovery against RC6W with 5 rounds. For the discussion of RC6W with more rounds, we use the same method as in [8] to estimate the complexities of key recovery algorithms; we estimate *slope*, that is, how many plaintexts are required to obtain similar values in a χ^2-test on $r+1$ rounds compared with r rounds.

Our key recovery algorithms are Algorithms 2, 3, and 4. The conditions of the χ^2-test of these three key recovery algorithms are classified into two cases: one is the case of both Algorithms 2 and 3, and the other is that of Algorithm 4. We discuss the slope in each case. Note that our algorithms are applied to RC6W, but from the point of view of the χ^2-value, we can make use of the χ^2-test of RC6. The χ^2-value without post-whitening is the same as that with post-whitening. The conditions without pre-whitening are the same as those under which B_0 and D_0 induce zero rotation in the 1st round of RC6.

Table 6 $\log_2 \#$(texts) required for the χ^2-value of RC6 with each level.

	Condition 1		Condition 2	
Level	4 rounds	6 rounds	4 rounds	6 rounds
0.70	12.5	28.3	14.9	30.8
0.75	12.9	28.6	15.3	31.1
0.80	13.1	29.2	15.6	31.6
0.90	13.8	30.2	16.1	32.5
0.95	14.2	30.7	16.6	32.8

In the case of both Algorithms 2 and 3, the slope of the χ^2-test is estimated under the following conditions.

Condition 1 The χ^2-test on $lsb_3(A_{r+1})||lsb_3(C_{r+1})$ of RC6 in the case where both B_0 and D_0 introduce zero rotation in the 1st round, $lsb_5(A_0) = 0$, and $lsb_5(C_0) = 0$.

Condition 1 is the same as the conditions in the case of $n = 3$ in Test 3. The precise experimental results in Condition 1 are shown in Table 6. Table 6 shows the number of plaintexts required for the χ^2-value with levels, 0.70, 0.75, 0.80, 0.90, and 0.95, which were calculated to the first decimal place. From Table 6, we can estimate that, to obtain similar values in a χ^2-test on $r+1$ rounds, a factor of $2^{8.1}$ additional plaintexts is required compared to r rounds.

In the case of Algorithm 4, the slope of the χ^2-test is estimated under the following conditions.

Condition 2 The χ^2-test on $lsb_3(A_{r+1})||lsb_3(C_{r+1})$ of RC6 in the case where both B_0 and D_0 introduce zero rotation in the 1st round, $lsb_3(A_0) = 0$, and $lsb_3(C_0) = 0$.

The precise experimental results obtained under Condition 2 are also shown in Table 6. From Table 6, we can estimate that to obtain similar values in a χ^2-test on $r+1$ rounds, we require a factor of $2^{8.1}$ additional plaintexts compared to r rounds as in the case of Condition 1.

4. A Chosen Plaintext Correlation Algorithm

In this section, we present two chosen-plaintext key recovery algorithms against RC6W, Algorithms 2 and 3.

4.1 Algorithm 2

The conditions on plaintexts in Algorithm 2 are the same as those in [8], but Algorithm 2 is designed making use of the results of tests in Sect. 3 as follows.

1. The χ^2-statistic is not measured on a fixed part of $A_{r+1}||C_{r+1}$ (**Test 4**).
2. The degree of χ^2-statistic is flexibly set to 63 in such a way that Algorithm 2 is feasible, that is, the χ^2-statistic is computed on 6 bits of $A_{r+1}||C_{r+1}$ (**Test 3**).

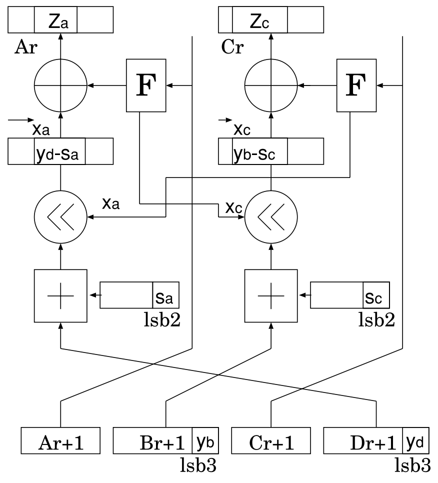

Fig. 4 Outline of Algorithm 2.

3. The χ^2-value is computed for $z_a||z_c$, to which $lsb_3(B_{r+1})||lsb_3(D_{r+1})$ is exactly decrypted by 1 round (see Fig. 4).
4. The decrypted data, $z_a||z_c$, is classified into 64 cases according to each rotation number of the r-th round, and the χ^2-value is computed for each classified case.

Algorithm 2:

```
This algorithm recovers both $lsb_2(S_{2r})$ and $lsb_2(S_{2r+1})$ of RC6W. Set $(lsb_3(B_{r+1}), lsb_3(D_{r+1})) = (y_b, y_d)$, $(lsb_2(S_{2r}), lsb_2(S_{2r+1})) = (s_a, s_c)$, and $(lsb_5(F(A_{r+1})), lsb_5(F(C_{r+1}))) = (x_c, x_a)$, where $x_a$ and $x_c$ are the rotation amounts on $A_r$ and $C_r$ in the $r$-th round, respectively.
  1. Choose a plaintext $(A_0, B_0, C_0, D_0)$ with $(lsb_5(A_0), lsb_5(C_0), lsb_5(F(B_0)), lsb_5(F(D_0))) = (0,0,0,0)$, and encrypt it.
  2. For each $s_a, s_c = 0,1,2,3$, set $s = s_a||s_c$ and $S^3_{2r}, S^3_{2r+1} = 0$, and decrypt $y_d||y_b$ with the key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round using the $r$-th round rotation amounts $x_a$ and $x_c$. The decryptions of $y_d||y_b$ are set to $z_a||z_c = z$.
  3. For each of $s, x_a, x_c$, and $z$, we update each array by incrementing $count[s][x_a][x_c][z]$.
  4. For each $s, x_a$, and $x_c$, compute $\chi^2[s][x_a][x_c]$.
  5. Compute the average $ave[s]$ of $\{\chi^2[s][x_a][x_c]\}$ for each $s$, and output $s$ with the highest $ave[s]$ as $lsb_2(S_{2r})||lsb_2(S_{2r+1})$.
```

Algorithm 2 yields the χ^2-value on z, to which y is decrypted by the final round subkey. Since the χ^2-values on decryption z using each key, $lsb_3(S_{2r})||lsb_3(S_{2r+1}) = 1||s_a||1||s_c, 1||s_a||0||s_c, 0||s_a||1||s_c, 0||s_a||0||s_c$, are coincident with each other [11], we may decrypt y by setting $S^3_{2r}, S^3_{2r+1} = 0$ temporarily. Algorithm 2 works as a 6-bit examination and 4-bit estimation, but it can work flexibly as a $2n$-bit examination and $2(n-1)$-bit estimation for $n = 3, 4$, and 5 according to the memory capacity. We can recover other bits of round keys S_{2r} and S_{2r+1} by repeating Algorithm 2 sequentially. Apparently, the number of available plaintexts is 2^{108}.

Table 7 Success probability and the χ^2-value of Algorithm 2 (in 100 trials).

#texts	#keys	χ^2-value(63 degree)		
		Average	Level	Variance
2^{17}	12	63.106	0.527	0.165
2^{18}	8	63.076	0.526	0.122
2^{19}	16	63.216	0.531	0.109
2^{20}	32	63.492	0.541	0.107
2^{21}	71	64.049	0.561	0.102
2^{22}	99	65.119	0.597	0.133
2^{23}	100	67.321	0.668	0.218

Table 8 $\log_2$(#texts) required for recovering a key with success probabilities of 90%, 70%, and 30% in Algorithm 2 (in 100 trials).

	90%	70%	30%
$\log_2$(#text)	21.4	21.0	20.0

Table 7 shows the results for RC6W with 5 rounds: the success probability among 100 trials, the average χ^2-value of recovered keys, the level, and the variance. Let us compare the results of Algorithm 2 with those in Table 4. In Algorithm 2, the χ^2-value is computed on each group, classified by the rotation number in the final round. Since all plaintexts in our experiments are randomly generated by m-sequences, plaintexts are roughly estimated to be uniformly distributed among all groups. Therefore, the χ^2-test is computed by using $1/2^{10}$ times the number of plaintexts, the results of which are shown in Table 7. The χ^2-test using $2^{20} - 2^{23}$ plaintexts in Algorithm 2 corresponds to that of $2^{10} - 2^{13}$ in the case of $n = 3$ of Test 3. In a sense, Algorithm 2 yields the χ^2-value for the sample mean, which keeps the average of χ^2-value but reduces the variance of χ^2-value from the statistical fact. Comparing Table 7 with Tables 3 and 4, we see that the variance of χ^2-value in Algorithm 2 is about $1/2^{10}$ as much as that in the corresponding Test 3, and that the average of χ^2-value in Algorithm 2 is almost the same as that in the corresponding Test 3. Algorithm 2 can recover a key with a rather low level by reducing the variance of the χ^2-value.

More precise experimental results are shown in Table 8. All results are calculated to the first decimal place. Using the data in Table 8, the number of plaintexts required for recovering a key in r rounds with the success probability of 90%, $\log_2$(#text), is estimated as

$$\log_2(\#\text{text}) = 8.1r - 19.1$$

using the slope computed in Sect. 3. By substituting $\log_2(\#\text{text}) = 108$, Algorithm 2 can break RC6W with 15 rounds with $2^{102.4}$ plaintexts with a probability of 90%. Algorithm 2 is faster than an exhaustive key search with 2^{20} memory.

Table 9 Success probability and the χ^2-value of Algorithm 3 (in 100 trials).

#texts	#keys	χ^2-value(63 degree)		
		Average	Level	Variance
2^{22}	21	63.067	0.526	0.003
2^{23}	54	63.135	0.528	0.003
2^{24}	93	63.267	0.533	0.005

4.2 Algorithm 3

We improve Algorithm 2 by making use of the results of Test 2, that is, we ease the conditions on (A_0, C_0) of plaintexts. The conditions on plaintexts in Algorithm 3 are that: both B_0 and D_0 induce zero rotation in the 1st round, and both $lsb_5(A_0)$ and $lsb_5(C_0)$ are merely fixed.

Algorithm 3:

```
This algorithm recovers both $lsb_2(S_{2r})$ and $lsb_2(S_{2r+1})$ of RC6W. Set $(lsb_3(B_{r+1}), lsb_3(D_{r+1})) = (y_b, y_d)$, $(lsb_2(S_{2r}), lsb_2(S_{2r+1})) = (s_a, s_c)$, and $(lsb_5(F(A_{r+1})), lsb_5(F(C_{r+1}))) = (x_c, x_a)$, where $x_a$ and $x_c$ are the rotation amounts on $A_r$ and $C_r$ in the $r$-th round, respectively.
  1. Choose a plaintext $(A_0, B_0, C_0, D_0)$ with $(lsb_5(F(B_0)), lsb_5(F(D_0)), lsb_5(C_0)) = (0, 0, 0)$, set $lsb_5(A_0) = t$, and encrypt it.
  2. For each $(s_a, s_c)$ $(s_a, s_c = 0, 1, 2, 3)$, set a 4-bit integer $s = s_a||s_c$, $S^3_{2r}, S^3_{2r+1} = 0$, and decrypt $y_d||y_b$ with the key $(S^3_{2r}||s_a, S^3_{2r+1}||\ s_c)$ by 1 round. We set a decryption of $y_d$,$y_b$ to $z_a$, $z_c$, which are 3-bit integers. We also set a 6-bit integer $z = z_a||z_c$.
  3. For each of $s$, $t$, $x_a$, $x_c$, and $z$, we update each array by incrementing $count[s][t][x_a][x_c][z]$.
  4. For each $s, t, x_a, x_c$, compute $\chi^2[s][t][x_a][x_c]$.
  5. Compute the average $ave[s]$ of $\{\chi^2[s][t][x_a]\ [x_c]\}$ for each $s$, and output $s$ with the highest $ave[s]$ as $lsb_2(S_{2r})||lsb_2(S_{2r+1})$.
```

The number of available plaintexts in Algorithm 3 is 2^{113}. Algorithm 3 uses plaintexts with $lsb_5(C_0) = 0$, but this condition is further eased by classifying the value of $lsb_5(C_0)$. Then the number of available plaintexts becomes 2^{118}.

Table 9 shows the results for RC6W with 5 rounds: the success probability among 100 trials, the average χ^2-value of recovered keys, the level, and the variance. Let us compare the results with those of Algorithm 2 in Table 7. In Algorithm 3, the plaintexts computed on the χ^2-value is further classified to each group by the value of $lsb_5(A_0)$. Since all plaintexts in our experiments are randomly generated by m-sequences, plaintexts are roughly estimated to be uniformly distributed among all groups. Therefore, the χ^2-test of using $2^{22} - 2^{24}$ plaintexts in Algorithm 3 corresponds to that of $2^{17} - 2^{19}$ in Algorithm 2. In the same way, the χ^2-test of using $2^{22} - 2^{24}$ plaintexts in Algorithm 3 corresponds to that of $2^7 - 2^9$ in the case of $n = 3$ of Test 3. We see the average χ^2-value by using 2^{22}, 2^{23}, or 2^{24} in Table 9 is roughly equal to those by using 2^{17}, 2^{18}, or 2^{19} in Table 7, and 2^7, 2^8, or 2^9 in Table 4, respectively. On the other hand, the variance of χ^2-value by using 2^{22}, 2^{23}, or 2^{24} in Table 9 is about $1/2^5$ that by using 2^{17}, 2^{18}, or 2^{19} in Table 7, and about $1/2^{15}$ that by using 2^7, 2^8, or 2^9 in Table 4, respectively. Algorithm 3 keeps the level of the average χ^2-value with less variance of χ^2-value. As a result, Algorithm 3 can recover a key with a lower level than Algorithm 2 by reducing the variance of χ^2-value.

Table 10 $\log_2$(#texts) required for recovering a key with success probabilities of 90%, 70%, and 30% in Algorithm 3 (in 100 trials).

	90%	70%	30%
$\log_2$(#text)	23.9	23.3	22.5

More precise experimental results are shown in Table 10. All results are calculated to the first decimal place. Using the data in Table 10, the number of plaintexts required for recovering a key in r rounds with the success probability of 90%, $\log_2$(#text) is estimated as

$$\log_2(\#\text{text}) = 8.1r - 16.6$$

using the bias computed in Sect. 3. By substituting $\log_2(\#\text{text}) = 118$, Algorithm 2 can break RC6W with 16 rounds with $2^{113.0}$ plaintexts with a probability of 90%. Algorithm 3 is faster than an exhaustive key search with 2^{25} memory.

5. A Known Plaintext Correlation Algorithm

In this section, we present another key recovery algorithm, Algorithm 4, which applies Algorithm 3 in such a way that all plaintexts are available. We have seen that it is very effective for key recovery to compute the χ^2-value for each appropriate group instead of computing the χ^2-value directly for any plaintexts. We apply the idea to the results of Tests 5 and 6 presented in Sect. 3. Algorithm 4 classifies any plaintext (A_0, B_0, C_0, D_0) into the same $lsb_3((A_0 \oplus F(B_0)) \lll F(D_0))$ and $lsb_3((C_0 \oplus F(D_0)) \lll F(B_0))$.

Algorithm 4:

```
This algorithm recovers both $lsb_2(S_{2r})$ and $lsb_2(S_{2r+1})$ of RC6W. Set $(lsb_3(B_{r+1}), lsb_3(D_{r+1})) = (y_b, y_d)$, $(lsb_2(S_{2r}), lsb_2(S_{2r+1})) = (s_a, s_c)$, and $(lsb_5(F(A_{r+1})), lsb_5(F(C_{r+1})) = (x_c, x_a)$, where $x_a$ and $x_c$ are the rotation amounts on $A_r$ and $C_r$ in the $r$-th round, respectively.
  1. Given a plaintext $(A_0, B_0, C_0, D_0)$, set $lsb_3((A_0 \oplus F(B_0)) \lll F(D_0)) = t_a$, $lsb_3((C_0 \oplus F(D_0)) \lll F(B_0)) = t_c$, and encrypt it.
  2. For each $(s_a, s_c)$ $(s_a, s_c = 0, 1, 2, 3)$, set a
```

```
   4-bit integer $s = s_a||s_c$, $S^3_{2r}, S^3_{2r+1} = 0$, and
   decrypt $y_d||y_b$ with the key $(S^3_{2r}||s_a, S^3_{2r+1}|| s_c)$
   by 1 round. We set a decryption of $y_d, y_b$
   to $z_a, z_c$, which are 3-bit integers. We also
   set a 6-bit integer $z = z_a||z_c$.
3. For each of $s$, $t_a$, $t_c$, $x_a$, $x_c$, and $z$, we update
   each array by incrementing $count[s][t_a][t_c][x_a][x_c][z]$.
4. For each $s$, $t_a$, $t_c$, $x_a$, $x_c$, compute $\chi^2[s][t_a][t_c][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t_a][t_c][x_a][x_c]\}$ for each $s$, and output $s$ with the
   highest $ave[s]$ as $lsb_2(S_{2r})||lsb_2(S_{2r+1})$.
```

The number of available plaintexts in Algorithm 4 is 2^{128}. Algorithm 4 classifies plaintexts according to each 3-bit $(A_0 \oplus F(B_0)) \lll F(D_0)$ and $(C_0 \oplus F(D_0)) \lll F(B_0)$, which may be enlarged to, for example, 5, like the conditions of Tests 5 and 6. However, the larger the classified bit size is, the greater the memory is required.

Table 11 show the results for RC6W with 5 rounds: the success probability among 100 trials, the average χ^2-value of recovered keys, the level, and the variance. We see that, in Algorithm 4, the variance of χ^2-value is much more reduced than Algorithm 2 or 3. As a result, Algorithm 4 can recover a key more efficiently than Algorithms 2 and 3 by reducing the variance of χ^2-value.

More precise experimental results are shown in Table 12. All results are calculated to the first decimal place. Using the data in Table 12, the number of plaintexts required for recovering a key in r rounds with the success probability of 90%, $\log_2(\#\text{text})$, is estimated as

$$\log_2(\#\text{text}) = 8.1r - 13.8$$

using the slope computed in Sect. 3. By substituting $\log_2(\#\text{text}) = 128$, Algorithm 4 can break RC6W with 17 rounds using $2^{123.9}$ plaintexts with a probability of 90%.

Let us discuss the amount of work. For each plaintext in Algorithm 4, we encrypt a plaintext, and decrypt a ciphertext by 1 round with each candidate s. Here we set one unit of work as 1 encryption. In the case of RC6W with 5 rounds, we obtain a complexity of

$$2^{26.7}(1 + 2^4/5) = 2^{26.7} * 4.2 \leq 2^{26.7} * 2^{2.1} = 2^{28.8}.$$

In the case of RC6W with 17 rounds, we obtain a complexity of

$$2^{123.9}(1 + 2^4/17) \leq 2^{123.9} * (1 + 0.95) \leq 2^{124.9}.$$

In summary, Algorithm 4 can break, with the probability of 90%, RC6W with 5 rounds or 17 rounds using $2^{26.7}$ or $2^{123.9}$ plaintexts, and $2^{28.8}$ or $2^{124.9}$ work, respectively. Note that each case is within the same memory capacity of 2^{26}.

Table 11 Success probability and the χ^2-value of Algorithm 4 (in 100 trials).

#texts	#keys	χ^2-value(63 degrees)		
		Average	Level	Variance
2^{25}	26	63.057	0.526	0.0003
2^{26}	59	63.108	0.528	0.0005
2^{27}	100	63.230	0.532	0.0007

Table 12 $\log_2(\#\text{texts})$ required for recovering a key with success probabilities of 90%, 70%, and 30% in Algorithm 4 (in 100 trials).

	90%	70%	30%
$\log_2(\#\text{text})$	26.7	26.3	25.3

6. A Key Recovery Algorithm against RC6-64

In [13], a two-register version for RC6, which is oriented to 64-bit architecture, was also described. Here we call the two-register version for RC6 simply RC6-64. In this section, we apply the idea of Algorithm 4 to RC6-64, and discuss the difference between RC6 and RC6-64 from the point of view of the security.

6.1 A Two-Register Version for RC6

Here we present RC6-64. The round function of RC6-64 has almost the same structure as that of RC6, but it consists of two units (A_i, B_i). An input of the i-th round is denoted by (A_i, B_i), and (A_0, B_0) is a plaintext, where A_i and B_i are each 64 bits. The i-th subkey S_i is also 64 bits. Here the function F is modified to F_6 in a 64-bit-oriented manner;

$$F_6(X) = X(2X + 1) \pmod{2^{64}} \lll 6.$$

Algorithm 5 (Encryption with RC6-64):

```
1. $A_1 = A_0$; $B_1 = B_0 + S_0$;
2. for $i = 1$ to $r$ do:
     $t = F_6(B_i)$; $A_i = ((A_i \oplus t) \lll t) + S_i$;
     $A_{i+1} = B_i$; $B_{i+1} = A_i$;
3. $A_{r+2} = A_{r+1} + S_{r+1}$; $B_{r+2} = B_{r+1}$.
```

Part 1 and 3 of Algorithm 5 are called pre-whitening and post-whitening, respectively. We call the version of RC6-64 with neither pre-whitening nor post-whitening, simply RC6-64W. As we see in Algorithm 5, RC6-64 is designed based as the same concept as RC6. Therefore, we should expect that the security of the round-function is estimated to be almost the same as that of RC6. However, the security of RC6-64 is lower than that of RC6, as shown in the next section.

6.2 Key Recovery Algorithm to RC6-64W

We apply Algorithm 4 in Sect. 5 to RC6-64W.

Algorithm 6 (Algorithm to RC6-64W):

Table 13 Success probability and the χ^2-value of Algorithm 6 to RC6-64W with 5 and 7 rounds (in 100 trials).

5 rounds					7 rounds				
#texts	#keys	χ^2-value(63 degrees)			#texts	#keys	χ^2-value(63 degrees)		
		Average	Level	Variance			Average	Level	Variance
2^{15}	20	31.214	0.545	0.0296	2^{25}	30	31.278	0.548	0.0394
2^{16}	65	31.504	0.559	0.0290	2^{26}	53	31.512	0.559	0.0302
2^{17}	96	32.022	0.584	0.0335	2^{27}	95	32.050	0.586	0.0286

Table 14 $\log_2$(#texts) required for recovering a key with success probabilities of 90%, 70%, and 30% in Algorithm 6 to RC6-64W with 5 and 7 rounds (in 100 trials).

	5 rounds			7 rounds		
	90%	70%	30%	90%	70%	30%
$\log_2$(#text)	16.8	16.2	15.3	26.9	26.2	25.0

Table 15 The χ^2-value on $lsb_5(A_{r+1})$ in Test 7 (the average of 100 keys, the level and the variance).

4 rounds									
# texts	$2^{6.9}$			$2^{8.7}$			$2^{9.5}$		
The χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	34.600	0.700	86.071	40.893	0.890	126.840	51.261	0.988	188.444

6 rounds									
# texts	$2^{16.5}$			$2^{17.5}$			$2^{18.9}$		
The χ^2-value	Average	Level	Variance	Average	Level	Variance	Average	Level	Variance
	33.966	0.674	73.204	37.666	0.809	112.739	45.193	0.952	131.131

```
This algorithm recovers $lsb_4(S_r)$ of RC6-64W. Set $lsb_5(B_{r+1}) = y$, $lsb_4(S_r) = s$, and $lsb_6(F_6(A_{r+1})) = x$, where $x$ is the rotation amount on $A_r$ in the $r$-th round.
1. Given a plaintext $(A_0, B_0)$, set $lsb_5((A_0 \oplus F_6(B_0)) \lll F_6(B_0)) = t$, and encrypt it.
2. For each $s$ ($s = 0, \cdots, 15$), set $S_r^5 = 0$, and decrypt $y$ with the key $S_r^5||s$ by 1 round. We also set a decryption of $y$ to $z$, which is a 5-bit integer.
3. For each of $s$, $t$, $x$, and $z$, we update each array by incrementing $count[s][t][x][z]$.
4. For each $s$, $t$, and $x$, compute $\chi^2[s][t][x]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t][x]\}$ for each $s$, and output $s$ with the highest $ave[s]$ as $lsb_4(S_r)$.
```

The number of available plaintexts in Algorithm 6 is 2^{128}. Table 13 show the results for RC6-64W with 5 and 7 rounds: the success probability among 100 trials, the average χ^2-value of recovered keys, the level, and the variance. More precise experimental results are shown in Table 14. All results are calculated to the first decimal place. Using the data in Table 14, the number of plaintexts required for recovering a key in r rounds with the success probability of 90%, $\log_2$(#text), is estimated as

$$\log_2(\#\text{text}) = 5.0r - 8.2.$$

By substituting $\log_2(\#\text{text}) = 128$, Algorithm 6 can break RC6-64W with 27 rounds with $2^{126.8}$ plaintexts with a probability of 90%.

Here we estimate the amount of work in the same way as for Algorithm 4. In the case of RC6W-64 with 5, 7, or 27 rounds, we obtain a complexity of $2^{18.9}$, $2^{28.7}$, or $2^{127.5}$, respectively. In summary, Algorithm 6 can break, with the probability of 90%, RC6W-64 with 27 rounds by using $2^{126.8}$ plaintexts, $2^{127.5}$ work, and 2^{20} memory.

6.3 Further Discussion

We discuss the difference between the round function of RC6 and that of RC6-64 from the point of view of the security. First we conduct the following Test 7 of RC6-64, the results of which are shown in Table 15.

Test 7: χ^2-test on $lsb_5(A_{r+1})$ in RC6-64 with r rounds in the case where B_0 induces zero rotation in the 1st round, and $lsb_5(A_0) = 0$

Let us compare each round function between RC6-64 and RC6 using the data in Tables 15 and 2. The size of subkeys in RC6-64 is 64 bits. Hence, the security level of one round in RC6-64, the size of subkeys of one round, is estimated to be equal to that in RC6-32, which has two 32-bit subkeys in one round. Furthermore, the round function of RC6-64 has almost the same structure as that of RC6. However, the slope, defined in Sect. 3.6, of RC6-64 is apparently lower than that of RC6. This means that the correlations between an input of the round function and the output in RC6-64 is retained more than in RC6. The round function of RC6-64 mixes up data less than that of RC6. We often discuss that

Table 16 Our results with the success probability of 90%.

Cipher	#rounds	#works	#texts	#memory
RC6W	5	$2^{28.8}$	$2^{26.7}$	2^{26}
RC6W	17	$2^{124.9}$	$2^{123.9}$	2^{26}
RC6W-64	5	$2^{18.9}$	$2^{16.8}$	2^{20}
RC6W-64	7	$2^{28.7}$	$2^{26.9}$	2^{20}
RC6W-64	27	$2^{127.5}$	$2^{126.8}$	2^{20}

the weak point of RC5 is in a data-dependent rotation, which is defined by only 5 bits of subkey and data. Although this weakness of data-dependent rotation is improved in both RC6 and RC6-64, RC6-64 is much weaker than RC6. The difference between RC6-64 and RC6 is the data structure: RC6-64 consists of 2 units, and RC6 consists of 4 units. Both RC6-64 and RC6 make use of modular-additions in order to mix data within the unit. Correlations are induced by the consecutiveness of modular-additions. Our results indicate that the structure of RC6, 4-unit plaintexts, reduce correlations more efficiently than that of RC6-64, 2-unit plaintexts.

7. Conclusions

We have proposed an efficient and feasible known plaintext correlation attack on RC6W and RC6W-64. Our attack can break RC6W/r(RC6W-64/r) with a success probability of 90% using $2^{8.1r-13.8}(2^{5.0r-8.2})$ plaintexts. Therefore, our attack can break RC6W (RC6W-64) with 17 (27) rounds by using $2^{123.9}(2^{126.8})$ plaintexts. Table 16 summarizes our results. We have also found that the security of the round function of RC6 is enhanced by not only the data-dependent rotation being dependent on all bits of the input unit, but also by the consecutiveness of modular additions being broken by dividing data into 4 units.

Acknowledgments

The authors express our gratitude to Soichi Furuya and Hirotaka Yoshida for their kind support. The authors wish to thank the anonymous referees for invaluable comments.

References

[1] A. Biryukov and E. Kushilevitz, "Improved cryptanalysis of RC5," Advances in Cryptology-Proceedings of EUROCRYPT'98, Lecture Notes in Computer Science, vol.1403, pp.85–99, Springer-Verlag, 1998.

[2] J. Borst, B. Preneel, and J. Vandewalle, "Linear cryptanalysis of RC5 and RC6," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.1636, pp.16–30, Springer-Verlag, 1999.

[3] S. Contini, R. Rivest, M. Robshaw, and Y. Yin, "Improved analysis of some simplified variants of RC6," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.1636, pp.1–15, Springer-Verlag, 1999.

[4] J. Hayakawa, T. Shimoyama, and K. Takeuchi, "Correlation attack to the block cipher RC5 and the simplified variants of RC6," submitted paper in Third AES Candidate Conference, April 2000.

[5] B. Kaliski and Y. Lin, "On differential and linear cryptanalysis of the RC5 encryption algorithm," Advances in Cryptology-Proceedings of CRYPTO'95, Lecture Notes in Computer Science, vol.963, pp.171–184, Springer-Verlag, 1995.

[6] J. Kelsey, B. Schneier, and D. Wagner, "Mod n cryptanalysis, with applications against RC5P and M6," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.1636, pp.139–155, Springer-Verlag, 1999.

[7] L. Knudsen and W. Meier, "Improved differential attacks on RC5," Advances in Cryptology-Proceedings of CRYPTO'96, Lecture Notes in Computer Science, vol.1109, pp.216–228, Springer-Verlag, 1996.

[8] L. Knudsen and W. Meier, "Correlations in RC6 with a reduced number of rounds," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.1978, pp.94–108, Springer-Verlag, 2001.

[9] D. Knuth, The art of computer programming, vol.2, Seminumerical Algorithms, 2nd ed., Addison Wesley, Reading, Mass. 1981.

[10] A. Menezes, P.C. Oorschot, and S. Vanstone, Handbook of applied cryptography, CRC Press, 1996.

[11] A. Miyaji, M. Nonaka, and Y. Takii, "Improved Correlation Attack on RC5," IEICE Trans. Fundamentals, vol.E85-A, no.1, pp.44–57, Jan. 2002.

[12] http://cryptonessie.org

[13] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, The RC6 Block Cipher, vo1.1, v.1.1, 1998.
Available at http://www.rsasecurity.com/rsalabs/rc6

[14] R. Rivest, "The RC5 encryption algorithm," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.1008, pp.86–96, Springer-Verlag, 1995.

[15] S. Shirohata, An Introduction of Statistical Analysis, Kyouritu Syuppan, 1992.

[16] T. Shimoyama, M. Takenaka, and T. Koshiba, "Multiple linear cryptanalysis of a reduced round RC6," Proc. Fast Software Encryption, Lecture Notes in Computer Science, vol.2365, pp.76–88, Springer-Verlag, 2002.

Atsuko Miyaji received her B.Sc., M.Sc., and Dr. Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997, respectively. She was with Matsushita Electric Industrial Co., LTD from 1990 to 1998, where she engaged in research and development of secure communications. She has been an associate professor at JAIST (Japan Advanced Institute of Science and Technology) since 1998, and with the computer science department of University of California, Davis since 2002. Her research interests include the application of projective varieties theory into cryptography and information security. She received the IPSJ Sakai Special Researcher Award in 2002. She is a member of the International Association for Cryptologic Research and the Information Processing Society of Japan.

Masao Nonaka received his B.Sc. degree in computer science and engineering from University of Aizu, and M. Info. Sci. degree from Japan Advanced Institute of Science and Technology in 2000 and 2002, respectively. He joined Matsushita Electric Industrial Co., LTD. in 2002 and is engaged in research and development in the field of information security systems. He is a member of the Information Processing Society of Japan.