Optimized χ^2-Attack against RC6

Norihisa Isogai*, Takashi Matsunaka, and Atsuko Miyaji

Japan Advanced Institute of Science and Technology.
{isogai, t-matsuna, miyaji }@jaist.ac.jp

Abstract. In this paper, we make progress on χ^2-attack by introducing the optimization. We propose three key recovery attacks against RC6 without post-whitening, and apply these three key recovery algorithms to RC6. We discuss their differences and optimization and thus our best attack can break 16-round RC6 without pre-whitening with 128-bit key (resp. 16-round RC6 with 192-bit key) by using $2^{117.84}$ (resp. $2^{122.84}$) chosen plaintexts with a success probability of 95% (resp. 90%). As far as the authors know, this is the best result of attacks to RC6.

Keywords: Block Cipher, Cryptanalysis, RC6, χ^2-attack

1 Introduction

RC6 [11] is a block cipher designed by Rivest *et al*. in 1998. RC6-$w/r/b$ means that four w-bit-word plaintexts are encrypted with r rounds by b-byte keys. Currently, RC6-32/20 is recommended to give sufficient resistance against the known attacks [1,2,3,5,7,9,12,13]. In this paper, RC6-32 is simply denoted by RC6. RC6 operates as an unit of w-bit word using five basic operations such as an addition, a subtraction, a bitwise exclusive-or, a multiplication, and a data dependent rotation. Therefore, this block cipher has a wonderful capability for performing high-speed software implementation especially on Intel processors. Up to the present, linear attacks, differential attacks, and χ^2-attacks against RC6 and some simplified variants of RC6 have been analyzed intensively. Table 1 summarizes the previous results on RC6. In [2], the security of RC6 against the differential and linear cryptanalysis was given. They estimated that 12-round RC6 is not secure against the differential cryptanalysis. As for linear cryptanalysis using multiple approximations and linear hulls, it was reported that RC6 with 16 or more rounds is secure. As a result, they concluded that 20-round RC6 is secure against differential and linear cryptanalysis. In [12], on the other hand, a correct key of 14-round RC6 with 256-bit key can be recovered by using multiple linear attack, and a weak key of 18-round RC6 can be recovered with the probability of about $1/2^{90}$.

The χ^2-attack is one of the most effective attacks on RC6. The χ^2-attack was originally proposed by Vaudenay as an attack on the Data Encryption Standard

* The author is currently with Matsushita Information System Research Laboratory Nagoya Co., LTD.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 16–32, 2003.

(DES) [14], and Handschuh *et al*. applied that to SEAL [6]. In [5,7,9], the χ^2-attacks were applied to RC6 or a simplified variant of RC6. The χ^2-attack can be used for both distinguishing attacks and key recovery attacks. Distinguishing attacks handle plaintexts in such a way that the χ^2-value of a part of ciphertexts becomes significantly a higher value. Key recovery attacks have to rule out all wrong keys, single out exactly a correct key by using the χ^2-value, and thus they often require more work and memory than distinguishing attacks. In [5,7], they just focused on such plaintexts that outputs high χ^2-value on ciphertext, and in [9], they made progress by introducing a notion of *variance* as well as χ^2-value itself. But, unfortunately, optimization of χ^2-value has never been discussed, that is, what level of variance is optimal.

In this paper, we propose three key recovery attacks against RC6 without post-whitening and discuss the differences and optimization. We also apply the key recovery attacks to RC6 and demonstrate one of them on RC6-8. Our key recovery attack itself gives a remarkable impact on RC6: our best attack can break 16-round RC6 without pre-whitening with 128-bit key (resp. 16-round RC6 with 192-bit key) by using $2^{117.84}$ (resp. $2^{122.84}$) chosen plaintexts with a success probability of 95% (resp. 90%).

This paper is organized as follows. Section 2 summarizes the notation, RC6 algorithm, and the χ^2-test. Section 3 investigates the χ^2-statistic of RC6. Section 4 presents three key recovery attacks against RC6 without post-whitening, Algorithms 2, 3, and 4. We evaluate the security against RC6 in Section 5. Conclusion is given in Section 6.

Table 1. Attacks on RC6

Attack	Target RC6	Rounds	#Texts
Linear Attack [2]	RC6	16	2^{119}
χ^2 Attack [7]	RC6 with 256-bit key	15	2^{119}
Multiple Linear Attack [12]	RC6 with 192-bit key	14[1]	$2^{119.68}$
χ^2 Attack [9]	RC6W[2] with 128-bit key	17	$2^{123.9}$
Our result	RC6P[3] with 128-bit key	16	$2^{117.84}$
	RC6 with 192-bit key	16	$2^{122.84}$

1: A weak key of 18-round RC6 with 256-bit key can be recovered by $2^{126.936}$ plaintexts with the probability of about $1/2^{90}$.
2: RC6W means RC6 without pre- or post-whitening.
3: RC6P means RC6 without post-whitening.

2 Preliminary

We summarize the notation, RC6 algorithm, and the χ^2-test, used in this paper.

2.1 Notation

Let us use the following notation:

$+$: addition;
$-$: subtraction;

$\oplus$: bitwise exclusive-or;
r : number of rounds;
$a \lll b$: cyclic rotation of a to the left by b-bit;
$a \ggg b$: cyclic rotation of a to the right by b-bit;
(A_i, B_i, C_i, D_i) : input of the i-th round; (A_0, B_0, C_0, D_0) : plaintext;
$(A_{r+2}, B_{r+2}, C_{r+2}, D_{r+2})$: ciphertext after r-round encryption;
S_i : i-th subkey;
$\mathrm{lsb}_n(X)$: least significant n-bit of X;
$\mathrm{msb}_n(X)$: most significant n-bit of X;
X^i : i-th bit of X;
$f(x)$: $x \times (2x + 1)$;
$F(x)$: $f(x) \pmod{2^{32}} \lll 5$;
$x||y$: concatenated value of x and y.

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.

2.2 Block Cipher RC6

Algorithm 1 (RC6 Encryption Algorithm)

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

Parts 1 and 3 of Algorithm 1 are called pre-whitening and post-whitening, respectively. We call the version of RC6 without post-whitening to, simply, RC6P.

2.3 χ^2-Test

We make use of the χ^2-tests for distinguishing a non-uniformly random distribution from uniformly random distribution [7]. 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:

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

Table 2 presents each threshold for 63 degrees of freedom. For example, (level, χ^2) = (0.95, 82.53) for 63 degrees of freedom in Table 2 means that the value of χ^2-statistic exceeds 82.53 in the probability of 5% if the observation X is uniform.

3 χ^2-Statistic of RC6

We improve the distinguishing attacks in such a way that the χ^2-values become significantly high and that the available number of plaintexts is not reduced.

Table 2. Selected threshold values of χ^2-distribution with 63 degrees of freedom

Level	0.50	0.60	0.70	0.80	0.90	0.95	0.99
63 degrees of freedom	62.33	65.20	68.37	72.20	77.75	82.53	92.01

3.1 Overview of Experiments

The previous results [7,9] of χ^2-statistic are summarized as follows: 1. 10-bit outputs of $\mathrm{lsb}_5(A_{r+1})||\mathrm{lsb}_5(C_{r+1})$ lead to much stronger biases if both $\mathrm{lsb}_5(A_0)$ and $\mathrm{lsb}_5(C_0)$ are fixed and both B_0 and D_0 introduce zero rotation in the 1st round;
2. 10-bit outputs of $\mathrm{lsb}_5(A_{r+1})||\mathrm{lsb}_5(C_{r+1})$ lead to much stronger biases if $\mathrm{lsb}_5(A_0)$ is fixed, $\mathrm{lsb}_5(C_0) = 0$, and both B_0 and D_0 introduce zero rotation in the 1st round;
3. $2n$-bit outputs ($n = 3, 4, 5$) of $\mathrm{lsb}_n(A_{r+1})||\mathrm{lsb}_n(C_{r+1})$ lead to much stronger biases if $\mathrm{lsb}_5(A_0) = 0$, $\mathrm{lsb}_5(C_0) = 0$, and both B_0 and D_0 introduce zero rotation in the 1st round.

In other words, the previous key recovery algorithms make use of the distinguishing algorithms that fix $\mathrm{lsb}_n(A_0)$, $\mathrm{lsb}_n(C_0)$, or both and that introduce zero rotation in the 1st round. However, fixing the 1st-round rotation requires much memory for the key recovery attack and reduces the available number of plaintexts [7]. Here, in order to investigate other conditions that have almost the same effect but that do not reduce the available number of plaintexts, we conduct the following three experiments.

Test 1: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0)$ is set to 0.
Test 2: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0)$ is set to 0.
Test 3: The χ^2-test on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in the case of which $\mathrm{lsb}_4(B_0)||\mathrm{lsb}_4(D_0)$ is set to 0.

Test 1 corresponds to the previous χ^2-test [7,9]. Since we have known in [9] that the χ^2-value of $\mathrm{lsb}_n(A_{r+1})||\mathrm{lsb}_n(C_{r+1})$ ($n = 2, 3, 4$) outputs almost the same bias, we present only the results of $n = 3$ to compare the difference between $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0) = 0$ and $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0) = 0$. Test 2 or 3 fixes $\mathrm{lsb}_5(B_0)||\mathrm{lsb}_5(D_0)$ or $\mathrm{lsb}_4(B_0)||\mathrm{lsb}_4(D_0)$ instead of $\mathrm{lsb}_5(A_0)||\mathrm{lsb}_5(C_0)$, respectively. Our experiments generate all plaintexts by using M-sequence [8]. For example, 118-, 123-, and 128-bit random numbers are generated by M-sequence, whose primitive polynomials of M-sequence are $x^{118} + x^{36} + x^8 + x + 1$, $x^{123} + x^{16} + x^{12} + x + 1$, and $x^{128} + x^7 + x^2 + x + 1$, respectively. Our platforms are IBM RS/6000 SP (PPC 604e/332MHz $\times$ 256) with memory of 32 GB and PC cluster system (Pentium III/1GHz $\times$ 50) with memory of 12.5 GB. All tests use 10^3 keys and 10^2 kinds of plaintexts, and thus conduct 10^5 trials in total.

3.2 Test 1 and Test 2

The results of Tests 1 or 2 are shown in Tables 3 or 4, respectively. These results indicate that Test 1 outputs more bias than Test 2, but that Test 2 also outputs enough bias by using the same number of plaintexts. As reported in [9], we do not necessarily need much bias like level of 0.95 as in [7] to recover a correct key, which will be also shown in the subsequent sections. In fact, the level of more than 0.57 is enough for key recovering. Furthermore if we employ Test 1 to key recovery algorithm, the 1st-round rotation has to be fixed to zero in order to maintain the effect after post-whitening. However it requires extremely much memory. Considering these conditions, we employ Tests 2 and 3 to key recovery algorithm.

3.3 Test 2 and Test 3

Table 5 shows the results of Test 3. Tables 4 and 5 indicate that Test 2 outputs higher χ^2-value with fewer number of plaintexts than Test 3; but that Test 3 also outputs enough high bias.

Suppose that $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ is fixed to some value except $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0) = 0$ ($n = 4, 5$). Then, $\mathrm{lsb}_n(A_2)||\mathrm{lsb}_n(C_2)$, i.e. $(\mathrm{lsb}_n(B_0) + \mathrm{lsb}_n(S_0)) \pmod{2^n}||(\mathrm{lsb}_n(D_0) + \mathrm{lsb}_n(S_1)) \pmod{2^n}$, is fixed in the same way as $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0) = 0$. Namely, whatever value $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ ($n = 5, 4$) in Test 2 or 3 is fixed to, the same result as Table 4 or 5 is expected. Thus, we can generalize Test 2 or 3 to use any plaintext by just classifying it to each $\mathrm{lsb}_n(B_0)$ and $\mathrm{lsb}_n(D_0)$, and thus the number of available plaintexts in each Test is 2^{128}.

There is each naturally-extended key recovery attack that makes use of Test 2 or 3 as χ^2-test. In the next section, we apply Test 2 or 3 to the key reovery algorithm to RC6P, Algorithms 2 and 3, or 4. The number of available plaintexts of Algorithms 2 and 3, or 4 is 2^{118} and 2^{123}, or 2^{128}, respectively. These further differ in the number of *classifications*, which has an influence on the memory size or variance of key recovery attacks. *Classification* means the number of groups, in which plaintexts are classified and the average of χ^2-value is computed. In the subsequent sections, we will see how these differences work on each corresponding key recovery attack.

3.4 Slope

To extend our discussion on lower rounds to that on higher rounds, we estimate the slope of Tests 2 and 3 as in [7], that is, how many plaintexts are required to get similar values in a χ^2-test on $r + 2$ rounds compared with r rounds. Table 6 shows the number of plaintexts required for the χ^2-values with each level of 0.55, 0.60, 0.65, and 0.70, and estimates that each average slope of Test 2 (resp. Test 3) is $2^{16.01}$ (resp. $2^{16.03}$). Both Tests output almost the same slope, but Test 2 outputs slightly smaller slope than Test 3. This is because Test 2 fixes more bits of input than that of Test 3. In our estimation, we take each largest value $2^{16.04}$

or $2^{16.06}$ as each slope of Test 2 or 3 to make our estimation strict, respectively. In the following sections, we will show Algorithms 2 and 3 to RC6P, Algorithms 5 and 6 to RC6 (resp. Algorithm 4 to RC6P, Algorithm 7 to RC6), which are based on Test 2 (resp. Test 3). Each algorithm conducts the same χ^2-test as that of each corresponding Test. Therefore, to extend our discussion on lower rounds to that on higher rounds, we use the slope of each corresponding Test.

Table 7 shows the efficiency of each Test from the point of view of distinguishing attack. Considering the number of available plaintexts of Test 2 (resp. Test 3), 2^{118} (resp. 2^{120}), Test 2 (resp. Test 3) can distinguish output of 15-round RC6 from a randomly chosen permutation by using $2^{112.0}$ plaintexts (resp. $2^{112.90}$ plaintexts). Test 2 can work better than Test 3 from the point of view of distinguishing attack as we noted the above. In the subsequent sections, we will show some key recovery algorithms based on Test 2 or 3 that differ each other in the number of classifications.

Table 3. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 1 (the average of 10^5 trials)

2 rounds				4 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^8	63.402	0.538	126.731	2^{24}	63.489	0.541	127.840
2^9	63.875	0.554	129.299	2^{25}	64.028	0.560	129.847
2^{10}	64.729	0.584	133.864	2^{26}	65.006	0.593	134.789
2^{11}	66.415	0.640	142.293	2^{27}	67.052	0.660	144.714
2^{12}	69.939	0.744	157.668	2^{28}	71.000	0.771	167.825

Table 4. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 2 (the average of 10^5 trials)

3 rounds				5 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^8	63.224	0.532	125.883	2^{24}	63.262	0.533	126.990
2^9	63.416	0.538	126.119	2^{25}	63.429	0.539	127.497
2^{10}	63.819	0.553	129.069	2^{26}	63.790	0.552	128.212
2^{11}	64.669	0.582	132.916	2^{27}	64.521	0.578	131.408
2^{12}	66.352	0.638	140.551	2^{28}	66.373	0.639	140.554

4 Cryptanalysis against RC6 without Post-whitening

We present three key recovery algorithms against RC6P, and discuss their differences and the optimal condition to attack to RC6P. The main idea of these algorithms follow [9], but we fix some bits out of $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$ instead of

Table 5. The χ^2-value on $\mathrm{lsb}_3(A_{r+1})||\mathrm{lsb}_3(C_{r+1})$ in Test 3 (the average of 10^5 trials)

3 rounds				5 rounds			
# Texts	χ^2-value	Level	Variance	# Texts	χ^2-value	Level	Variance
2^9	63.166	0.530	125.506	2^{25}	63.251	0.533	128.115
2^{10}	63.449	0.540	127.468	2^{26}	63.450	0.540	127.756
2^{11}	63.878	0.555	128.891	2^{27}	63.849	0.554	130.461
2^{12}	64.865	0.589	132.279	2^{28}	64.800	0.586	132.642
2^{13}	66.778	0.651	141.879	2^{29}	66.744	0.650	141.138

Table 6. $\log_2(\#\text{texts})$ and the slope required for the χ^2-value of each level (the average of 10^5 trials)

	Test 2			Test 3		
Level	3 rounds	5 rounds	slope	3 rounds	5 rounds	slope
0.55	9.92	25.89	15.97	10.80	26.83	16.03
0.60	11.45	27.49	16.04	12.26	28.32	16.06
0.65	12.17	28.17	16.00	13.00	29.00	16.00
0.70	12.71	28.72	16.01	13.53	29.57	16.04
average			16.01			16.03

Table 7. $\log_2(\#\text{texts})$ and linear equations for Tests 2 and 3 (the average of 10^5 trials)

	Test 2			Test 3		
Level	3 rounds	5 rounds	linear equation	3 rounds	5 rounds	linear equation
0.99	15.7	31.8	$8.02r - 8.30$	16.6	32.6	$8.03r - 7.55$

$\mathrm{lsb}_n(A_0)||\mathrm{lsb}_n(C_0)$ or the first-round-rotation amount. Intuitively, our algorithms fix some bits out of $\mathrm{lsb}_n(B_0)||\mathrm{lsb}_n(D_0)$, check the χ^2-value of $\mathrm{lsb}_3(A_r)||\mathrm{lsb}_3(C_r)$, and recover both $\mathrm{lsb}_2(S_{2r})$ and $\mathrm{lsb}_2(S_{2r+1})$ of r-round RC6P. Here we set $(y_b, y_d) = (\mathrm{lsb}_3(B_{r+1}), \mathrm{lsb}_3(D_{r+1}))$, $(x_c, x_a) = (\mathrm{lsb}_5(F(A_{r+1})), \mathrm{lsb}_5(F(C_{r+1})))$, $(s_a, s_c) = (\mathrm{lsb}_2(S_{2r}), \mathrm{lsb}_2(S_{2r+1}))$, $s = s_a||s_c$, and $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$, where x_a (resp. x_c) is the rotation amounts on A_r (resp. C_r) in the r-th round.

4.1 Key Recovery Algorithms Based on Test 2

Algorithm 2 and 3 are based on Test 2 in Section 3. Algorithm 2 averages the χ^2-value among 2^{10} classifications, while Algorithm 3 averages it among 2^{15} classifications.

Algorithm 2

1. `Choose a plaintext` (A_0, B_0, C_0, D_0) `with` $(\mathtt{lsb}_5(B_0), \mathtt{lsb}_5(D_0)) = (0, 0)$ `and encrypt it.`
2. `For each` (s_a, s_c)`, decrypt` $y_d||y_b$ `with a key` $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ `by 1 round`[1]`.`

[1] Since any (S^3_{2r}, S^3_{2r+1}) outputs the same χ^2-value of z [9], we may decrypt y by setting $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$.

The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, x_a, x_c, and z, update each array by incrementing count$[s][x_a][x_c][z]$.
4. For each s, x_a, and x_c, compute $\chi^2[s][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][x_a][x_c]\}_{x_a,x_c}$ for each s and output s with the highest $ave[s]$ as $\mathtt{lsb}_2(S_{2r})||\mathtt{lsb}_2(S_{2r+1})$.

Algorithm 3

1. Choose a plaintext (A_0, B_0, C_0, D_0) with $\mathtt{lsb}_5(D_0) = 0$, set $t = \mathtt{lsb}_5(B_0)$, and encrypt it.
2. For each (s_a, s_c), decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are also denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, t, x_a, x_c, and z, update each array by incrementing count$[s][t][x_a][x_c][z]$.
4. For each s, t, x_a, and x_c, compute $\chi^2[s][t][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t][x_a][x_c]\}_{x_a,x_c,t}$ for each s and output s with the highest $ave[s]$ as $\mathtt{lsb}_2(S_{2r})||\mathtt{lsb}_2(S_{2r+1})$.

Table 8 shows the results of Algorithms 2 and 3 on 4-round RC6P: SUC, the average of χ^2-values $ave[s]$ on recovered keys, the level, and the variance, where SUC is the success probability among 1000 keys. Before comparing the results of Algorithms 2 and 3 (Table 8) with that of Test 2 (Table 4), we may review the fact of distribution of the mean [4], that is, for the mean μ or the variance σ^2 of a population, the mean or the variance of the distribution of the mean of a random sample with the size n drawn from the population are μ or σ^2/n, respectively. Plaintexts in Algorithm 2 or 3 are classified into 2^{10} or 2^{15} groups of $\{x_a, x_c\}$ or $\{\mathtt{lsb}_5(B_0), x_a, x_c\}$ and $ave[s]$ is computed over each group. On the other hand, all plaintexts are uniformly distributed to each group since they are randomly generated by M-sequences in our experiments. Therefore, the χ^2-value $ave[s]$ in Algorithm 2 or 3 is computed by using $1/2^{10}$ or $1/2^{15}$ times the number of plaintexts in Table 8. Applying this discussion to the experimental results, we see that the above fact of distribution of the mean exactly holds in Algorithms 2 and 3: the average of χ^2-value on $2^{18} - 2^{22}$ or $2^{23} - 2^{25}$ plaintexts in Algorithm 2 or 3 corresponds to that of $2^8 - 2^{12}$ or $2^8 - 2^{10}$ plaintexts in the case of $r = 3$ of Table 4; the variance of χ^2-values in Algorithm 2 or 3 corresponds to about $1/2^{10}$ or $1/2^{15}$ as much as that of Table 4; the averages of χ^2-values by using $2^{23} - 2^{25}$ plaintexts in Algorithm 3 are roughly equal to those by using $2^{18} - 2^{20}$ plaintexts in Algorithm 2; and the variances of χ^2-values by using $2^{23} - 2^{25}$ plaintexts in Algorithm 3 are about $1/2^5$ as much as those by using $2^{18} - 2^{20}$ plaintexts in Algorithm 2. We also remark that the level of χ^2-value more than 0.57 or 0.53 is enough for key recovering in Algorithm 2 or 3, respectively.

Let us discuss the security in higher rounds. Since Algorithms 2 and 3 are based on the χ^2-test of Test 2, we may expect that the slope in Test 2 holds in Algorithms 2 and 3. By using detailed experimental results in Table 9 and the

slope in Test 2, the number of plaintexts required for recovering a key in r-round RC6P with the success probability of 95%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = \begin{cases} 8.02r - 10.48 \text{ (Algorithm 2)} \\ 8.02r - 7.98. \text{ (Algorithm 3).} \end{cases}$$

Let us investigate the amount of work by setting one unit of work to one encryption. Algorithms 2 and 3 encrypts each plaintext and decrypts a ciphertext by 1 round with each key candidate. Therefore, the amount of work is $\#texts \times (1 + 1/r \times 2^4)$. Thus, by substituting the number of available plaintexts 2^{118} or 2^{123}, Algorithm 2 or 3 can break 16-round RC6P by using $2^{117.84}$ or $2^{120.34}$ plaintexts, $2^{118.84}$ or $2^{121.34}$ work, and 2^{20} or 2^{25} memory with a probability of 95%, respectively.

Table 8. The average of χ^2-value and the variance in Algorithms 2, 3, and 4 on 4-round RC6P (in 1000 trials)

Algorithm 2					Algorithm 3				
#texts	SUC	χ^2-value	Level	Variance	#texts	SUC	χ^2-value	Level	Variance
2^{18}	0.097	63.122	0.5280	0.1241	2^{21}	0.122	63.102	0.5273	0.0020
2^{19}	0.155	63.261	0.5329	0.1260	2^{22}	0.247	63.114	0.5278	0.0022
2^{20}	0.344	63.534	0.5425	0.1289	2^{23}	0.526	63.157	0.5293	0.0026
2^{21}	0.744	64.096	0.5621	0.1278	2^{24}	0.938	63.278	0.5336	0.0038
2^{22}	0.995	65.187	0.5994	0.1316	2^{25}	1.000	63.561	0.5435	0.0044

Algorithm 4				
#texts	SUC	χ^2-value	Level	Variance
2^{23}	0.117	63.011	0.5241	0.0003
2^{24}	0.177	63.020	0.5244	0.0004
2^{25}	0.347	63.037	0.5250	0.0004
2^{26}	0.768	63.067	0.5261	0.0005
2^{27}	1.000	63.139	0.5286	0.0005

Table 9. $\log_2(\#texts)$ required for key recovering of 4-round RC6P with each success probability (in 1000 trials)

Success	Algorithm 2			Algorithm 3			Algorithm 4		
Probability	#texts	χ^2-value	Level	#texts	χ^2-value	Level	#texts	χ^2-value	Level
95%	$2^{21.6}$	64.539	0.5778	$2^{24.1}$	63.295	0.5341	$2^{26.6}$	63.102	0.5273
50%	$2^{20.4}$	63.721	0.5507	$2^{23.0}$	63.157	0.5293	$2^{25.4}$	63.045	0.5253

4.2 Key Recovery Algorithm Based on Test 3

Algorithm 4 is based on the χ^2-test of Test 3 in Section 3 and averages it among 2^{18} classifications.

Algorithm 4

1. Choose a plaintext (A_0, B_0, C_0, D_0), set $(t_b, t_d) = (\mathrm{lsb}_4(B_0), \mathrm{lsb}_4(D_0))$, and encrypt it.
2. For each (s_a, s_c), decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, which are also denoted by a 6-bit integer $z = z_a||z_c$.
3. For each value s, t_b, t_d, x_a, x_c, and z, update each array by incrementing $\mathrm{count}[s][t_b][t_d][x_a][x_c][z]$.
4. For each s, t_b, t_d, x_a, and x_c, compute $\chi^2[s][t_b][t_d][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][t_b][t_d][x_a][x_c]\}_{t_b,t_d,x_a,x_c}$ for each s, and output s with the highest $ave[s]$ as $\mathrm{lsb}_2(S_{2r})||\mathrm{lsb}_2(S_{2r+1})$.

Table 8 shows the results of Algorithm 4. Algorithm 4 classifies plaintexts into 2^{18} groups of $\{\mathrm{lsb}_4(B_0), \mathrm{lsb}_4(D_0), x_a, x_c\}$ and averages χ^2-value over each group. In the same discussion as Algorithms 2 and 3, we see that the average of χ^2-values by using 2^{27} plaintexts in Table 8 is roughly equal to that by using 2^9 plaintexts in the case of $r = 3$ of Table 5; and the variance of χ^2-values by using 2^{27} plaintexts in Table 8 is about $1/2^{18}$ as much as that by using 2^9 plaintexts in the case of $r = 3$ of Table 5. We note that the χ^2-value level of more than 0.527 is enough for key recovering in Algorithm 4.

Let us discuss the security in higher rounds. In the same discussion as Algorithms 2 and 3, we apply the slope of Test 3 in that of Algorithm 4. By using more detailed experimental results in Table 9 and the slope of Test 3, the number of plaintexts required for recovering a key in r-round RC6P with the success probability of 95%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.03r - 5.52.$$

By substituting the number of available plaintexts 2^{128}, Algorithm 4 can break 16-round RC6P by using $2^{122.96}$ plaintexts, $2^{123.96}$ work, and 2^{28} memory with a probability of 95%.

4.3 Comparison of Algorithms 2, 3, and 4

Algorithms 2, 3, and 4 differ mainly in the number of classifications. In other words, they differ in the number of plaintexts that the χ^2-values are averaged. We investigate how such a difference influences on a key recovery algorithm. Table 10 summarizes results of three algorithms: the applicable rounds and the efficiency. Algorithm 4 can break 16-round RC6P in the success probability of 95% with the lowest level of χ^2-value, at most 0.528. Because, the more the number of classifications is, the smaller the variance of χ^2-value are, as we reviewed the fact above. The smaller variance is one of necessary factors to single out a correct key as in [9]. However, in contrast to [9], Algorithm 4 is not the most efficient attack of three algorithms. Three algorithms can analyze RC6P with the same number of rounds. That is, it does not necessarily holds that the more number of classifications, the larger applicable rounds. Generally, the larger the number of classifications, the lower level of χ^2-value are required to recover a correct key but the more necessary plaintexts and work are required. On the other hand,

there exists an upper limit of the available plaintexts and work amount. This is why the optimization of the number of classifications is necessary.

There are two factors of the number of both available texts and classifications to discuss the optimization. Fixing the number of available texts to 2^{128}, let us investigate the optimal number of classifications: the χ^2-value is averaged over groups $\{\mathrm{lsb}_3(B_0), \mathrm{lsb}_3(D_0), x_a, x_c\}$, $\{\mathrm{lsb}_4(B_0), \mathrm{lsb}_4(D_0), x_a, x_c\}$, or $\{\mathrm{lsb}_5(B_0), \mathrm{lsb}_5(D_0), x_a, x_c\}$, namely the number of classifications is 2^{16}, 2^{18}, or 2^{20}, respectively. This means that we optimize Algorithm 4 by changing the number of classification. Table 11 shows the results, which indicates that the key recovery attack with 2^{18} classifications, i.e. Algorithm 4, is the optimal. The number of classifications of Algorithms 2 and 3 is also optimized to attack RC6 well.

Table 10. Comparison of Algorithms 2, 3, and 4 on RC6P: applicable rounds and the efficiency

Algorithm	#classifications	#available texts	nemory	rounds	#texts	work	χ^2-value (level)	variance
2	2^{10}	2^{118}	2^{20}	16	$2^{117.84}$	$2^{118.84}$	64.539 (0.578)	0.1319
3	2^{15}	2^{123}	2^{25}	16	$2^{120.34}$	$2^{121.34}$	63.295 (0.535)	0.0039
4	2^{18}	2^{128}	2^{28}	16	$2^{122.96}$	$2^{123.96}$	63.102 (0.528)	0.0005

Table 11. #texts necessary for variations of Algorithm 4 on 4-round RC6P

	#classifications		
	2^{16}	2^{18}	2^{20}
#texts (95%)	$2^{27.0}$	$2^{26.6}$	$2^{26.9}$
#texts (50%)	$2^{25.8}$	$2^{25.4}$	$2^{25.7}$

5 Cryptanalysis against RC6

In this section, we apply Algorithm 2, 3, or 4 to RC6 with a 24-byte key, which is called Algorithm 5, 6, or 7, respectively. They recover a 68-bit subkey of $\mathrm{lsb}_2(S_{2r})$, $\mathrm{lsb}_2(S_{2r+1})$, S_{2r+2}, and S_{2r+3}. We demonstrate Algorithm 5 to RC6-8 and discuss how to analyze the security to RC6 with a 24-byte key.

5.1 Attacks on RC6

Let us set $(y_b, y_d) = (\mathrm{lsb}_3(B_{r+2}), \mathrm{lsb}_3(D_{r+2}))$, $(s_a, s_c) = (\mathrm{lsb}_2(S_{2r}), \mathrm{lsb}_2(S_{2r+1}))$, $s = s_a||s_c||S_{2r+2}||S_{2r+3}$, $(S^3_{2r}, S^3_{2r+1}) = (0, 0)$, and $(x_c, x_a) = (\mathrm{lsb}_5(F(A_{r+2} - S_{2r+2})), \mathrm{lsb}_5(F(C_{r+2} - S_{2r+3})))$, where x_a or x_c is the r-round rotation amounts on A_r or C_r, respectively.

Algorithm 5

1. Choose a plaintext (A_0, B_0, C_0, D_0) with $(\mathtt{lsb}_5(B_0), \mathtt{lsb}_5(D_0)) = (0, 0)$ and encrypt it.
2. For each subkey S_{2r+2} and S_{2r+3}, decrypt $y_d||y_b$ with a key $(S^3_{2r}||s_a, S^3_{2r+1}||s_c)$ by 1 round. The decryptions of y_d and y_b are set to z_a and z_c, respectively, which are denoted as a 6-bit integer $z = z_a||z_c$.
3. For each value s, x_a, x_c, and z, update each array by incrementing $\mathtt{count}[s][x_a][x_c][z]$.
4. For each s, x_a, and x_c, compute $\chi^2[s][x_a][x_c]$.
5. Compute the average $ave[s]$ of $\{\chi^2[s][x_a][x_c]\}_{x_a,x_c}$ for each s, and output s with the highest $ave[s]$ as $\mathtt{lsb}_2(S_{2r})||\mathtt{lsb}_2(S_{2r+1})||S_{2r+2}||S_{2r+3}$.

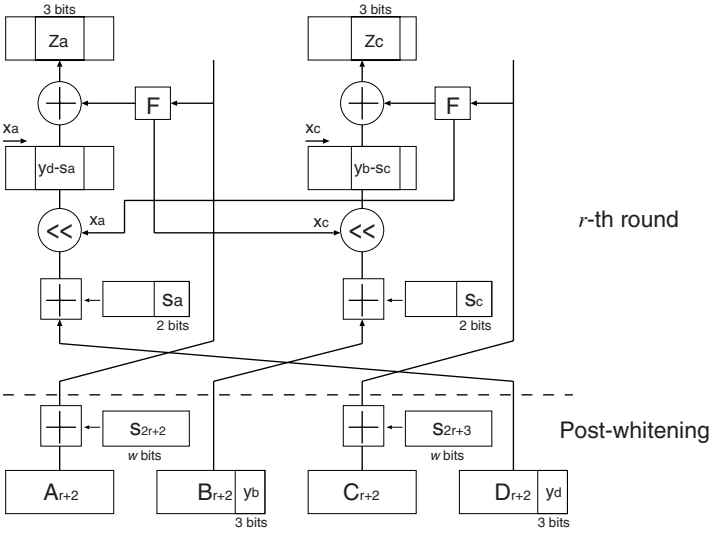

Fig. 1. Outline of Algorithm 5

Figure 1 shows the outline of Algorithm 5. Algorithm 5 differ with Algorithm 2 in a way of handling both S_{2r+2} and S_{2r+3}: Algorithm 2 uses a correct key on S_{2r+2} and S_{2r+3}; but Algorithm 5 has to guess a correct key of S_{2r+2} and S_{2r+3}. Therefore, the results of Algorithm 5 against r-round RC6 is coincident with those of Algorithm 2 against r-round RC6P whenever correct keys on S_{2r+2} and S_{2r+3} are used. As a result, to discuss the security on RC6 against Algorithm 5, we have only to investigate the behavior of χ^2-value with using wrong-keys of S_{2r+2} and S_{2r+3}.

5.2 Differences between Algorithms 2 and 5

To investigate the difference between two algorithms, let us observe how wrong-keys of S_{2r+2} have an influence on a key recovery in Algorithm 5 when a correct key is set to S_{2r+3}. Table 12 shows the experimental results of Algorithm 2 on RC6P-8 or Algorithm 5 on RC6-8, in which Algorithm 2 recovers 4-bit subkeys of $\mathrm{lsb}_2(S_8)$ and $\mathrm{lsb}_2(S_9)$; and Algorithm 5 recovers 12-bit subkeys of $\mathrm{lsb}_2(S_8)$, $\mathrm{lsb}_2(S_9)$, and S_{10}. Table 12 indicates that: Algorithm 5 cannot work as effectively as Algorithm 2 if a few plaintexts like 2^{11} or 2^{12} are used; but Algorithm 5 can work as effectively as Algorithm 2 if enough many plaintexts like 2^{14} or 2^{15} are used. They differ in the number of wrong keys: the number of wrong keys of Algorithm 5 is 2^8 times as many as that of Algorithm 2. If a few (i.e. not enough) plaintexts are used, then the χ^2-value on even a correct key is rather low and thus the χ^2-value on wrong keys disturbs us to single out a correct key. As a result, the difference in the number of wrong keys influences the probability that can single out a correct key. On the other hand, if *enough* number of plaintexts are used, then the χ^2-value on a correct key becomes enough high, while that on wrong keys does not become high, and, thus, the difference in the number of wrong keys does not have a great influence on singling out a correct key. As a result, Algorithm 5 can single out a correct key with almost the same high probability like more than 90% as Algorithm 2 if *enough* number of plaintexts are used. The remaining problem is how to define *enough* number of plaintexts. We may note that the key recovery attacks compute the χ^2-value on a part for every key candidate and output a key with the highest χ^2-value as a correct key. This means that an algorithm can single out a correct key if and only if a correct key outputs higher χ^2-value than that on all wrong keys. In other words, the lowest χ^2-value on correct keys has only to be higher than the highest χ^2-value on wrong keys. Thus, *enough* number of plaintexts necessary to single out a correct key is defined as the number of plaintexts that makes the lowest χ^2-value on correct keys higher than the highest χ^2-value on wrong keys.

As the final step, we investigate a good sample on wrong keys of S_a, S_c, S_{2r+2} and S_{2r+3} that may output the the highest χ^2-value. Let us set the *almost-correct wrong key* that differs a correct key in only the most-significant-one bit of S_{2r+2}: the other bits, in other words, S_a, S_c, S_{2r+3} and $\mathrm{lsb}_7(S_{2r+2})$, are the same as a correct key. Apparently, this is the most similar to a correct key and is expected to output the highest χ^2-value of wrong keys. Thus, we define enough number of plaintexts to single out a correct key as the number of plaintexts such that the lowest χ^2-value on correct keys becomes higher than the highest χ^2-value on almost-correct wrong keys. To find out enough number of plaintexts in the case of Algorithm 5 on RC6-8, we conduct the following two experiments:

- **Test 4**[2]**: [Behavior of** $\chi^2 - value$ **of correct keys]**
  Compute the highest and lowest χ^2-value on correct keys.
- **Test 5: [Behavior of** $\chi^2 - value$ **of almost-correct wrong keys]**
  Compute the highest χ^2-value on almost-correct wrong keys.

[2] Test 4 is the same as the results of correct keys in Algorithm 2 to RC6P.

The results are shown in Table 13, where SUC means the success probability to recover a correct key of S_a and S_b in Algorithm 2 to RC6P-8. From Table 13, we see that enough number of plaintexts is defined as $2^{14.5}$ plaintexts. Comparing with Table 12, we convince that enough number is well difined and, thus, we estimate that Algorithm 5 can recover a correct key with the success probability of about 90% by using $2^{14.5}$ plaintexts. Table 13 also indicates that the χ^2-value recovered by almost-correct wrong keys does not become high even if many plaintexts are used. This reflects that: the f-function of RC6 is the nonlinear conversion which depends on all 32-bit inputs; and thus the recovered value does not output high χ^2-value if only the input of f-function differs with a correct input even in 1 bit.

Table 12. Success probability of Algorithm 2 (resp. 5) on 4-round RC6P-8 (resp. RC6-8) (in 1000 trials)

#texts	Algorithm 2	Algorithm 5
2^{11}	0.125	0.001
2^{12}	0.241	0.014
2^{13}	0.486	0.177
2^{14}	0.887	0.886
2^{15}	1.000	1.000

Table 13. Results of Tests 4 and 5 in Algorithm 5 on 4-round RC6-8 (in 1000 trials)

		Test 4		Test 5
#texts	SUC	the highest χ^2	the lowest χ^2	the highest χ^2
$2^{12.0}$	0.232	69.711	60.689	68.634
$2^{12.5}$	0.332	70.435	61.680	67.794
$2^{13.0}$	0.491	72.167	62.226	68.181
$2^{13.5}$	0.699	74.458	63.266	67.450
$2^{14.0}$	0.868	77.427	65.359	68.393
$2^{14.5}$	0.972	81.609	68.971	68.174
$2^{15.0}$	0.999	88.978	70.890	68.211

5.3 The Security of RC6 against Algorithms 5

The previous section have defined enough number of plaintexts and seen that Algorithm 5 can recover a correct key with the success probability of 90% by using enough many plaintexts. We conduct Tests 4 and 5 to Algorithm 5 on 4-round RC6 to find out enough number of plaintexts. The results are shown in Table 14, where SUC means the success probability to recover a correct key of S_a and S_b in Algorithm 2 to RC6P. Table 14 indicates that enough number of plaintexts is set to 2^{22} plaintexts; and it is roughly equal to that which outputs

the success probability of more than 95% in Algorithm 2 on RC6P. Then, we estimate that Algorithm 5 can recover a correct key with the success probability of more than 90% by using 2^{22} plaintexts.

Let us discuss the security in higher-round RC6. We increase the number of plaintexts by up to a factor of 2^2 to analyze the security strictly and use the same slope in Test 2 since Algorithm 5 is based on the χ^2-test of Test 2. Then, the number of plaintexts required for recovering a key in r-round RC6 with the success probability of 90%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.02r - 8.08.$$

By substituting the number of available plaintexts 2^{118}, Algorithm 5 can break 15-round RC6 by using $2^{112.22}$ plaintexts with a probability of about 90%.

5.4 Applying Algorithms 3 to RC6

Algorithm 3 can be applied to RC6 in the same way as Algorithm 2, which is called Algorithms 6. We omit the repetitious detail of the algorithm. To find out enough number of plaintexts, we conduct the same experiments of Tests 4 and 5 to Algorithm 6 on 4-round RC6, whose results are shown in Table 14. Table 14 indicates that enough number of plaintexts to Algorithm 6 on RC6 is $2^{24.6}$ plaintexts.

We increase the number of plaintexts by up to a factor of 2^2 to analyze the security strictly and use the slope of Test 2 in Section 3. Then, the number of plaintexts required for a key recovering in r-round RC6 with the success probability of 90%, $\log_2(\#texts)$, is estimated to

$$\log_2(\#texts) = 8.02r - 5.48.$$

By substituting the number of available plaintexts 2^{123}, Algorithms 6 can break 16-round RC6 with 192-bit key by using $2^{122.84}$ plaintexts with a probability of 90%.

Let us compare Algorithms 5 and 6 from the point of view of the number of plaintexts and the amount of work, where one unit of work is set to one encryption. Both algorithms encrypt each plaintext, and decrypt a ciphertext by 1 round with each key candidate, where the number of key candidates is 2^{68}. Thus, the amount of work is computed by $\#texts \times (1 + 2^{68}/r)$. These results are shown in Table 15. In the final paper, the optimization of each algorithm including the results of Algorithm 7 will be discussed.

6 Conclusion

In this paper, we have discussed the optimization of the number of classification by presenting three key recovery attacks against RC6P, Algorithms 2, 3, and 4. As a result of optimization, Algorithm 2 can break 16-round RC6P by

Table 14. Results of Tests 4 and 5 in Algorithms 5 and 6 on 4-round RC6 (in 1000 trials)

Algorithm 5					Algorithm 6				
		Test 4					Test 5		
#texts	SUC	the highest	the lowest	Test 5	#texts	SUC	the highest	the lowest	Test 5
$2^{20.0}$	0.344	63.534	62.644	62.994	$2^{22.5}$	0.385	63.288	62.916	63.197
$2^{20.5}$	0.539	63.769	62.743	62.979	$2^{23.0}$	0.519	63.338	62.905	63.211
$2^{21.0}$	0.744	64.096	62.865	62.976	$2^{23.5}$	0.752	63.395	62.963	63.199
$2^{21.5}$	0.946	64.540	62.904	62.991	$2^{24.0}$	0.934	63.501	63.013	63.218
$2^{22.0}$	0.995	65.187	63.038	62.978	$2^{24.6}$	1.000	63.648	63.223	63.201

Table 15. #Texts and Work for Algorithms 5 and 6 on r-round RC6 (Estimated)

Algorithm	initial Key	rounds	#texts	work	linear equation (SUC$\geqq$90%)
5	192-bit	15	$2^{112.22}$	$2^{176.32}$	$8.02r - 8.08$
6	192-bit	16	$2^{122.84}$	$2^{186.84}$	$8.02r - 5.48$

using $2^{117.84}$ plaintexts with the success probability of 95%. We have also investigated how to estimate the security of RC6 to these key recovery algorithms by introducing the idea of *enough* number of plaintexts and *almost-correct* wrong key. We have shown that Algorithm 6 is estimated to break 16-round RC6 with 24-byte keys by using $2^{122.84}$ plaintexts with the success probability of 90%.

References

1. J. Borst, B. Preneel, and J. Vandewalle, "Linear Cryptanalysis of RC5 and RC6," Proc. Fast Software Encryption, LNCS 1636, pp.16–30, 1999.
2. S. Contini, R. Rivest, M. Robshaw, and Y. Yin, "The Security of the RC6 Block Cipher. v 1.0," August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
3. S. Contini, R. Rivest, M. Robshaw, and Y. Yin, "Improved analysis of some simplified variants of RC6," Proc. Fast Software Encryption, LNCS 1636, pp.1–15, 1999.
4. R.J. Freund and W.J, Wilson, *Statistical Method*, Academic Press, San Diego, 1993.
5. H. Gilbert, H. Handschuh, A. Joux, and S. Vaudenay, "A Statistical Attack on RC6," Proc. Fast Software Encryption, LNCS 1978, pp.64–74, 2000.
6. H. Handschuh and H. Gilbert, "χ^2 Cryptanalysis of the SEAL Encryption Algorithm," Proc. Fast Software Encryption, LNCS 1267, pp.1–12, 1997.
7. L. Knudsen and W. Meier, "Correlations in RC6 with a reduced number of rounds," Proc. Fast Software Encryption, LNCS 1978, pp.94–108, 2001.
8. A. Menezes, P.C. van Oorschot, and S. Vanstone, Handbook of applied cryptography, CRC Press, Inc., Boca Raton, 1996.
9. A. Miyaji and M. Nonaka, "Cryptanalysis of the Reduced-Round RC6," Proc. ICICS 2002, LNCS 2513 pp.480–494, 2002.
10. R. Rivest, "The RC5 Encryption Algorithm," Proc. Fast Software Encryption, LNCS 1008, pp.86–96, 1995.

11. R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, "The RC6 Block Cipher. v1.1," August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
12. T. Shimoyama, M. Takenaka, and T. Koshiba, "Multiple linear cryptanalysis of a reduced round RC6," Proc. Fast Software Encryption, LNCS 2365, pp.76–88. 2002.
13. T. Shimoyama, K. Takeuchi, and J. Hayakawa, "Correlation Attack to the Block Cipher RC5 and the Simplified Variants of RC6," 3rd AES Candidate Conference, April 2000.
14. S. Vaudenay, "An Experiment on DES Statistical Cryptanalysis," Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, pp.139–147, 1996.