Known plaintext correlation attack against RC5

Atsuko Miyaji[1], Masao Nonaka[1], and Yoshinori Takii[2]

[1] School of Information Science, Japan Advanced Institute of Science and Technology
1-1, Asahidai, Tatsunokuchi, Nomi, 923-1292, Japan
{miyaji, m-nonaka}@jaist.ac.jp
http://grampus.jaist.ac.jp:8080/miyaji-lab/index.html
[2] Japan Air Self Defense Force
5-1, Ichigaya Honmura-cho, Shinjyuku-ku, Tokyo, 162-0845, JAPAN

Abstract. We investigate a known plaintext attack on RC5 based on correlations. Compared with the best previous known-plaintext attack on RC5-32, a linear cryptanalysis by Borst, Preneel, and Vandewalle, our attack applies to a larger number of rounds. RC5-32 with r rounds can be broken with a success probability of 90% by using $2^{6.14r+2.27}$ plaintexts. Therefore, our attack can break RC5-32 with 10 rounds (20 half-rounds) with $2^{63.67}$ plaintexts with a probability of 90%. With a success probability of 30%, our attack can break RC5-32 with 21 half-rounds by using $2^{63.07}$ plaintexts.

1 Introduction

RC5, designed by Rivest([11]), is a block cipher 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 by software with small amount of memory. RC5-32/r means that two 32-bit-block plaintexts are encrypted by r rounds, where one round consists of two half-rounds. Various attacks against RC5 have been analyzed intensively([1, 2, 4–7]). The best chosen-plaintext attack ([1]), up to the present, breaks RC5-32/12 by using $(2^{44}, 2^{54.5})$ pairs of chosen plaintexts and known plaintexts. However, it requires many 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 infeasible to employ such an algorithm on a modern computer. On the other hand, a known plaintext attack can work more efficiently and practically, even though it has not been reported far higher round like 12.

The best known-plaintext attack against RC5 is a linear cryptanalysis([2]). They have reported that RC5-32 with 10 rounds is broken by 2^{64} plaintexts under the heuristic assumption, that is, RC5-32 with r rounds is 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 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 to 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%.

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	$2^{6.14r+2.27}$	2^{15}	100/100	2^{22}	100/100	2^{28}	99/100	2^{33}	90/100
attack		2^{14}	95/100	2^{21}	95/100	2^{27}	96/100	2^{32}	60/100

In this paper, we investigate a known plaintext attack by improving a correlation attack against RC6([7]). 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. Correlation attack makes use of correlations between an input and an output, which is measured by the χ^2 test: 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 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. 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. We 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: 1. the least significant five bits of L_0, $lsb_5(L_0)$, is 0; 2. $lsb_5(R_0)$ 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 are extremely fewer because 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 a higher round. This is why their attack does not work well on RC5 directly. In fact, they also report 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 fewer available plaintexts. The important factor to target at RC5 is how to increase the available plaintexts.

We investigate how an output after h half-rounds, L_{h+1}, 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, even large one, 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 unlikely to be uniformly distributed if a plaintext introduces small rotation amounts such as a zero in the 1st half-round([7]). However, from Feature 1, output of RC5 is also highly unlikely 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 given 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 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 the 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. As a result, our attack can break RC5-32/10 with $2^{63.67}$ plaintexts in a probability of 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 2 summarizes some notations and definitions in this paper. Section 3 applies Knudsen-Meier's correlation attack to RC5, and discusses the differences between RC5 and RC6. Section 4 describes some experimental results including the above features of RC5. Section 5 presents the chosen plaintext algorithm, Main algorithm. Section 6 discusses how to extend Main algorithm to the known plaintext algorithm, Extended algorithm. Section 7 applies Extended algorithm to 31-bit key recovery in the final round.

2 Preliminary

This section denotes some notations, definitions, and experimental remarks. First we describe RC5 algorithm after defining the following notations.

$\boxplus$ ($\boxminus$): an addition(subtraction) mod 2^{32}; $\oplus$: a bit-wise exclusive OR;
r : the number of rounds; h : the number of half-rounds ($h = 2r$);
$a \lll b (a \ggg b)$: a cyclic rotation of a to the left(right) by b bits;
(L_i, R_i): an input of the i-th half-round, and (L_0, R_0) is a plaintext;
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 denote the least significant bit(LSB) to 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: 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. Two consecutive half-rounds correspond to one round of RC5.

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

We make use of the χ^2-tests for distinguishing a random sequence from non-random sequence([5, 7, 8]). 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, $\chi^2(X)$, estimates the difference between X and the uniform distribution as follows: $\chi^2(X) = \frac{m}{n} \sum_{i=0}^{m-1} \left(N_{a_i}(X) - \frac{n}{m}\right)^2$. We use the threshold for 31 degrees of freedom 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 5% if the observation X is uniform. Here, we set the level to 0.95 in order to distinguish a sequence X from a random sequence.

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

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

3 Applying Knudsen-Meier's correlation attack to RC5

Knudsen and Meier([7]) 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 $s (s = 0, 1, \cdots, 31)$, set such an x that leads to a zero rotation in the 1-st half-round, that is, 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 $y = lsb_5(L_{h+1})$
3. For each (L_0, R_0), update each array by incrementing $count[s][y]$.
4. For each s, compute the χ^2-value $\chi^2[s]$, and output s with the highest value $\chi^2[s]$ as $lsb_5(S_1)$.

Table 3. Success probability of Algorithm 2(in 100 trials)

4 half-rounds			6 half-rounds			8 half-rounds		
#texts	#keys	χ^2-value*	#texts	#keys	χ^2-value*	#texts	#keys	χ^2-value*
2^{10}	25	530.12	2^{13}	20	139.59	2^{17}	13	69.56
2^{19}	28	256034.12	2^{19}	31	6639.60	2^{21}	26	443.06
2^{23}	28	4097164.86	2^{23}	32	105660.92	2^{26}	29	13303.08

* The average of the maximum χ^2-value in each trial.

Table 3 shows the experimental results of Algorithm 2. From Table 3, we see that a correct key can not be efficiently recovered, even though the maximum χ^2-value is enough high. Apparently, plaintexts with a zero rotation in the 1st round outputs the high χ^2-value, but a small-absolute-value one such as $\pm 1, \pm 2$, etc., also outputs the high χ^2-value. Since Algorithm 2 uses only plaintexts with a zero rotation in the 1st round, it suffers from other high-χ^2-value plaintexts, and thus cannot recover keys efficiently. Algorithm 2 is distinguishable, but unrecoverable. Furthermore, Algorithm 2 can use only 2^{54} plaintexts. In general, the number of plaintexts on RC5 is not so large as RC6. Another problem may occur in recovering other bits of S_1 because the rotation in the 1st half-round is determined only by $lsb_5(S_1)$. In RC6, the rotation in the 1st half-round is determined by all bits of S_1, and thus their algorithm can work well to recover all bits of S_1. This is why it is not efficient to apply their attack to RC5.

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 a zero rotation in the 1st half-round like Algorithm 2, 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([3]), then $lsb_5(L_0) = 0$ can not fix $lsb_5(R_2)$ (i.e. $lsb_5(L_3)$) for any available plaintext. We experimentally compare the effect of $lsb_5(L_0) = 0$, $lsb_5(R_0) = 0$, or both. We also investigate which parts output the higher χ^2-statistics. To observe these, we conduct the following five experiments in each h half-round.

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

4.1 Test 1, 2, and 3

Table 4. #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	—	—

Here we show the experimental results of Test 1, 2, and 3 after discussing the differences among these Tests. The condition of $lsb_5(R_0) = 0$ means that the rotation amount of the 1st half-round is fixed. The purpose of Tests 1 and 2 is to observe the effect of handling a plaintext in the part corresponding to the χ^2-test: Test 1 handles it; and Test 2 does not handle it. On the other hand, Test 3 sets only $lsb_5(L_0) = 0$, so it cannot control the rotation amount of the 1st half-round at all.

Table 4 shows the experimental results of Test 1, 2, and 3 which represent 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 4, we see that the χ^2-value in Test 3 is much lower than that in Test 1, and also lower than that in Test 2. Test 3

requires about $2^5(2^3)$ times as many plaintexts as Test 1(Test 2) in order to get the same effect as Test 1(Test 2). As a result, we see that fixing the rotation amount of the 1st half-round, that is, $lsb_5(R_0) = 0$, causes highly nonuniform distribution, and that Test 3 has no advantage to both Tests 1 and 2.

Next we focus on the effect of $lsb_5(L_0) = 0$ under $lsb_5(R_0) = 0$. From Table 4, 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. The 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. On the other hand, the number of available plaintexts in Test 1(Test 2) is $2^{54}(2^{59})$. By substituting the number of available 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 and 5

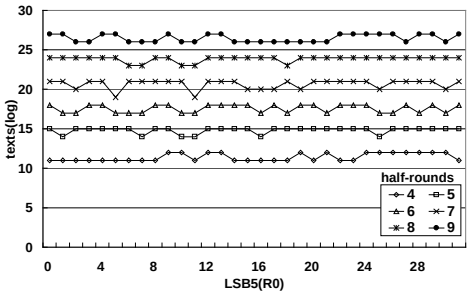

Fig. 1. #texts required for χ^2-value $>$ 44.99 in each $lsb_5(R_0)$ (on the average of 100 keys)

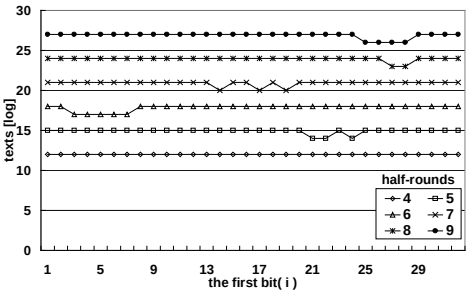

Fig. 2. #texts required for χ^2-value $>$ 44.99 in each consecutive 5 bits of L_{h+1} (on the average of 100 keys)

We observe the experimental results of Test 4 in Figure 1. As we have discussed the above, setting $lsb_5(R_0) = 0$ means to fix the rotation 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 1, the horizontal line corresponds to the fixed value of $lsb_5(R_0)$ and the vertical line corresponds to the number of plaintexts required for χ^2-value exceeding 44.99. From Figure 1, we see that any $lsb_5(R_0)$ can be distinguished from a random sequence in almost the same way as $lsb_5(R_0) = 0$. To sum up, we do not have to set $lsb_5(R_0) = 0$ in order to

increase the χ^2-value. We can use any plaintext (L_0, R_0) with any R_0 by just classifying it into the same $lsb_5(R_0)$.

In Test 5, we compute the χ^2-value in each consecutive 5 bits of L_{h+1}, and how many plaintexts are required in order to exceed the threshold χ^2-value of 44.99. Figure 2 shows the experimental results. The horizontal line corresponds to the first bit of consecutive 5 bits of L_{h+1}, and each plot presents the number of plaintexts required for χ^2-value exceeding 44.99 for each consecutive 5 bits. For example, the case of $i = 1$, or $i = 32$ corresponds to $L_{h+1}^{[5,1]}$, or $\{L_{h+1}^{32}, L_{h+1}^{[4,1]}\}$. From Figure 2, we see that any consecutive five bits can be distinguished from a random sequence in almost the same way as $L_{h+1}^{[5,1]}$. Correlations are observed on any consecutive five bits of L_{h+1}.

5 A chosen plaintext correlation algorithm

In this section, we present a key recovery algorithm, called Main algorithm. Main algorithm is designed by making use of the results of tests in Section 4 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 3);
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 $lsb_5(L_{h+1}) = x$.

Algorithm 3 (Main algorithm)
This algorithm recovers $lsb_4(S_{h+1})$. Set $(lsb_5(L_{h+1}), lsb_5(R_{h+1})) = (x, y)$, and $lsb_4(S_{h+1}) = s$, where x is the rotation amount 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 $s (s = 0, 1, \cdots, 15)$, set $S_{h+1}^5 = 0$, and decrypt R_{h+1} by 1 half-round. Note that, from the rotation amount x in the h-th half-round, we exactly know where y is decrypted by 1 half-round, 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 $\{\chi^2[s][x]\}$ for each s, and output s with the highest $ave[s]$ as $lsb_4(S_{h+1})$.

Main algorithm computes the χ^2-value on z, to which y is decrypted by the final round subkey. Therefore, the χ^2-value in $lsb_5(S_{h+1}) = 1s$ is coincident with that in $lsb_5(S_{h+1}) = 0s$ in the following reason. For $s = lsb_4(S_{h+1})$, we set two candidates of $lsb_5(S_{h+1})$, $t = 1s$ and $t' = 0s$. So $t = t' + 16 \pmod{32}$. We also set each decrypted value of $y = lsb_5(R_{h+1})$ by using each key, t or t', to z or z', respectively. Then $z^{[4,1]} = z'^{[4,1]}$, and $z^5 = \overline{z'^5}$, and thus $z = z' + 16 \pmod{32}$. Two distribution of $count[t][x][z]$ and $count[t'][x][z']$ in Algorithm 3 satisfy

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

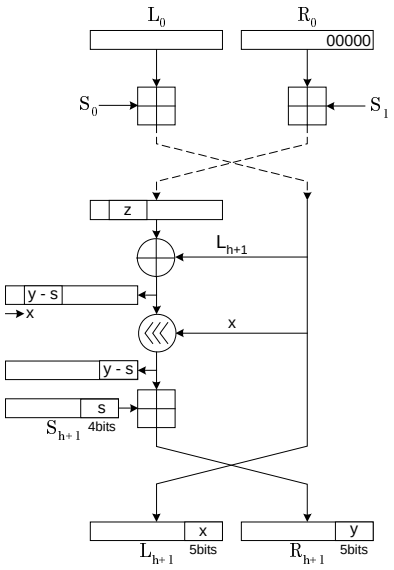

Fig. 3. Outline of Main algorithm

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 $S_{h+1}^{[5,1]} = 1s$ is coincident with that in $S_{h+1}^{[5,1]} = 0s$ for $s = S_{h+1}^{[4,1]}$.

Figure 4 and Table 5 show the success probability among 100 trials for RC5 with $4-10$ half-rounds. More precise experimental results are shown in Table 6. All experiments are calculated to the second decimal place. In our policy, we design all experiments as precisely as possible in order to estimate the efficiency of algorithm strictly. From Table 6, the number of plaintexts required for recovering a key in r rounds(h half-rounds) with the success probability of 90%, $log_2(\#text)$, is estimated[1],

$$log_2(\#text) = 6.23r + 0.07,\ (log_2(\#text) = 3.12h + 0.07),$$

by using the least square method. In the case of success probability of 45%, the number of required plaintexts is estimated as follows,

$$log_2(\#text) = 6.18r - 1.47\ (log_2(\#text) = 3.09h - 1.47).$$

[1] Our estimation is computed by using the results of 5-10 half-rounds except for 4 half-rounds because key recovery with 4 half-rounds depends deeply on the choice of S_0, and plaintexts R_0. The key recovery with 4 half-rounds examines a bias for distribution of consecutive 5 bits in L_4, which is coincident with that in L_2. A bias for distribution of L_2 depends only on the 2nd half-round operation, S_0, and plaintexts R_0. In fact, the coefficient of determination([12]) for approximation polynomial including the result of 4 half-rounds is worse than that for approximation polynomial without the result of 4 half-rounds.

By substituting $log_2(\#text) = 59$, Main algorithm can break RC5-32 with 18 rounds with $2^{56.23}$ plaintexts with a probability of 90%. With a success probability of 30%, Main algorithm can break RC5-32 with 19 half-rounds by using $2^{57.24}$ plaintexts. From these results, we see that it is indispensable to increase available plaintexts in order to break RC5 with higher round.

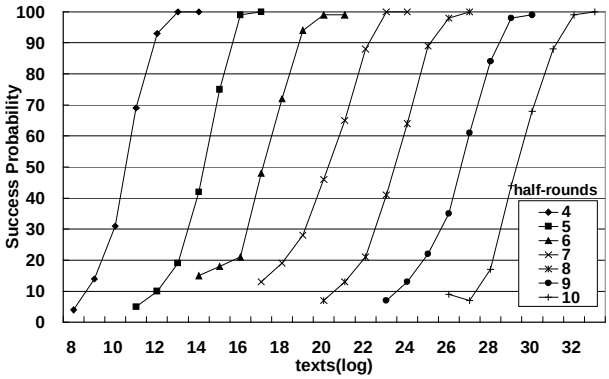

Fig. 4. Success probability of Main algorithm(in 100 trials)

6 A known plaintext correlation algorithm

In this section, we present a key recovery algorithm, called Extended algorithm, which applies Main algorithm to a known plaintext attack.

6.1 Extended algorithm

Extended algorithm classifies any plaintext (L_0, R_0) into the same $lsb_5(R_0)$, and applies Main algorithm as follows.

Algorithm 4 (Extended algorithm)
`This algorithm recovers` $lsb_4(S_{h+1})$. `Set` $s = lsb_4(S_{h+1})$, `and` $x = lsb_5(L_{h+1})$ `in the same way as Algorithm 3.`

1. `Given` n `known plaintexts` (L_0, R_0), `set` $l = lsb_5(R_0)$, `and encrypt it.`
2. `For each` l, `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` $sum[s]$ `as` $lsb_4(S_{h+1})$.

Figure 5 and Table 7 show that the success probability among 100 trials for RC5 with $4-10$ half-rounds. More precise experimental results are shown in

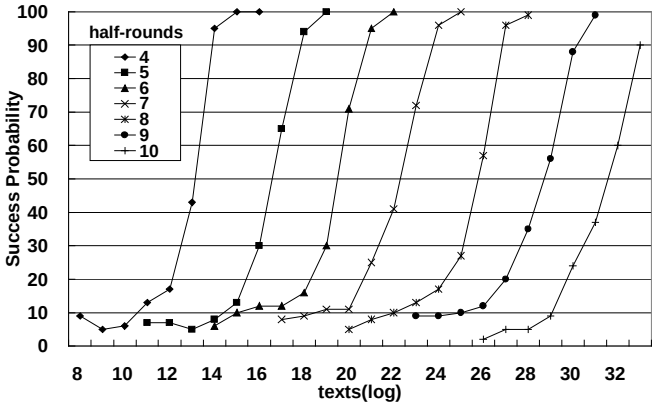

Fig. 5. Success probability of Extended algorithm(in 100 trials)

Table 8. From Table 8, 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 $log_2(\#text) = 64$, 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 on RC5 until 21 half-rounds with $2^{63.07}$ plaintexts in the success probability of 30%.

6.2 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 7 and 5. Since all plaintexts in our experiments are randomly generated by m-sequences, $lsb_5(R_0)$ of plaintexts in Extended algorithm are roughly estimated to be uniformly distributed in $\{0, 1, \cdots, 31\}$. As a result, the χ^2-value in Extended algorithm is computed by using about 2^{-3} times as many plaintexts as Main algorithm because the χ^2-value in Extended algorithm is computed for each $lsb_5(R_0)$. We investigate experimentally the relation between χ^2-value of correct keys and that of wrong keys in both algorithms. Table 9 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 9, 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 recovers a key with the lower χ^2-value. Thus, Extended algorithm can recover a key efficiently with the lower χ^2-value.

7 31-bit key recovery in the final round

Here discusses two algorithms that recover 31-bit-final-round subkey: the serial key recovery algorithm, and the parallel key recovery algorithm.

7.1 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 $S_{h+1}^{[31,28]}$ by using Algorithm 4. 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 4 by setting $s = S_{h+1}^{[8,5]}$ and $y = R_{h+1}^{[9,5]}$. As for the final 3 bits, $S_{h+1}^{[31,29]}$, we apply Algorithm 4 by using $s = S_{h+1}^{[31,28]}$ and recover $S_{h+1}^{[31,28]}$. After repeating the above procedures by eight times, $S_{h+1}^{[31,1]}$, that is, all bits of S_{h+1} except for MSB are recovered. The experimental results of serial key recovery are shown in Figure 6. Compared with Figure 6 and Table 7, we see that any key of any interval from $S_{h+1}^{[31,28]}$ 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 4 half-rounds, 6 half-rounds, or 8 half-rounds, we can recover $S_5^{[31,1]}$, $S_7^{[31,1]}$, or $S_9^{[31,1]}$ with a success probability of about 99%, 97%, and 92% by using about 2^{15}, 2^{21}, or 2^{27} plaintexts on the average, respectively.

7.2 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 significantly high success probability. However, unfortunately, the serial key recovery algorithm can not work in parallel. This section investigates how to recover each subkey of $S_{h+1}^{[31,28]}$,..., $S_{h+1}^{[4,1]}$ in parallel. Before showing our parallel key recovery algorithm, we conduct the next experiment.

Test 6: Apply Algorithm 4 to $S_{h+1}^{[4+4i,1+4i]}$ $(i = 0, 1, \cdots, 6)$ or $S_{h+1}^{[31,28]}$ by setting lower bits of S_{h+1} than $S_{h+1}^{[4+4i,1+4i]}$ or $S_{h+1}^{[31,28]}$ to 0. Compute the probability with which a correct key can be recovered.

Figure 7 shows the experimental results in Test 6. The result of $S_{h+1}^{[4,1]}$ in Test 6 is the same as that in Extended algorithm (Table 7). Figure 7 shows 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,

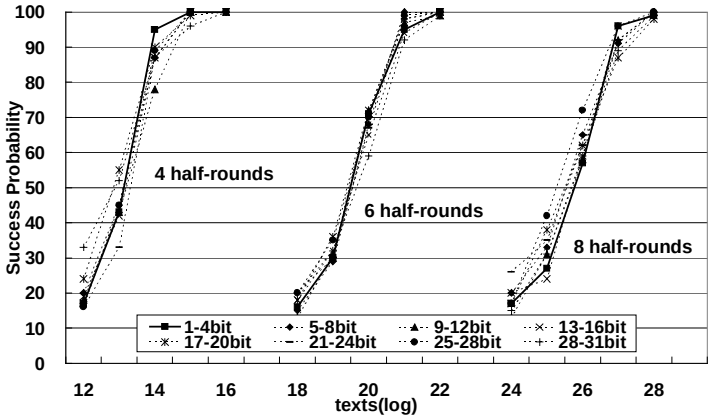

Fig. 6. Success probability of serial key recovery(in 100 trials)

the probability has never become higher than an upper bound. From this, we put forward a hypothesis that some specific keys are not recovered. In fact, we see experimentally that, in the case of recovering keys of $S_{h+1}^{[8,5]}$, $S_{h+1}^{[8,5]}$ with the lower bits $S_{h+1}^{[4,1]} = 8, \cdots, 15$ can not be almost recovered. Especially, any $S_{h+1}^{[8,5]}$ with $S_{h+1}^{[4,1]} = 13, \cdots, 15$ can not be recovered at all however many plaintexts are used. The success probability of recovering keys in $S_{h+1}^{[8,5]}$ deeply depends on the lower bits $S_{h+1}^{[4,1]}$.

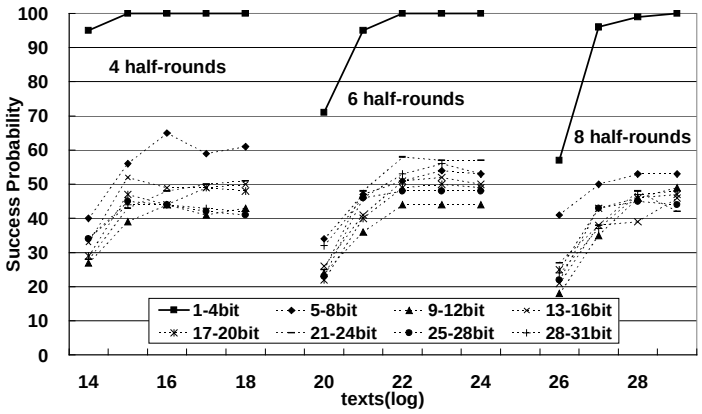

Fig. 7. Success probability of Test 6(in 100 trials)

For simplicity, let us investigate the case of recovering $S_{h+1}^{[8,5]}$. The same discussion also holds in other cases of $S_{h+1}^{[31,28]}, \cdots, S_{h+1}^{[12,9]}$. 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. In Tests 6, $\beta = 0$. 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 $(R_{h+1}^{[4,1]} - \beta)$. 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 $R_{h+1}^{[4,1]}$. Here, $R_{h+1}^{[4,1]}$ is called an error-bridging for $S_{h+1}^{[4,1]}$ if the bridging on $(R_{h+1}^{[4,1]} - \beta)$ is different with that on $(R_{h+1}^{[4,1]} - S_{h+1}^{[4,1]})$. For each $S_{h+1}^{[4,1]}$, we compute $R_{h+1}^{[4,1]}$ that is an error-bridging, and the error-bridging probability,

$$\frac{\#\{R_{h+1}^{[4,1]} \ni t \mid t \text{ is an error-bridging.}\}}{\#\{R_{h+1}^{[4,1]}\}}.$$

Table 10 shows the results. From Table 10, 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 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 the success probability of a key recovering and the error-bridging probability. Table 11 shows the success probability of keys with the error-bridging probability of less than $1/2$, that of $1/2$ or above, and that of $13/16$ or above. In Table 11, we see 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 $13/16$ 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 $1/2$,

$$\Pr(\beta) = \frac{\#\{S_{h+1}^{[4,1]} \ni t \mid \text{the error-bridging probability of } t < 1/2\}}{\#\{S_{h+1}^{[4,1]}\}}.$$

We get $Pr(0) = 1/2 = 0.50$, which reflects the experimental results in Figure 7 and Table 11.

In order to improve the parallel attack, we have searched all available $\beta (0 \leq \beta \leq 15)$ 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 12 shows error-bridging $R_{h+1}^{[4,1]}$ and the error-bridging probability for each $S_{h+1}^{[4,1]}$. Table 13 shows the 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 13, 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 shown in Table 14 and Figure 8. From Table 14, we see that the parallel algorithm can recover a 31-bit key with the success probability of about 90% by using roughly the same plaintexts as that of about 50% in Test 6, and about twice as many plaintexts as that of the serial algorithm(Figure 6).

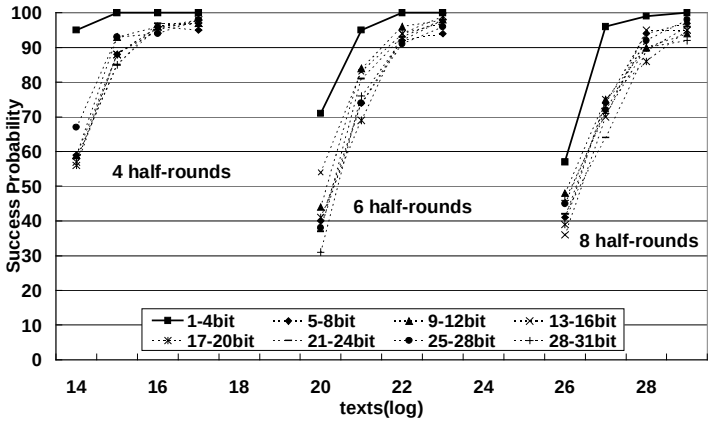

Fig. 8. Success probability of improved parallel key recovery(in 100 trials)

8 Conclusions

In this paper, we have proposed a known plaintext correlation attack on RC5. Our attack can 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. In a success probability of 30%, our attack can break RC5-32 with 21 half-rounds by using $2^{63.07}$ plaintexts.

We have also shown a parallel key recovery algorithm which recover 31-bit subkey of the final half-round on RC5. A parallel key recovery algorithm can work in parallel, and recover 31-bit keys with the high success probability.

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).

Table 5. 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 6. # 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

Table 7. 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 8. # 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 9. 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.

Table 10. 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
error-bridging $R_{h+1}^{[4,1]}$	-	0	0,1	0,1,2	$0, \cdots, 3$	$0, \cdots, 4$	$0, \cdots, 5$	$0, \cdots, 6$
Probability	0	1/16	2/16	3/16	4/16	5/16	6/16	7/16

8	9	10	11	12	13	14	15
$0, \cdots, 7$	$0, \cdots, 8$	$0, \cdots, 9$	$0, \cdots, 10$	$0, \cdots, 11$	$0, \cdots, 12$	$0, \cdots, 13$	$0, \cdots, 14$
1/2	9/16	10/16	11/16	12/16	13/16	14/16	15/16

Table 11. Success probability of each key recovering in $S_{h+1}^{[8,5]}$ with each error-bridging probability ($\beta = 0$, in 100 keys)

Error-bridging probability	$S_{h+1}^{[4,1]}$	4 half-rounds			6 half-rounds			8 half-rounds		
		2^{14}	2^{17}	2^{20}	2^{20}	2^{23}	2^{26}	2^{26}	2^{29}	2^{32}
$< 1/2$	$0, \cdots, 7$	33/55	55/55	55/55	30/51	51/51	51/51	31/51	50/51	51/51
$\geq 1/2$	$8, \cdots, 15$	7/45	4/45	5/45	4/49	3/49	2/49	10/49	3/49	2/49
$\geq 13/16$	$13, \cdots, 15$	0/13	0/13	0/13	0/23	0/23	0/23	0/22	0/22	0/22

Table 12. 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
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
Probability	1/2	7/16	6/16	5/16	4/16	3/16	2/16	1/16

8	9	10	11	12	13	14	15
–	8	8, 9	8, 9, 10	$8, \cdots, 11$	$8, \cdots, 12$	$8, \cdots, 13$	$8, \cdots, 14$
0	1/16	2/16	3/16	4/16	5/16	6/16	7/16

Table 13. Success probability of each key recovering in $S_{h+1}^{[8,5]}$ with each error-bridging probability ($\beta = 8$, in 100 keys)

Error-bridging probability	$S_{h+1}^{[4,1]}$	4 half-rounds			6 half-rounds			8 half-rounds		
		2^{14}	2^{17}	2^{20}	2^{20}	2^{23}	2^{26}	2^{26}	2^{29}	2^{32}
$< 1/2$	$1, \cdots, 15$	55/92	92/92	92/92	37/92	90/92	91/92	40/95	93/95	95/95
1/2	0	4/8	3/8	4/8	3/8	4/8	3/8	1/5	4/5	5/5

Table 14. Success probability of improved parallel key recovery (the average of 7*100 trials of $S_{h+1}^{[31,28]}, S_{h+1}^{[28,25]}, \cdots, S_{h+1}^{[8,5]}$ (100 trials an interval))

4 half-rounds		6 half-rounds		8 half-rounds	
#texts	#keys	#texts	#keys	#texts	#keys
2^{13}	30.1	2^{19}	18.1	2^{25}	20.7
2^{14}	59.1	2^{20}	40.7	2^{26}	42.4
2^{15}	88.6	2^{21}	77.3	2^{27}	71.6
2^{16}	95.7	2^{22}	93.1	2^{28}	90.9
2^{17}	97.1	2^{23}	97.1	2^{29}	95.3