Improved Correlation Attack on RC5

Atsuko Miyaji
Osaka University
306 PUBLICATIONS **3,876** CITATIONS

PAPER *Special Section on Cryptography and Information Security*

Improved Correlation Attack on RC5

Atsuko MIYAJI[†], ***Member***, **Masao NONAKA**[†], ***and*** **Yoshinori TAKII**[†], ***Nonmembers***

SUMMARY Various attacks against RC5 have been analyzed intensively([1], [2], [4]–[7]). A known plaintext attack([2]) has not been reported that it works on so higher round as a chosen plaintext attack([1]), but it can work more efficiently and practically. In this paper we investigate a known plaintext attack against RC5 by improving a correlation attack ([7]). As for a known plaintext attack against RC5, the best known result is a linear cryptanalysis([2]). They have reported that RC5-32 with 10 rounds can be broken by 2^{64} plaintexts under the heuristic assumption: RC5-32 with r rounds can be broken with a success probability of 90% by using 2^{6r+4} plaintexts. However their assumption seems to be highly optimistic. Our known plaintext correlation attack can break RC5-32 with 10 rounds (20 half-rounds) in a more strict sense with a success probability of 90% by using $2^{63.67}$ plaintexts. Furthermore our attack can break RC5-32 with 21 half-rounds in a success probability of 30% by using $2^{63.07}$ plaintexts.

key words: *RC5, a known plaintext attack, a correlation attack*

1. Introduction

RC5 is a block cipher designed by Rivest([11]), which is constructed by only simple arithmetic such as an addition, a bit-wise exclusive-or(XOR), and a data dependent rotation. Therefore RC5 can be implemented efficiently with small memory. RC5-32/r means that two 32-bit-block plaintexts are encrypted by r rounds, where one round consists of two half-rounds. RC6 is the next version of RC5, which has almost the same construction as RC5: RC6 consists of a multiplication, an addition, XOR, and a data dependent rotation. While the input of RC5 consists of 2 words such as (L_0, R_0), that of RC6 consists of 4 words. This is why approach of attacks on RC5 is similar to that on RC6, but a slight difference is needed.

Various attacks against RC5 have been analyzed intensively([1], [2], [4]–[7]). As for a chosen plaintext attack, the best known algorithm([1]) can break RC5-32/12 by using $(2^{44}, 2^{54.5})$ pairs of chosen plaintexts and known plaintexts. However it requires further much stored plaintexts such as $2^{54.5}$. Even in the case of RC5-32/10, it requires $(2^{36}, 2^{50.5})$ pairs of chosen plaintexts and known plaintexts with stored $2^{50.5}$ plaintexts. In a realistic sense, it would be rather infeasible to employ such an algorithm. On the other hand, a known plaintext attack can work more efficiently and practically although it has not been reported that it works on so higher round like 12. In this paper we investigate a known plaintext attack against RC5.

As for a known plaintext attack against RC5, the best known result is a linear cryptanalysis([2]). They have reported that RC5-32 with 10 rounds can be broken by 2^{64} plaintexts under the heuristic assumption: RC5-32 with r rounds can be broken with a success probability of 90% by using 2^{6r+4} plaintexts. However their assumption seems to be highly optimistic. Table 1 shows both their results and our results. In fact, their experimental results report that RC5-32 with 3 or 4 rounds is broken with a success probability of only 81% or 82% if we use 2^{22} or 2^{28} plaintexts respectively. This means that their estimation does not hold even in such lower rounds as 3 or 4. On the other hand, they also discussed the theoretical complexity of breaking RC5-32 with r rounds: RC5-32 with r rounds can be broken with a success probability of 90% by using $2^{6.8r+2.4}$ plaintexts. According as their theoretical assumption, it requires $2^{22.8}$ or $2^{29.6}$ plaintexts in order to break RC5-32/3 or RC5-32/4 with a success probability of 90%. Actually it seems that the theoretical estimate reflects their experimental results. Note that under the theoretical assumption, their known plaintext attack can break RC5-32/9 but not RC5-32/10 with a success probability of 90%. In this paper, we improve a correlation attack([7]) as a known plaintext attack. Our attack can break RC5-32/10 with a success probability of 90% by using $2^{63.67}$ plaintexts. As we see in Table 1, our attack can break solidly RC5-32/2, RC5-32/3, RC5-32/4, and also RC5-32/5 with less plaintexts and a higher probability compared with [2]. In a more strict sense our attack can break RC5-32 with 10 rounds by a known plaintext attack.

Correlation attack makes use of correlations between an input and the output, which is measured by the χ^2 test([6], [7]): the specific rotation in both RC5 and RC6 is considered to cause the correlations between the corresponding two 5-bit integer values. In [7], correlation attacks against RC6-32 recover subkeys from the 1st round to the r-th round by handling a plaintext in such a way that the χ^2-test after one round becomes significantly higher value. Their main idea is to choose such a plaintext that the least significant five bits in the first and third words are constant after one-round encryption. Therefore plaintexts in RC6 are chosen as follows: 1. the least significant five bits in the first

Manuscript received March 23, 2001.

[†]The author is with Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.

Table 1 Required plaintexts for attack on RC5

	$r-round$ estimation	2 rounds		3 rounds		4 rounds		5 rounds	
	theoretical(heuristic)	#texts	#keys	#texts	#keys	#texts	#keys	#texts	#keys
[2]	$2^{6.8r+2.4}(2^{6r+4})$	2^{16}	92/100	2^{22}	81/100	2^{28}	82/100	2^{34}	9/10
our attack	$2^{6.14r+2.27}$	2^{15}	100/100	2^{22}	100/100	2^{28}	99/100	2^{33}	90/100
		2^{14}	95/100	2^{21}	95/100	2^{27}	96/100	2^{32}	60/100

and third words are zero; 2. the fourth word is set to the values that introduce a zero rotation in the 1st round. To sum up, their attack controls a plaintext in two parts with 5 bits: 5 bits corresponding to the χ^2-test and 5 bits in relation to data dependent rotations. Let us apply their attack to RC5, where a plaintext is represented by 2 words (L_0, R_0). According to their approach, it is necessary to control a plaintext in each block with each 5 bits: Choose a plaintext such that the least significant five bits of L_0 is 0 and that R_0 introduces a zero rotation in the 1st round. As a result available plaintexts are reduced by 2^{10}. Compared with RC6, available plaintexts to attack RC5 is extremely less since the block size of RC5 is just half of RC6. Therefore it is critical to reduce available plaintexts of RC5 by 2^{10} in order to break RC5 with more higher round. This is why their attack does not work well on RC5 directly. In fact they also reported that their attacks do not work well on RC5 compared with the existing attack([1]). In [3], a correlation attack is also applied to RC5. Their algorithm searches subkeys from the final round to the 1st round by fixing both $lsb_5(R_0)$ and $lsb_5(L_0)$ to be 0. Therefore their attack also suffers from the same problem of less available plaintext. The important factor to target at RC5 is how to increase the available plaintexts.

In this paper, we investigate how output L_{h+1} after h half-rounds depends on a chosen plaintext, and find experimentally the following features of RC5.

1. The χ^2-values for the least significant five bits on L_{h+1} become significantly high by simply setting such R_0 that fixes a rotation amount in the 1st half-round. Note that any rotation amount, which is not necessarily small rotation amount, outputs the higher χ^2-values.

2. Any consecutive five bits on L_{h+1} outputs similarly high χ^2-values by simply setting such R_0 that fixes a rotation amount in the 1st half-round.

Usually we know that output of RC5 is highly unlike to be uniformly distributed if a plaintext is chosen in such a way that it introduces a zero rotation or small amount rotation in the 1st half-round([7]). However from the above feature 1, output of RC5 is also highly unlike to be uniformly distributed if only a rotation in the 1st half-round is fixed. Apparently a rotation in the 1st half-round is fixed if and only if the least significant five bits of R_0 is fixed. This means that any plaintext can be used for correlation attack by classifying it in the same least significant five bits. In this way we can extend a chosen plaintext correlation attack to a known plaintext attack without any cost. From the above feature 2, any consecutive five bits on L_{h+1} can be used to compute the χ^2-values in the similar success probability.

We improve a correlation attack as a known plaintext attack by taking full advantage of the above features. The main points of our attack on RC5 are as follows:

1. Use any plaintext by classifying it into the same least significant five bits.

2. Determine the parts, on which the χ^2-statistic is measured, according to ciphertexts.

We also present two algorithms to recover 31 bits of the final half-round key: one recovers each 4 bits in serial and the other recovers each 4 bits in parallel. By employing our correlation attack, RC5-32 with r rounds(h half-rounds) can be broken with a success probability of 90% by using $2^{6.14r+2.27}(2^{3.07h+2.27})$ plaintexts. Therefore in a more strict sense our attack can break RC5-32/10 with $2^{63.67}$ plaintexts in a probability of more than 90%. In the case of success probability 30%, our attack can break RC5-32 with r rounds(h half-rounds) by using $2^{5.90r+1.12}(2^{2.95h+1.12})$ plaintexts. Therefore our attack can break RC5-32 with 21 half-rounds by using $2^{63.07}$ plaintexts in a probability of 30%.

This paper is organized as follows. Section 1 summarizes some notations and definitions, which are used in this paper. Section 3 applies Knudsen-Meier's correlation attack on RC5, and discusses the differences between RC5 and RC6. Section 4 describes some experimental results including the above features of RC5. Section 5 presents our key recovery algorithms, main algorithm, a serial key recovery algorithm, and a parallel key recovery algorithm. Section 6 discusses how to extend our main algorithm to a known plaintext algorithm.

2. Preliminary

This section denotes some notations and definitions, which are used in the following sections.

2.1 RC5

Here we describe RC5 algorithm after defining the following notations.

$+, \boxplus$:	an addition mod 2^{32};
$-, \boxminus$:	a subtraction mod 2^{32};
$\oplus$:	a bit-wise exclusive OR;
r :	the number of (full)rounds;
h :	the number of half-rounds ($h = 2r$);
$a \lll b$:	a cyclic rotation of a to the left by b bits;
$a \ggg b$:	a cyclic rotation of a to the right by b bits;
(L_i, R_i):	an input of the i-th half-round, (L_0, R_0), (L_{h+1}, R_{h+1}) is a plaintext, a ciphertext after h half-rounds encryption, respectively;
S_i :	the i-th subkey(S_{h+1} is a subkey of the h-th half-round);
$lsb_n(X)$:	the least significant n bits of X;
X^i :	denotes the i-th bit of X;
$X^{[i,j]}$:	denotes from the i-th bit to the j-th bit of X $(i > j)$;
$\overline{X}$:	a bit-wise inversion of X.

We set the least significant bit(LSB) as the 1st bit, and the most significant bit(MSB) as the 32-th bit for any 32-bit element. RC5 encryption is defined as follows. Figure 1 shows 1 half-round of encryption and decryption: a plaintext (L_0, R_0) is encrypted to (L_{h+1}, R_{h+1}) by h half-rounds iterations of a main loop, which is called one half-round. Note that two consecutive half-rounds correspond to one round of RC5.

Algorithm 1 (RC5 encryption):
```
1.  $L_1 = L_0 + S_0$;  $R_1 = R_0 + S_1$;
2.  for $i = 1$ to $h$ do:
    $L_{i+1} = R_i$;  $R_{i+1} = ((L_i \oplus R_i) \lll R_i) + S_{i+1})$.
```

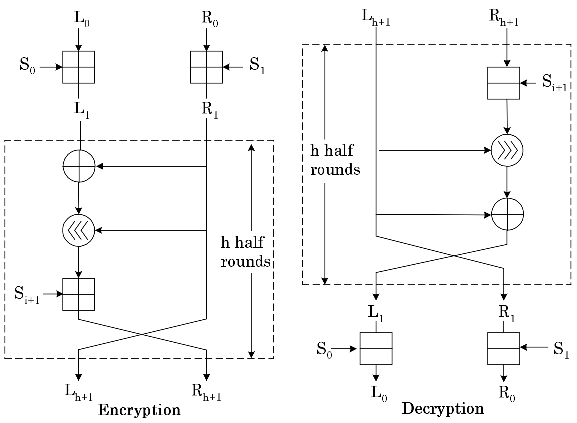

Fig. 1 Encryption and decryption with RC5-$w/r/b$

2.2 χ^2-Test

We make use of the χ^2-tests for distinguishing a random sequence from nonrandom sequence([5], [7], [8]). Let $X = X_0, ..., X_{n-1}$ be sets of $\{a_0, ..., a_{m-1}\}$. Let $N_{a_j}(X)$ be the number of X which takes on the value a_j. The χ^2-statistic of X which estimates the difference between X and the uniform distribution is defined as follows:

$$\chi^2(X) = \frac{m}{n} \sum_{i=0}^{m-1} \left(N_{a_i}(X) - \frac{n}{m} \right)^2 .$$

In our investigation of RC5, we use the threshold for 31 degrees of freedom, which is shown in Table 2. For example, (level, χ^2)=(0.95, 44.99) in Table 2 means that the value of χ^2-statistic exceeds 44.99 in the probability of only 5% if the observation X is uniform. Here, we set the level to 0.95 in order to distinguish the observation X from a random permutation.

2.3 Experimental remark

In our experiments, all plaintexts are generated by using m-sequence([9]). For example, Main algorithm uses 59-bit random number generated by m-sequence, and Extended algorithm in Section 6 uses 64-bit random number generated by m-sequence. The platforms in our experiments are IBM RS/6000 SP (PowerPC 604e/332MHz $\times$ 256) with memory of 32 GB.

3. Applying Knudsen-Meier's correlation attack to RC5

In [7], Knudsen and Meier proposed a key-recovery attack to RC6, which estimates a subkey from the 1st round to the r-th round by handling a plaintext. Their main idea is to choose such a plaintext that the least significant five bits in the first and third words are constant after one-round encryption. Therefore plaintexts in RC6 are chosen as follows: 1. the least significant five bits in the first and third words are zero; 2. the fourth word is set to the values that introduce a zero rotation in the 1st round. To sum up, their attack controls a plaintext in two parts with 5 bits: 5 bits corresponding to the χ^2-test and 5 bits in relation to data dependent rotations. Let us apply their idea to RC5 directly.

Algorithm 2 (Knudsen-Meier's attack to RC5):
```
This algorithm recovers $lsb_5(S_1)$.
Set $s = lsb_5(S_1)$, and $lsb_5(R_0) = x$.
1.  For each $s (s = 0, 1, \cdots, 31)$ of $lsb_5(S_1)$,
    compute $x$ in such a way that it leads to
    a zero rotation in the 1-st half-round.
    In other words, set $x + s = 0 \pmod{32}$.
2.  Choose plaintexts $(L_0, R_0)$ with
    $(lsb_5(L_0), lsb_5(R_0)) = (0, x)$, and set
    $y = lsb_5(L_{2h+1})$
3.  For each plaintext $(L_0, R_0)$, update each
    array by incrementing $count[s][y]$.
4.  For each $s$, compute the $\chi^2$-value $\chi^2[s]$,
    set $lsb_5(S_1) = s$ for the highest value $\chi^2[s]$,
    and output $lsb_5(S_1)$.
```

In order to fix $lsb_5(R_2)$ after the 1st half-round, Algorithm 2 fixes $lsb_5(L_0)$ to be 0 and $lsb_5(R_0)$ to be the value that leads zero rotation amount in the 1st

Table 2 χ^2-distribution with 31 degree of freedom

Level	0.50	0.60	0.70	0.80	0.90	0.95	0.99	0.999	0.9999
χ^2	30.34	32.35	34.60	37.36	41.42	44.99	52.19	61.10	69.11

half-round. As a result, Algorithm 2 can use only 2^{54} plaintexts. In general, the number of plaintexts on RC5 is not so large as RC6. Therefore it is not efficient to apply their attack to RC5. An attack should be improved in such a way that more plaintexts are available. In [3], a correlation attack is also applied to RC5. Their algorithm searches subkeys from the final subkey by fixing both $lsb_5(R_0)$ and $lsb_5(L_0)$ to be 0. Therefore their algorithm also suffers from the same problem of less available plaintext. Note that available plaintexts are 2^{54} in both case.

4. χ^2-statistic of RC5

In this section, we investigate how to reduce the constraint of plaintexts in order to increase available plaintexts. In RC5, $lsb_5(R_0)$ determines the 1st half-round data dependent rotation, so it would be desirable to handle $lsb_5(R_0)$ in some way. On the other hand, the effect of $lsb_5(L_0) = 0$ deeply depends on $lsb_5(R_0)$ as follows: 1. if $lsb_5(R_0)$ is fixed to a value that leads zero rotation amount in the 1st half-round in the same way as [7], then $lsb_5(L_0) = 0$ can fix $lsb_5(R_2)$, that is, fix the rotation amount of the 2nd half-round, and can also fix $lsb_5(L_3)$ for any available plaintext; 2. if $lsb_5(R_0)$ is fixed to just 0 in the same way as [3], then $lsb_5(L_0) = 0$ can not fix $lsb_5(R_2)$ (i.e. $lsb_5(L_3)$) for any available plaintext. Therefore we want to investigate the effect of $lsb_5(L_0) = 0$ in the case of $lsb_5(R_0) = 0$. Furthermore we want to investigate which parts are suitable for being measured by χ^2-statistic. In other words, we investigate which parts output the higher χ^2-statistics. To observe these situations, we conduct the following five experiments in each h half-round.

Test 1: χ^2-test on $lsb_5(L_{h+1})$ in setting $lsb_5(R_0) = lsb_5(L_0) = 0$.
Test 2: χ^2-test on $lsb_5(L_{h+1})$ in setting $lsb_5(R_0) = 0$.
Test 3: χ^2-test on $lsb_5(L_{h+1})$ in setting $lsb_5(L_0) = 0$.
Test 4: χ^2-test on $lsb_5(L_{h+1})$ in setting $lsb_5(R_0) = x (x = 0, 1, ..., 31)$. (There are 32 cases in Test 4.)
Test 5: χ^2-test on any consecutive 5 bits of L_{h+1} in setting $lsb_5(R_0) = 0$. (There are 32 cases in Test 5.)

4.1 Test 1, 2, and 3

Here we investigate the experimental results of Test 1, 2, and 3. Before showing experimental results, we discuss the differences between Test 1 and Test 2. The condition of $lsb_5(R_0) = 0$ means that the rotation amount of the 1st half-round is only fixed. The purpose of two tests is to observe the effect of handling a plaintext in the part corresponding to the χ^2-test under the condition that the rotation amount of the 1st half-round is fixed. Apparently Test 1 handles a plaintext in the part corresponding to the χ^2-test, but Test 2 does not handle it. On the other hand, Test 3 sets only $lsb_5(L_0) = 0$, so cannot control the rotation amount of the 1st half-round at all. In [3], Test 1 and Test 3 are experimented and investigated, but Test 2 is not discussed.

Table 3 shows the experimental results of Test 1, 2, and 3 which describe the number of plaintexts required for χ^2-value exceeding 44.99. These tests are computed to the second decimal place, and the χ^2-value is computed on the average of 100 different keys. From Table 3, we see that the χ^2-value in setting only $lsb_5(L_0) = 0$ (Test 3) is much lower than Test 1, and also lower Test 2. This means that fixing the rotation amount of the 1st half-round, that is $lsb_5(R_0) = 0$, causes highly nonuniform distribution. In fact, Test 3 requires about 2^5 times as many plaintexts as Test 1 in order to get the same effect as Test 1. This is why Test 3 has no advantage to Test 1 even if available number of plaintexts is considered. The same result on Test 1 and 3 is also reported in [3] although Test 2 is neither experimented nor investigated.

Next we focus on the effect of $lsb_5(L_0) = 0$ under the condition that the rotation amount of the 1st half-round is fixed. From Table 3, we see that in each half-round, the χ^2-value in Test 1 is higher than that in Test 2, but that almost the same effect of Test 1 is expected in Test 2 if we use about 2^2 times plaintexts as many as Test 1. On the other hand, each available number of plaintexts in Test $1(lsb_5(L_0), lsb_5(R_0) = 0)$ or Test $2(lsb_5(R_0) = 0)$ is 2^{54} or 2^{59}, respectively. From Table 3, each number of plaintexts required for χ^2-value exceeding 44.99 on h half-rounds, $log_2(\#text)$, is estimated

$$log_2(\#text) = 3.03h - 3.21 \text{ (Test 1)},$$
$$log_2(\#text) = 2.93h - 0.73 \text{ (Test 2)}$$

by using the least square method. Therefore by substituting each available number of plaintexts, we conclude that the case of Test 1 or Test 2 is estimated to be distinguishable from a random sequence by 18 half-rounds or 20 half-rounds, respectively. As a result, Test 2 is more advantageous than Test 1.

4.2 Test 4

Next, we observe the experimental results of Test 4, which are shown in Figure 2. As we have discussed the above, setting $lsb_5(R_0) = 0$ means to fix the rota-

Table 3 #texts required for χ^2-value > 44.99 in Test 1, 2 and 3(on the average of 100 keys)

#half-rounds	4	5	6	7	8	9	10
Test 1($\log_2(\#texts)$)	9.08	11.77	14.92	18.05	20.93	24.36	26.98
Test 2($\log_2(\#texts)$)	10.94	14.05	16.83	19.84	22.79	25.74	28.57
Test 3($\log_2(\#texts)$)	14.26	17.26	19.62	23.14	25.75	—	—

tion amount in the first round. Note that the rotation amount is not necessarily equal to 0. Therefore the same effect as $lsb_5(R_0) = 0$ would be expected if only fixing $lsb_5(R_0)$. Test 4 examines the hypothesis. In Figure 2, the horizontal line corresponds to the fixed value of $lsb_5(R_0)$ and the vertical line corresponds to required plaintexts in order to exceed the threshold χ^2-value of 44.99 for each $lsb_5(R_0)$. From Figure 2, we see that any $lsb_5(R_0)$ can be distinguished from a random permutation in almost the same way as $lsb_5(R_0) = 0$, i.e. just fixing the rotation amount in the 1st half-round. To sum up, we do not have to set $lsb_5(R_0) = 0$ in order to increase the χ^2-value. We use any R_0 by just classifying it into the same $lsb_5(R_0)$. As a result, we can use all plaintexts(2^{64}) to attack on RC5.

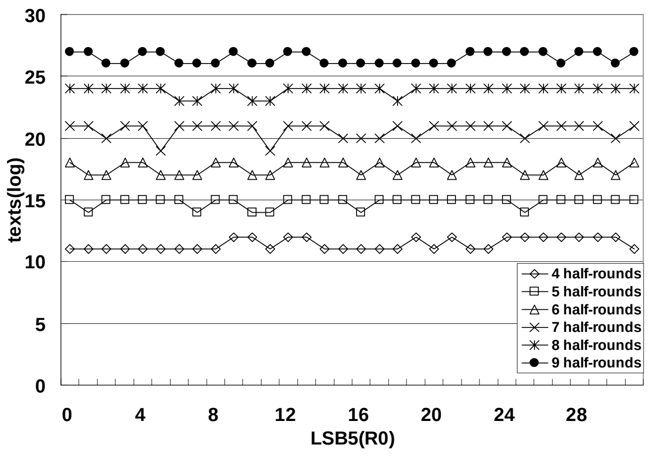

Fig. 2 #texts for distinguishing from random permutation in each $lsb_5(R_0)$ (on the average of 100 keys)

4.3 Test 5

We observe the experimental results of Test 5, which are shown in Figure 3. The horizontal line corresponds to the first bit of consecutive 5 bits of L_{h+1}, and each plot presents required plaintexts in order to exceed the threshold χ^2-value of 44.99 for each consecutive 5 bits. In the case of $i = 1$, it corresponds to $L_{h+1}^{[5,1]}$. In the case of $i = 32$, it corresponds to L_{h+1}^{32}, $L_{h+1}^{[4,1]}$. In Test 5, we compute the χ^2-value in each part with 5 bits, and compute how many plaintexts are required in order to exceed the threshold χ^2-value of 44.99. From Figure 3, we see that any consecutive five bit can be distinguished from a random permutation in almost the same way as $L_{h+1}^{[5,1]}$. In other word, correlations are observed on any consecutive five bits of L_{h+1}.

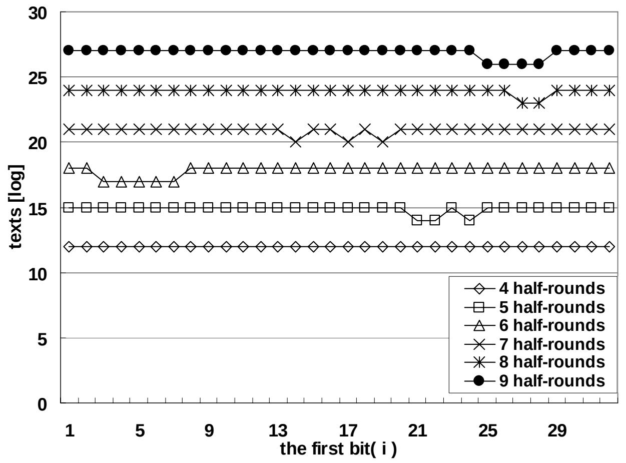

Fig. 3 #texts for distinguishing from random permutation in each part (on the average of 100 keys)

5. Key recovery algorithm

In this section, we present a key recovery algorithm, called Main algorithm, by using the results of tests in Section 4.

5.1 Key recovery of the least significant 4 bits

Main algorithm is designed by making use of the results of tests as follows:

1. Only $lsb_5(R_0)$ is fixed to 0 (**Test 1, 2**);
2. The parts measured by χ^2-statistic are not fixed to $lsb_5(L_{h+1})$ (**Test 5**);
3. The χ^2-value is computed on z to which consecutive 5 bits y is exactly decrypted by 1 half-round (see Figure 4);
4. The decrypted z is classified into 32 cases according to $lsb_5(L_{h+1}) = x$, and the χ^2-value is computed on each distribution of z for each $lsb_5(L_{h+1}) = x$.

In Section 6, Main algorithm is extended to a known plaintext correlation attack in such a way that uses any fixed $lsb_5(R_0)$ by using the result of **Test 4**. This algorithm is called Extended algorithm. Note that in Main algorithm or Extended algorithm, the number of available plaintexts is 2^{59} or 2^{64}, respectively.

Algorithm 3 (Main algorithm):
`This algorithm recovers` $lsb_4(S_{h+1})$. `Let`

$(lsb_5(L_{h+1}), lsb_5(R_{h+1})) = (x, y)$, where x is just rotation amounts in the h-th half-round.

1. Choose a plaintext (L_0, R_0) with $lsb_5(R_0)$=0, and encrypt it.
2. For each candidate $s(s = 0, 1, \cdots, 15)$ of $lsb_4(S_{h+1})$, set $S^5_{h+1} = 0$, and decrypt R_{h+1} by 1 half-round. Note that we know the exact rotation number x in the final round, and that we exactly know where y is decrypted by 1 half-round shown in Figure 4, which is set to z.
3. For each value s, x, and z, we update each array by incrementing $count[s][x][z]$.
4. For each s and x, compute $\chi^2[s][x]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][x]\}_x$ for each s, and output s with the highest $ave[s]$ as $lsb_4(S_{h+1})$.

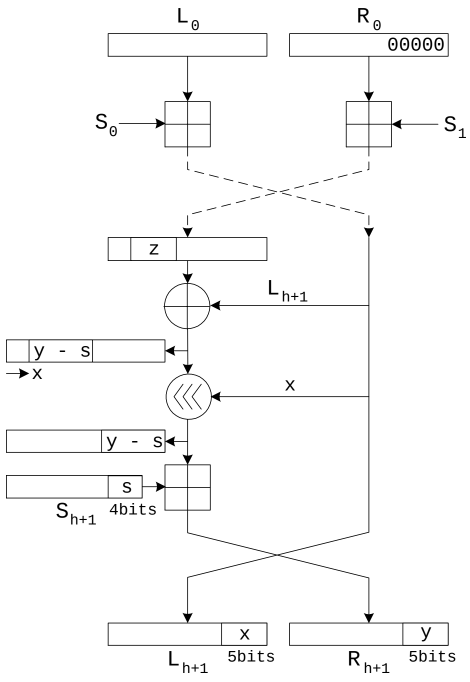

Fig. 4 Outline of our key recovery algorithm

Main algorithm computes the χ^2-value on z decrypted of exactly y by the final round subkey. Therefore Main algorithm estimates only $lsb_4(S_{h+1})$ since the χ^2-value in $lsb_5(S_{h+1}) = 1s$ is the same as that in $lsb_5(S_{h+1}) = 0s$ in the following reason. For $s = lsb_4(S_{h+1})$ we set two candidates t and t' of $lsb_5(S_{h+1})$ to $t = 1s$ and $t' = 0s$. So $t = t' + 16 \pmod{32}$. We also set $(lsb_5(L_{h+1}), lsb_5(R_{h+1})) = (x, y)$, and the decrypted value of y by using t or t' to z or z', respectively. Then $z^{[4,1]} = z'^{[4,1]}$ and $z^5 = \overline{z'^5}$. Therefore $z = z' + 16$ (mod 32). As a result, each distribution by using t or t', $count[t][x][z]$ or $count[t'][x][z']$ satisfies

$$count[t][x][z] = count[t'][x][z' + 16 \pmod{32}],$$

so we get $\chi^2[t][x] = \chi^2[t'][x]$ from the definition of χ^2-value. This is why the χ^2-value in $S^{[5,1]}_{h+1} = 1s$ is the same as that in $S^{[5,1]}_{h+1} = 0s$ for $s = S^{[4,1]}_{h+1}$.

Figure 5 and Table 4 show the success probability among 100 trials for RC5 with $4, \cdots, 10$ half-rounds. More precise experimental results are shown in Table 5. In our policy, we design all experiments as precisely as possible. Usually experimental results are shown rather roughly such as Table 1 and 4. For example, 2^{28} plaintexts can recover a key with the success probability 82%[2] seen in Table 1. However it is not necessarily minimum number of plaintexts that can recover keys with the success probability 82%. Generally the efficiency of algorithm is discussed with the success probability of 80% or 90%. So such discussion will be done rather roughly. In our experiment, we calculate all experiments to the second decimal place which is shown in Table 5. From Table 5, the number of plaintexts required for recovering a key on r rounds(h half-rounds) with the success probability of 90%, $log_2(\#text)$, is estimated

$$\begin{aligned} log_2(\#text) &= 6.23r + 0.07 \\ (log_2(\#text) &= 3.12h + 0.07) \end{aligned}$$

by using the least square method. By substituting $log_2(\#text) = 59$, we conclude that our algorithm is estimated to recover a key on RC5 with 18 half-rounds with $2^{56.23}$ plaintexts in the success probability of 90%. In the case of success probability of 45%, the number of required plaintexts is estimated as follows,

$$\begin{aligned} log_2(\#text) &= 6.18r - 1.47 \\ (log_2(\#text) &= 3.09h - 1.47). \end{aligned}$$

Therefore our algorithm can recover a key correctly on RC5 until 19 half-rounds with $2^{57.24}$ plaintexts in the success probability of 45%. As we have seen the above result, it is indispensable to increase available plaintexts in order to break RC5 by higher round. In Section 6, we will extend Main algorithm to break RC5 by 20 half-rounds.

In our approximation, we do not use the result of 4 half-rounds but all other results except for 4 half-rounds in order to improve the approximation efficiency. The key recovery with 4 half-rounds examines a bias for distribution of consecutive 5 bits in L_4, which is equivalent to examine a bias for distribution of a consecutive 5 bits in L_2. A bias for distribution of L_2 depends on the 2nd half-round operation, S_0, and plaintexts R_0. This is why, compared with key recovery of more than 4 half-rounds, key recovery with 4 half-rounds depends deeply on the choice of S_0, and plaintexts R_0. In fact, we have examined the coefficient of determination([12]) for each approximation polynomial, which evaluates the approximation efficiency of the least square method: the coefficient of determination for approximation polynomial

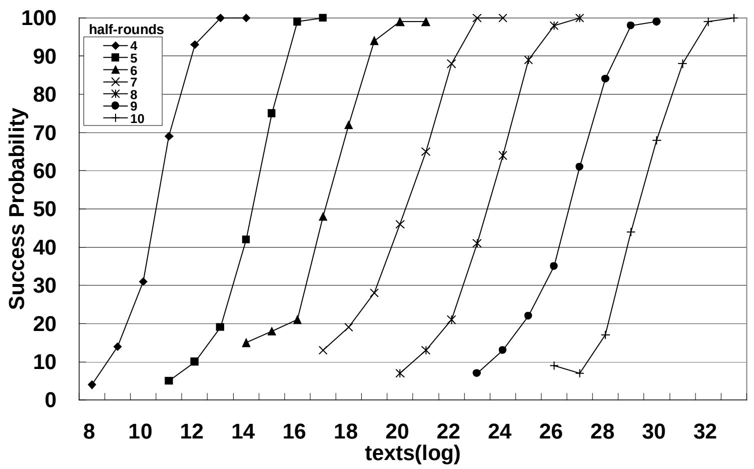

Fig. 5 Success probability of Main algorithm(in 100 trials)

Table 4 Success probability of Main algorithm(in 100 trials)

4 half-rounds		5 half-rounds		6 half-rounds	
#texts	#keys	#texts	#keys	#texts	#keys
2^{10}	31	2^{14}	42	2^{17}	48
2^{11}	69	2^{15}	75	2^{18}	72
2^{12}	93	2^{16}	99	2^{19}	94

7 half-rounds		8 half-rounds		9 half-rounds		10 half-rounds	
#texts	#keys	#texts	#keys	#texts	#keys	#texts	#keys
2^{20}	46	2^{23}	41	2^{26}	35	2^{29}	44
2^{22}	88	2^{25}	89	2^{28}	84	2^{31}	88
2^{23}	100	2^{26}	98	2^{29}	98	2^{32}	99

Table 5 # texts required for recovering a key with the success probability 90% and 45% in Main algorithm

#half-rounds	4	5	6	7	8	9	10
$\log_2(\#text)$ (90%)	11.89	15.43	18.78	22.07	25.15	28.28	30.92
$\log_2(\#text)$ (45%)	10.76	13.94	17.23	19.96	23.22	26.59	29.30

including the result of 4 half-rounds is worse than that for approximation polynomial without the result of 4 half-rounds. This is why we do not use the result of 4 half-rounds but all other results, 5-10 half rounds. Note that the only approximation polynomial with the result of 4 half-rounds indicates lower coefficient of determination. We get almost the same approximation polynomial even if the result of any half-round except 4 half-rounds is used.

5.2 Key recovery of any consecutive 4 bits

Here discusses two algorithms that recover any consecutive 4 bits: the serial key recovery algorithm and the parallel key recovery algorithm.

5.2.1 The serial key recovery algorithm

The serial key recovery algorithm recovers each 4 bits sequentially from $S_{h+1}^{[4,1]}$ to $S_{h+1}^{[28,25]}$, and 3-bit $S_{h+1}^{[31,29]}$ by using Algorithm 3. For example, in the case of recovering $S_{h+1}^{[8,5]}$, we set $S_{h+1}^{[4,1]}$ to the value recovered before and apply Algorithm 3 by setting $s = S_{h+1}^{[8,5]}$ and $y = R_{h+1}^{[9,5]}$. After repeating the above procedures by eight times, $S_{h+1}^{[31,1]}$, that is all bits of S_{h+1} except for MSB of S_{h+1} are recovered. Our experimental results of serial key recovery are shown in Figure 6 and Table 6. The results in Table 6 are divided into two cases of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and $S_{h+1}^{[31,29]}$ since only $S_{h+1}^{[31,29]}$ is 3-bit estimation. Table 6 shows the average of success probability of each interval of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and the success probability of $S_{h+1}^{[31,29]}$. Compared with Table 6 and 4, we see that any key of intervals from $S_{h+1}^{[31,29]}$ to $S_{h+1}^{[8,5]}$ can be recovered with almost the same high probability as $S_{h+1}^{[4,1]}$. In the case of 6 half-rounds ($h = 6$), 8 half-rounds ($h = 8$), and 10 half-rounds ($h = 10$), we can recover $S_{h+1}^{[31,1]}$ with a success probability of about 95%, 98%, and 97% by using about 2^{19}, 2^{26}, and 2^{32} plaintexts on the average, respectively.

5.2.2 The parallel key recovery algorithm

We have seen that the serial key recovery algorithm can recover the final half-round key $S_{h+1}^{[31,1]}$ with the sig-

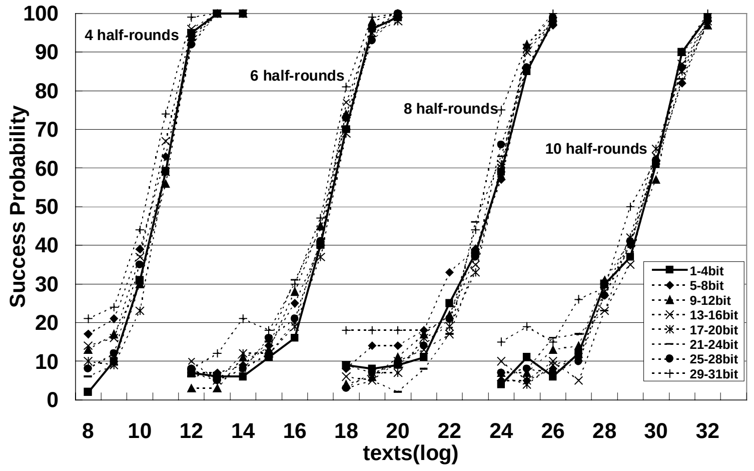

Fig. 6 Success probability of serial key recovery(in 100 trials)

Table 6 Success probability of serial key recovery (the average of 6*100 trials of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and $S_{h+1}^{[31,29]}$ (100 trials an interval))

4 half-rounds			5 half-rounds			6 half-rounds		
	#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{10}	32.3	44	2^{14}	45.0	57	2^{16}	24.0	30
2^{11}	60.7	74	2^{15}	80.2	90	2^{18}	72.7	81
2^{12}	94.0	99	2^{16}	98.3	100	2^{19}	95.3	99

7 half-rounds			8 half-rounds			9 half-rounds			10 half-rounds		
	#keys			#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{19}	26.0	38	2^{23}	38.2	44	2^{26}	35.8	41	2^{28}	26.5	29
2^{22}	89.2	96	2^{25}	88.5	92	2^{28}	87.2	90	2^{31}	85.5	90
2^{23}	98.2	100	2^{26}	98.2	100	2^{29}	97.2	98	2^{32}	97.7	100

I^*: the average of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$, II^*: $S_{h+1}^{[31,29]}$

nificantly high success probability. However, unfortunately, the serial key recovery algorithm can not work in parallel. This section investigates how to recover each subkey of $S_{h+1}^{[31,29]}$,..., $S_{h+1}^{[4,1]}$ in parallel. Before showing our parallel key recovery algorithm, we explain the next experiments.

Test 6: Apply Algorithm 3 to $S_{h+1}^{[4+4i,1+4i]}$ $(i = 0, 1, \cdots, 6)$ or $S_{h+1}^{[31,29]}$ by setting lower bits of S_{h+1} than $S_{h+1}^{[4+4i,1+4i]}$ or $S_{h+1}^{[31,29]}$ to 0. Compute the probability with which a correct key can be recovered.

Test 7: Apply Algorithm 3 to $S_{h+1}^{[4+4i,1+4i]}$ $(i = 0, 1, \cdots, 6)$ or $S_{h+1}^{[31,29]}$ by setting lower bits of S_{h+1} than $S_{h+1}^{[4+4i,1+4i]}$ or $S_{h+1}^{[31,29]}$ to 0. Compute the probability with which a key including ± 1 (mod 16) error or ± 1 (mod 8) error can be recovered respectively.

Figure 7 and Table 7 (Figure 8 and Table 8) show the experimental results in Test 6(Test 7). Table 7 or 8 shows the average of success probability of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and the success probability of $S_{h+1}^{[31,29]}$ in Test 6 or 7. Table 8 also shows the success probability of $S_{h+1}^{[4,1]}$ in Test 7. The result of $S_{h+1}^{[4,1]}$ in Test 6 is the same as that in Main algorithm (Table 4). From the experimental results, we see that the probability of recovering a correct key(Test 6) is about 50% while that of recovering a key including ± 1 error(Test 7) is more than 90%. The reason is that key recovering with ± 1 error absorbs the error bridging of lower bits much better than correct-key recovering. In fact, the success probability of $S_{h+1}^{[4,1]}$ in Test 7, which does not suffer from bridging on lower bits, is higher than that of $S_{h+1}^{[31,29]}, \cdots, S_{h+1}^{[8,5]}$ in Test 7. Let us discuss the difference between Test 7 and the serial key recovery algorithm in Section 5.2.1. From the experimental results, we see that the probability of $S_{h+1}^{[31,29]}, \cdots, S_{h+1}^{[8,5]}$ in Test 7 is almost the same but slightly worse than that in the serial key recovery algorithm. The reason is expected that the serial key recovery in $S_{h+1}^{[31,29]}, \cdots, S_{h+1}^{[8,5]}$ can compute the right bridging on lower bits by using the recovered correct keys for the lower bits.

Next we focus our mind on Test 6. Both Figure 7 and Table 7 show that Test 6 suffers completely from error bridging of lower bits. More importantly, the probability of Test 6 converges to about 50 %: however many plaintexts are used, the probability has never become higher than an upper bound. From this, we put forward a hypothesis that some specific keys are not almost recovered. In fact, we see experimentally that, in the case of recovering keys of $S_{h+1}^{[8,5]}$, $S_{h+1}^{[8,5]}$ can not be almost recovered when the lower bits $S_{h+1}^{[4,1]} = 8, \cdots, 15$. Especially when $S_{h+1}^{[4,1]} = 11, \cdots, 15$, any $S_{h+1}^{[8,5]}$ can not be recovered at all however many plaintexts are used. Thus we have observed that the success probability of key recovering in $S_{h+1}^{[8,5]}$ deeply depends on the lower bits $S_{h+1}^{[4,1]}$.

For simplicity, let us investigate the case of recovering $S_{h+1}^{[8,5]}$. The same discussion holds in other cases of $S_{h+1}^{[31,29]}, \cdots, S_{h+1}^{[12,9]}$. In Test 6, we set lower 4 bits than $S_{h+1}^{[8,5]}$ to be 0. To make the discussion clear, we denote the real value by $S_{h+1}^{[4,1]}$, and the assumed value by β. So in Tests 6, $\beta = 0$. Then for any R_{h+1}, $(R_{h+1} - S_{h+1})^{[9,5]}$ is determined by $S_{h+1}^{[9,5]}$, $R_{h+1}^{[9,5]}$, and the bridging on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$, which is estimated by $(R_{h+1}^{[4,1]} - \beta)$ in Test 6. This is why key recovering is failed if and only if the bridging on $(R_{h+1}^{[4,1]} - \beta)$ is not coincident with that on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$. The probability that bridging on $(R_{h+1}^{[4,1]} - \beta)$ is not coincident with that on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$ is different for each $S_{h+1}^{[4,1]}$. For example in the case of $S_{h+1}^{[4,1]} = 0$ a bridging on $(R_{h+1}^{[4,1]} - 0)$ is apparently coincident with that on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$ for any $R_{h+1}^{[4,1]}$. We investigate, for each $S_{h+1}^{[4,1]}$, $R_{h+1}^{[4,1]}$ that becomes an error-bridging and compute the error-bridging probability to all $R_{h+1}^{[4,1]}$. Table 9 shows the results. From Table 9, in the case of $S_{h+1}^{[4,1]} = 8, \cdots, 15$ a bridging on $(R_{h+1}^{[4,1]} - 0)$ is not coincident with that on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$ with the probability of 1/2 and over. In these keys, the χ^2-value is also computed by using invalid value with the probability of 1/2 and over. Therefore it is expected that recovering such keys is difficult even if many plaintexts are used. To observe this, we conduct an experiment on the relation between success probability of a key recovering and error-bridging probability of the key in each half-round. Table 10 shows each success probability of keys with the error-bridging probability of less than 1/2, that of 1/2 or above, and that of 10/16 or above. We see in Table 10 that: 1. keys with the error-bridging probability of less than 1/2 can be recovered correctly by using enough many plaintexts; 2. keys with the error-bridging probability of 1/2 and over cannot almost be recovered correctly even if many plaintexts are used; 3. any key with the error-bridging probability of 11/16 and over cannot be recovered correctly. From these observation, we estimate the lower bound of probability to recover a correct key, $Pr(\beta)$, as the probability of keys with the error-bridging-probability of less than 1/2,

$$Pr(\beta) = \frac{\#\text{ keys with (the error-bridging probability)}_{<1/2}}{\#\text{ keys in } S_{h+1}^{[4,1]}}.$$

Therefore we get $Pr(0) = 1/2 = 0.50$, which reflects the experimental results in Figure 7, Table 7 and 10.

In order to improve the parallel attack, we have searched all available $\beta (0 \le \beta \le 15)$ to find β with the maximum $Pr(\beta)$. The maximum $Pr(\beta)$ is $15/16 = 0.9375$, which is given by $\beta = 7, 8$. We have also investigated a type of two-valued β such as

$$\beta = \begin{cases} 4 & \text{if } R_{h+1}^{[4,1]} < 8, \\ 11 & \text{otherwise.} \end{cases}$$

However, even in this type, the maximum $Pr(\beta)$ is 15/16. There are total 256 kinds of β including two-valued, out of which 87 kinds give the maximum $Pr(\beta)$.

We discuss the improved parallel attack by using $\beta = 8$, in which $Pr(8)$ is just 15/16. Table 11 shows $R_{h+1}^{[4,1]}$ that becomes an error-bridging and the error-bridging probability for each $S_{h+1}^{[4,1]}$. Table 12 shows each success probability of keys with the error-bridging probability of less than 1/2, and that of keys with the error-bridging probability of 1/2. From Table 12, in the same way as $\beta = 0$, keys with the error-bridging probability of less than 1/2 can be recovered correctly by using enough many plaintexts. More precise experimental results are shows in Figure 9 and Table 13. From Table 13, we see that a parallel algorithm can be improved to recover a correct key with the success probability of about 90% by using roughly twice plaintexts as many as Test 7.

6. A known plaintext correlation algorithm

In this section, we extend Algorithm 3 in such a way that any plaintext, including $lsb_5(R_0) \neq 0$, can be used as follows: For given any plaintext(L_0, R_0), classify them into the same $lsb_5(R_0)$, and apply Algorithm 3. The algorithm is as follows.

Algorithm 4 (Extended algorithm):

```
This algorithm recovers $lsb_4(S_{h+1})$.  Let
$x = lsb_5(L_{h+1})$ in the same way as Algorithm 3.

1.  Given $n$ known plaintexts $(L_0, R_0)$, classify
    each plaintext $(L_0, R_0)$ into 32 cases
    $c[l] (l = 0, 1, \cdots, 31)$ according to $lsb_5(R_0)$.
2.  For each $c[l] (l = 0, 1, \cdots, 31)$, compute
    $\chi^2[l][s][x]$ according to Step 2-4 in
    Algorithm 3.
3.  Compute the average $ave[l][s]$ of $\{\chi^2[l][s][x]\}_x$
    for each $s$ and each $l$.
4.  Compute $sum[s] = \sum_{l=0}^{31} ave[l][s]$ for each $s$,
    and output $s$ with the highest $sum[s]$
```

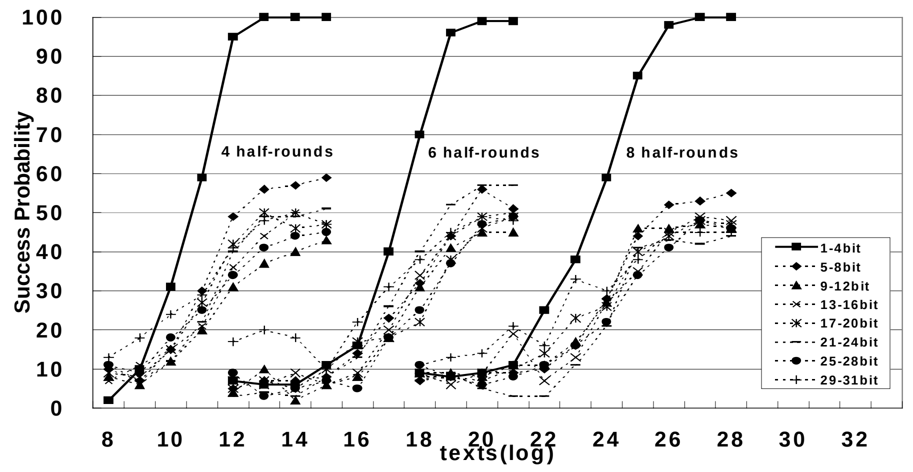

Fig. 7 Success probability of Test 6(in 100 trials)

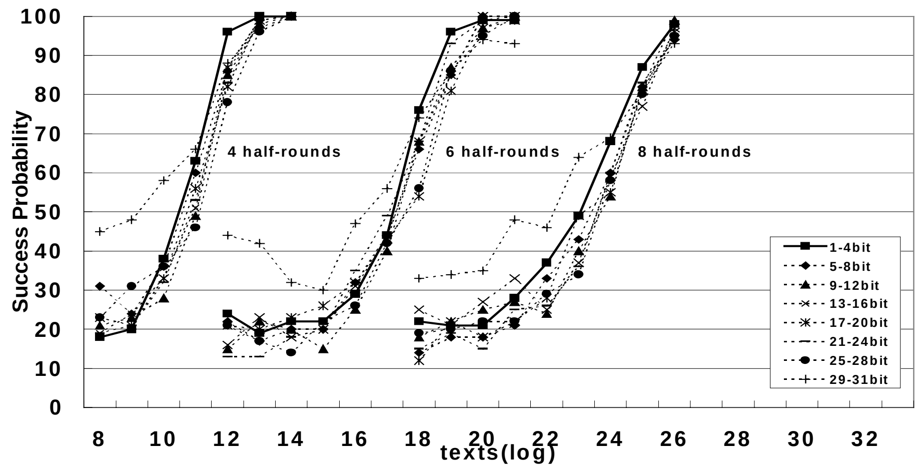

Fig. 8 Success probability of Test 7(in 100 trials)

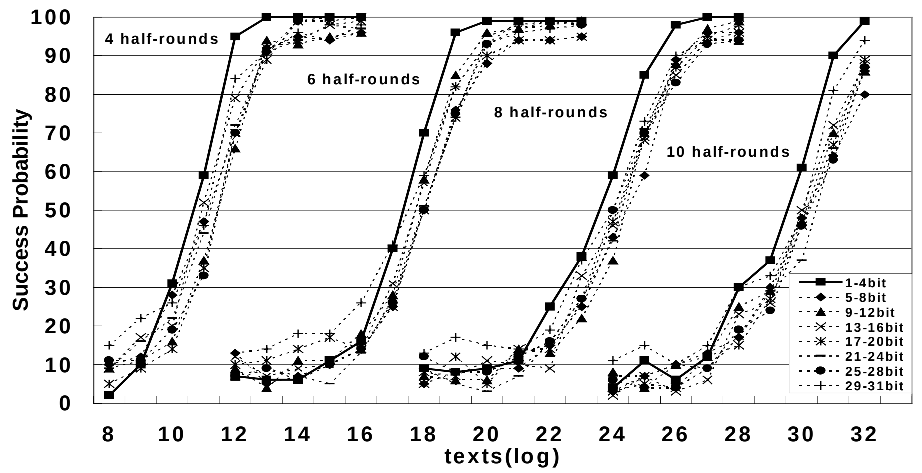

Fig. 9 Success probability of improved parallel key recovery(in 100 trials)

as $lsb_4(S_{h+1})$.

In Extended algorithm, all plaintexts are also randomly generated by m-sequences as we have described in Section 2. Therefore $lsb_5(R_0)$ of plaintexts used in our experiments are uniformly distributed in $\{0, 1, \cdots, 31\}$. Figure 10 and Table 14 show that the success probabil-

Table 7 Success probability of Test 6 (the average of 6*100 trials of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and $S_{h+1}^{[31,29]}$ (100 trials an interval))

4 half-rounds			5 half-rounds			6 half-rounds		
	#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{12}	38.7	41	2^{15}	33.2	33	2^{18}	30.7	38
2^{14}	47.7	50	2^{16}	46.8	45	2^{20}	50.0	49
2^{15}	48.7	47	2^{17}	50.3	46	2^{21}	50.3	48

7 half-rounds			8 half-rounds			9 half-rounds		
	#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{21}	25.2	35	2^{24}	25.2	30	2^{27}	26.8	23
2^{22}	37.2	42	2^{26}	45.2	45	2^{28}	39.8	47
2^{23}	43.7	46	2^{27}	47.8	45	2^{29}	49.3	56

Table 8 Success probability of Test 7 (the average of 6*100 trials of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$, $S_{h+1}^{[31,29]}$ and $S_{h+1}^{[4,1]}$ (100 trials an interval))

4 half-rounds				5 half-rounds				6 half-rounds			
	#keys				#keys				#keys		
#texts	I^*	II^*	III^*	#texts	I^*	II^*	III^*	#texts	I^*	II^*	III^*
2^{10}	33.0	58	38	2^{14}	41.2	65	56	2^{16}	30.2	49	29
2^{11}	52.5	66	63	2^{15}	67.2	82	82	2^{18}	63.3	78	76
2^{12}	83.5	88	96	2^{16}	93.0	95	99	2^{19}	86.2	92	96
2^{13}	98.3	97	100	2^{17}	98.8	100	100	2^{20}	98.0	99	99

7 half-rounds				8 half-rounds				9 half-rounds			
	#keys				#keys				#keys		
#texts	I^*	II^*	III^*	#texts	I^*	II^*	III^*	#texts	I^*	II^*	III^*
2^{19}	33.2	58	41	2^{23}	39.8	66	49	2^{25}	30.2	52	35
2^{22}	81.2	91	87	2^{25}	80.8	86	87	2^{28}	79.2	93	83
2^{23}	97.2	98	100	2^{26}	95.8	99	98	2^{29}	94.3	96	97

I^*: the average of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$, II^*: $S_{h+1}^{[31,29]}$, III^*: $S_{h+1}^{[4,1]}$

Table 9 Error-bridging $R_{h+1}^{[4,1]}$ and the probability for $S_{h+1}^{[4,1]}$ ($\beta = 0$)

$S_{h+1}^{[4,1]}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
error-bridging $R_{h+1}^{[4,1]}$	-	0	0,1	0,1,2	$0, \cdots, 3$	$0, \cdots, 4$	$0, \cdots, 5$	$0, \cdots, 6$	$0, \cdots, 7$	$0, \cdots, 8$	$0, \cdots, 9$	$0, \cdots, 10$	$0, \cdots, 11$	$0, \cdots, 12$	$0, \cdots, 13$	$0, \cdots, 14$
Probability	0	1/16	2/16	3/16	4/16	5/16	6/16	7/16	1/2	9/16	10/16	11/16	12/16	13/16	14/16	15/16

Table 10 Success probability of each key recovering in $S_{h+1}^{[8,5]}$ with each error-bridging probability ($\beta = 0$, in 100 keys)

Error-bridging		4 half-rounds			5 half-rounds			6 half-rounds			7 half-rounds			8 half-rounds		
probability	$S_{h+1}^{[4,1]}$	2^{14}	2^{17}	2^{20}	2^{17}	2^{20}	2^{23}	2^{20}	2^{23}	2^{26}	2^{23}	2^{26}	2^{29}	2^{26}	2^{29}	2^{32}
$< 1/2$	$0, \cdots, 7$	54/55	55/55	55/55	44/45	45/45	45/45	51/51	51/51	51/51	45/50	49/50	49/50	47/51	51/51	51/51
$\geq 1/2$	$8, \cdots, 15$	3/45	2/45	2/45	6/55	6/55	4/55	5/49	1/49	2/49	10/50	2/50	4/50	5/49	5/49	4/49
$\geq 11/16$	$11, \cdots, 15$	0/23	0/23	0/23	0/38	0/38	0/38	0/32	0/32	0/32	0/29	0/29	0/29	0/32	0/32	0/32

ity among 100 trials for RC5 with $4 - 10$ half-rounds. More precise experimental results are shown in Table 15. From Table 15, the number of plaintexts required for recovering a key on r rounds(h half-rounds) with the success probability of 90%, $log_2(\#text)$, is estimated

$$log_2(\#text) = 6.14r + 2.27$$
$$(log_2(\#text) = 3.07h + 2.27)$$

by using the least square method. By substituting

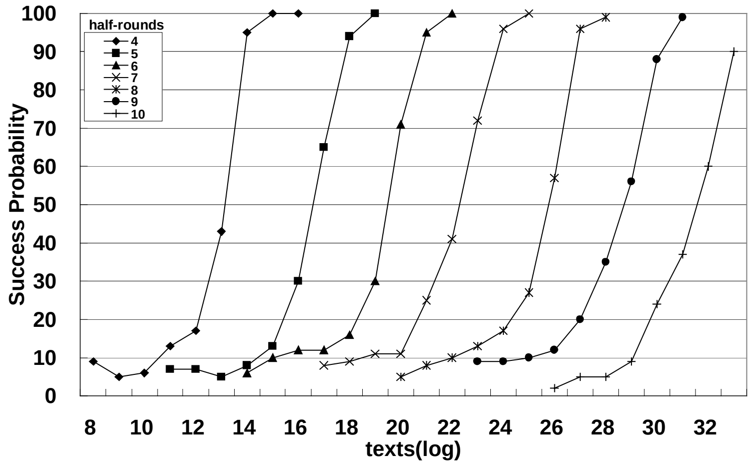

Fig. 10 Success probability of Extended algorithm(in 100 trials)

$log_2(\#text)$ = 64, we conclude that our algorithm is estimated to recover a key on RC5 with 20 half-rounds with $2^{63.67}$ plaintexts in the success probability of 90%. In the case of success probability of 30%, the number of required plaintexts is estimated as follows,

$$log_2(\#text) = 5.90r + 1.12$$
$$(log_2(\#text) = 2.95h + 1.12).$$

Therefore our algorithm can recover a key correctly on RC5 until 21 half-rounds with $2^{63.07}$ plaintexts in the success probability of 30%.

As for Extended algorithm, both the serial and improved parallel key recovery algorithm also work for recovering $S_{h+1}^{[31,1]}$ in the same way as Main algorithm.

6.1 Further discussion

Here we discuss the difference between Extended algorithm and Main algorithm. Extended algorithm requires about 2^2 times as many plaintexts as Main algorithm in order to recover correct keys as we have seen in Table 14 and 4. In our experiments all plaintexts are randomly generated. Therefore, strictly speaking, the χ^2-value in Extended algorithm is computed by using about 2^{-3} times as many plaintexts as Main algorithm since the χ^2-value in Extended algorithm is computed for each $lsb_5(R_0)$. As a result, Extended algorithm can recover a key correctly with the lower χ^2-value. We investigate experimentally the relation between χ^2-value of correct keys and that of wrong keys in both algorithms. Table 16 shows each average and variance in 100 keys. As for wrong keys, the highest χ^2-value among wrong keys is shown, which often causes to recover a wrong key. From Table 16, we see that the average among χ^2-value of correct keys in Extended algorithm is lower and the variance is much lower than that in Main algorithm. We expect that Extended algorithm reduces the variant of χ^2-value by using not specific $lsb_5(R_0)$ but all $lsb_5(R_0)$ and can recover a key correctly with the lower χ^2-value.

7. Conclusions

In this paper we have proposed a known plaintext correlation attack on RC5, which improves a chosen plaintext correlation attack on RC6 proposed by Knudsen-Meier. We have also shown three algorithms, main algorithm, a serial key recovery algorithm, and a parallel key recovery algorithm. Main algorithm recovers each consecutive 4-bit subkey of the final half-round on RC5. Two algorithms, a serial key recovery algorithm, and a parallel key recovery algorithm recover 31-bit subkey of the final half-round on RC5 by using main algorithm: a serial key recovery algorithm works in serial with the significantly high success probability, and a parallel key recovery algorithm works in parallel with rather lower success probability and much more plaintexts than serial algorithm.

Our correlation attack has been estimated to break RC5-32/r with a success probability of 90% by using $2^{6.14r+2.27}$ plaintexts. Therefore our attack can break RC5-32 with 20 half-rounds(10 rounds) by $2^{63.67}$ plaintexts. Furthermore our improved attack can break RC5-32 with 21 half-rounds in a success probability of 30% by using $2^{63.07}$ plaintexts.

Acknowledgments

The authors wish to thank Shiho Moriai and Takeshi Shimoyama for their valuable comments. The authors expresses my gratitude to Motoji Ohmori for helpful comments. The authors are also grateful to 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,

1403(1998), Springer-Verlag, 85-99.

[2] J. Borst, B. Preneel, and J. Vandewalle, "Linear Cryptanalysis of RC5 and RC6," *Proceedings of Fast Software Encryption*, Lecture Notes in Computer Science, **1636**(1999), Springer-Verlag, 16-30.

[3] 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.

[4] 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, **963**(1995), Springer-Verlag, 171-184.

[5] J. Kelsey, B. Schneier, and D. Wagner, "Mod n Cryptanalysis, with applications against RC5P and M6," *Proceedings of Fast Software Encryption*, Lecture Notes in Computer Science, **1636**(1999), Springer-Verlag, 139-155.

[6] L. Knudsen, and W. Meier, "Improved Differential Attacks on RC5," *Advances in Cryptology-Proceedings of CRYPTO'96*, Lecture Notes in Computer Science, **1109**(1996), Springer-Verlag, 216-228.

[7] L. Knudsen, and W. Meier, "Correlations in RC6 with a reduced number of rounds" *Proceedings of Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, to appear.

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

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

[10] R. Rivest, M. Robshaw, R. Sidney and Y. Yin, "The RC6 Block Cipher. v1.1," 1998.

[11] R. Rivest, "The RC5 Encryption Algorithm," *Proceedings of Fast Software Encryption*, Lecture Notes in Computer Science, **1008**(1995), Springer-Verlag, 86-96.

[12] S. Shirohata, *An introduction of statistical analysis*, Kyouritu Syuppan, 1992, (in Japanese).

Atsuko Miyaji received the B. Sc., the M. Sc., and Dr. Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997 respectively. She joined Matsushita Electric Industrial Co., LTD from 1990 to 1998 and engaged in research and development for secure communication. She has been an associate professor at JAIST(Japan Advanced Institute of Science and Technology) since 1998. Her research interests include the application of projective varieties theory into cryptography and information security. She is a member of the International Association for Cryptologic Research and the Information Processing Society of Japan.

Masao Nonaka received the degree of B.Sc. in Computer Science and Engineering from University of Aizu, JAPAN, in 2000. He has joined Japan Advanced Institute of Science and Technology(JAIST) since 2000. He is currently a master student at School of Information Science. His current research field is cryptology, especially cryptanalysis regarding symmetric key cryptsystem.

Yoshinori Takii received the B.E. from the Department of Computer Science, National Defense Academy and the M. info. Sc. from JAIST in 1999 and 2001 respectively. He has joined Japan Air Self Defese Force since 1995 and engages in research and development of secret key ciphers. He had studied the information security in JAIST from 1999 to 2001.

Table 11 Error-bridging $R_{h+1}^{[4,1]}$ and the probability for $S_{h+1}^{[4,1]}(\beta = 8)$

$S_{h+1}^{[4,1]}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
error-bridging $R_{h+1}^{[4,1]}$	$0, \cdots, 7$	$1, \cdots, 7$	$2, \cdots, 7$	$3, \cdots, 7$	$4, \cdots, 7$	5, 6, 7	6, 7	7	–	8	8, 9	8, 9, 10	$8, \cdots, 11$	$8, \cdots, 12$	$8, \cdots, 13$	$8, \cdots, 14$
Probability	1/2	7/16	6/16	5/16	4/16	3/16	2/16	1/16	0	1/16	2/16	3/16	4/16	5/16	6/16	7/16

Table 12 Success probability of each key recovering in $S_{h+1}^{[8,5]}$ with each error-bridging probability ($\beta = 8$, in 100 keys)

Error-bridging		4 half-rounds			5 half-rounds			6 half-rounds			7 half-rounds			8 half-rounds		
probability	$S_{h+1}^{[4,1]}$	2^{14}	2^{17}	2^{20}	2^{17}	2^{20}	2^{23}	2^{20}	2^{23}	2^{26}	2^{23}	2^{26}	2^{29}	2^{26}	2^{29}	2^{32}
$< 1/2$	$1, \cdots, 15$	91/92	92/92	92/92	87/92	91/92	91/92	83/92	89/92	90/92	89/96	94/96	95/96	86/95	94/95	94/95
1/2	0	4/8	4/8	5/8	5/8	4/8	4/8	5/8	6/8	5/8	2/4	2/4	3/4	3/5	2/5	4/5

Table 13 Success probability of improved parallel key recovery (the average of 6*100 trials of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ and $S_{h+1}^{[31,29]}$ (100 trials an interval))

4 half-rounds			5 half-rounds			6 half-rounds		
	#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{11}	41.3	46	2^{14}	29.2	48	2^{18}	52.7	59
2^{12}	71.2	84	2^{15}	57.7	55	2^{19}	77.7	82
2^{13}	91.5	91	2^{16}	84.7	83	2^{20}	92.5	93
2^{14}	95.7	96	2^{17}	93.2	95	2^{22}	97.2	97

7 half-rounds			8 half-rounds			9 half-rounds			10 half-rounds		
	#keys			#keys			#keys			#keys	
#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*	#texts	I^*	II^*
2^{20}	30.7	49	2^{23}	26.7	37	2^{27}	42.8	55	2^{30}	45.7	48
2^{22}	71.7	73	2^{26}	86.5	90	2^{28}	68.2	78	2^{31}	67.0	81
2^{23}	90.3	93	2^{27}	95.0	94	2^{29}	88.2	90	2^{32}	86.0	94

I^*: the average of $S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$, II^*: $S_{h+1}^{[31,29]}$

Table 14 Success probability of Extended algorithm (in 100 trials)

4 half-rounds		5 half-rounds		6 half-rounds	
#texts	#keys	#texts	#keys	#texts	#keys
2^{13}	43	2^{16}	30	2^{19}	30
2^{14}	95	2^{17}	65	2^{20}	71
2^{15}	100	2^{18}	94	2^{21}	95

7 half-rounds		8 half-rounds		9 half-rounds		10 half-rounds	
#texts	#keys	#texts	#keys	#texts	#keys	#texts	#keys
2^{22}	41	2^{25}	27	2^{28}	35	2^{31}	37
2^{23}	72	2^{26}	57	2^{30}	88	2^{32}	60
2^{24}	96	2^{27}	96	2^{31}	99	2^{33}	90

Table 15 # texts required for recovering a key with the success probability 90%, 45%, and 30% in Extended algorithm

#half-rounds	4	5	6	7	8	9	10
$\log_2(\#text)$ (90%)	13.96	17.73	20.63	23.71	26.64	30.01	33.00
$\log_2(\#text)$ (45%)	13.09	16.39	19.41	22.14	25.62	28.43	31.52
$\log_2(\#text)$ (30%)	12.53	15.94	18.81	21.63	25.07	27.53	30.70

Table 16 The χ^2-value of correct keys and wrong keys in 4 half-rounds (in 100 trials)

	# texts	Key recovering probability	Correct keys		Wrong keys*	
			Average	Variance	Average	Variance
Main algorithm	2^{12}	93%	40.50	3.69	38.42	2.41
Extended algorithm	2^{14}	95%	32.31	0.10	32.08	0.09

*: The highest χ^2-value among wrong keys is used for each trial.