Improved Cryptanalysis of RC5

Alex Biryukov and Eyal Kushilevitz

Applied Mathematics Department,
Computer Science Department,
Technion - Israel Institute of Technology, Haifa, Israel 32000.
Email: {albi,eyalk}@cs.technion.ac.il

Abstract. RC5 is a fast block cipher designed by Ron Rivest in 1994. Since then several attempts of cryptanalysis of this cipher were published. The best previously known attack requires 2^{54} chosen plaintexts in order to derive the full set of 25 subkeys for the 12 round RC5 with 32 bit words. In this paper we show a drastic improvement of these results due to a novel *partial* differential approach. Our attack requires 2^{44} chosen plaintexts. We show that the 64 bit word version of RC5 is also much weaker than it was expected.
Keywords: block ciphers, RC5, differential cryptanalysis.

1 Introduction

RC5 is a fast block cipher designed by Ron Rivest in 1994 [8]. RC5 has an attractively simple and easy to analyze structure. It is also very flexible to changes of parameters. RC5 has adaptable word size w in order to suit processors of different word-lengths, a changeable number of rounds r and a variable-length b cryptographic key (so that the user can choose the level of security appropriate for his application).

The "nominal" choice of the parameters proposed in [8] is: 32 bit words, 12 rounds and a 16 byte key. This version of RC5 is referred to as: RC5-32/12/16. Another version with 64 bit words and 16 rounds was suggested for future 64 bit architectures (RC5-64/16/16). The main feature of the cipher is intensive use of data dependent rotations.

Kaliski and Yin [3] evaluated the strength of the RC5 algorithm with respect to differential [1] and linear [6] attacks. They found a linear attack on RC5 with 6 rounds that uses 2^{57} known plaintexts and whose plaintext requirement is impractical after 6 rounds[1]. Their differential attack on RC5-32/12/16 uses 2^{63} chosen plaintexts. An improvement of this attack by a factor of up to 512 was given by Knudsen and Meier [4]. Their idea was to find plaintexts so that there are no rotations in the first few half-rounds. Once these plaintexts have been identified the differential attack of Kaliski and Yin [3] can be performed with differentials of higher probability. The attack in [4] uses 2^{54} chosen plaintexts

[1] As of today no known-plaintext attack on RC5, even with reduced number of rounds exists due to recent result [9] which found gaps in the linear attack on RC5 described in [3].

on RC5-32/12/16. A survey of the current state of knowledge on RC5 appears in [10].

In this paper we demonstrate the effectiveness of *partial* approach to differential cryptanalysis on an example of RC5. We study more complex differentials than in previous works and define a more general notion of a "good pair" with respect to data dependent rotations. Our differentials are *partial* in the sense, that only $\lg w$ least significant bits of the output difference are relevant to our analysis (see also truncated differentials in [5]). Thus we analyze all pairs that escape differences in rotation amounts, and not only pairs that follow specific, easy to analyze, patterns as in previous attacks. We introduce a new notion — "space oracle", which helps to show that data complexity of cryptanalysis of RC5 is bounded by the probability of a good pair and by our capability to detect them. We also develop very efficient filtering techniques, in order to detect good pairs.

RC5 is heavily based on data dependent rotations. We show that the probability of a pair to escape differences in rotations amounts is much higher than it was expected by the designers of RC5. This causes weak avalanche properties and high key dependence of the cipher's output. We start by analyzing a simpler version of RC5, where all additions are replaced by XORs, as a first order approximation to original RC5 (we denote this version as RC5$\oplus$). We successfully attack RC5$\oplus$-32/12/16 with only 2^{28} chosen plaintexts. Based on this result we show an attack which is capable of breaking the standard RC5-32/12/16 with 2^{44} chosen plaintexts. The complexity of the data analysis phase of this attack is negligible and takes less than a minute on a workstation. We successfully attack RC5 up to 10 rounds and experimental results confirm our estimates. We suspect that it may be possible to attack RC5-32/12/16 with only 2^{38} chosen plaintexts by using a tradeoff between data requirements and complexity of analysis. Finally we show that RC5-64/16/16 is also weaker than it was expected.

We conclude that RC5 is not secure against chosen plaintext attacks. Though RC5 has already 12 rounds (24 half-rounds) we suggest to increase this number to at least 16 rounds in order to increase security against differential cryptanalysis.

2 Differential Cryptanalysis

We use a description of RC5 as a so-called Feistel cipher from [3]. Let us denote by (L_0, R_0) the left and right halves of the plaintext, and let S_i be the ith subkey from the expanded key table S generated before encryption. The particular expansion algorithm has no influence on our cryptanalysis. As in all previous attacks, we assume that the subkeys produced by the key schedule are uniformly random. This is a reasonable assumption which helps us concentrate on the properties of the encryption engine itself (it also simplifies the analysis of probabilities). Let w denote the word size which is 32 for RC5-32/12/16. Then the ciphertext (L_{2r+1}, R_{2r+1}) is calculated by the following equations:
$L_1 = L_0 + S_0$
$R_1 = R_0 + S_1$

```
for i = 2 to 2r + 1 do
    L_i = R_{i-1}
    R_i = ((L_{i-1} ⊕ R_{i-1}) <<< R_{i-1}) + S_i,
```

where r – is the number of rounds, $A \lll B$ denotes the rotation of word A by $(B \bmod w)$ positions to the left (if $w = 32$, rotation amount is contained in five least significant bits of B), $+$ denotes addition $\pmod{2^w}$, and $\oplus$ denotes bitwise XOR. The two equations in the body of the loop we call *half-round*. Note that two consecutive half-rounds correspond to one original round of RC5. The two initial equations are called the zeroth half-round.

The idea of differential cryptanalysis is to analyze pairs of plaintexts instead of single plaintexts. An attacker chooses the difference between plaintext pairs and studies propagation (avalanche) of the changes during the encryption.

Definition 1 *The difference between two bit-strings X and X^* of equal length is defined as $X \oplus X^* = \triangle X$, where $\oplus$ is a bitwise XOR operation.*

Definition 2 *We call a pair of plaintexts (P, P^*) a* ***good pair*** *if P and P^* have equal data-dependent rotations in all rounds[2].*

Definition 3 *We call* ***noise*** *all pairs that are suspected to be good pairs (i.e. pass all our criteria for good pairs (filters which are described in the section 3.1)), but which have different rotation amounts in some intermediate rounds of encryption.*

The rotation operation is linear (if individual bits are viewed as variables), and if there is no difference in the rotation amounts we have:

$$(A \lll s) \oplus (B \lll s) = (A \oplus B) \lll s.$$

The $+$ operation is very "close" to linear operation: a difference in one bit remains unchanged with probability 1/2 after the addition and with probability 1/4 turns into a difference in two adjacent bits (if a difference is in the most significant bit, it remains unchanged with probability 1).

Due to a regular iterative structure of RC5 we use the notion of a *difference pattern* (which corresponds to the notion of a characteristics in [1]), for a sequence of round differences, caused by a particular input difference. Each pattern is associated with its probability to appear in the encryption of a plaintext pair. Two difference patterns can be concatenated into a longer pattern if the output difference of the first is equal to the input difference of the second. A pattern is iterative, if it can be concatenated with itself into a longer pattern. We denote by e_i a w-bit vector with one at position i and zeroes at other positions.

---

[2] We studied only pairs that totally avoid differences in rotations amounts. Note that some periodic patterns, like $0101\ldots01$ being rotated by different rotation amounts can preserve the difference as well. Of course S_i on each round spoils the propagation of such patterns, but it might be that some statistical properties still can be observed. We leave the question of whether such patterns can be used to a future research.

Throughout this paper all numbers corresponding to plaintexts or their differences will be hexadecimal, so $e_w = 80\ 00\ 00\ 00$. The ability to contain many patterns with the same input/output differences is captured by the notion of *differential*, defined in [2].

Definition 4 *An r-round* ***differential*** *is a pair $(\triangle P, \triangle C_r)$, where $\triangle P = P \oplus P^*$ is the plaintext difference and $\triangle C_r$ is the output difference at the r-th round. The probability of an r-round differential is the conditional probability that given an input difference $\triangle P$ at the first round, the output difference at the r-th round will be $\triangle C_r$, when the plaintext P and the subkeys S_i are independent and uniformly random.*

Note that the probability of a differential may be much higher than the probability of a particular pattern with the same input and output differences (in the case when there are many other patterns, with the same input/output differences). In Appendix B we provide an example of differential behavior of RC5-32/8/16. For a more detailed description of differential cryptanalysis we refer the reader to [1].

3 Our Differential Cryptanalysis of RC5

In this section we develop our differential attack on RC5. We use a novel approach towards differential cryptanalysis, since RC5 defeats standard approaches. We look at differences where the five least significant bits (corresponding to rotation amount) are equal to zero. Behavior of the other 27 bits of the difference is not restricted. We consider all differentials that lead to good pairs. The probability of these differentials is much higher than that of the one-bit differentials studied in [3, 4] (their differentials are based on propagation of one-bit differences). Our differentials are harder to analyze but a much faster attack can be performed with their help. In order to study the behavior of these complex differentials we start with the "XOR" version of RC5, where all $+$ are changed for $\oplus$'s.

As in the previous works [3, 4] we find the subkey of the last round S_{2r+1} and then perform an attack on RC5 which is shorter by one half-round using the same pool of already collected encryptions.

3.1 RC5 with XOR Instead of ADD

The XOR version of RC5 was shown in [4] to be weak because the parity bit of the plaintext exclusive-or'ed with the parity bit of all subkeys equals the parity bit of the ciphertext. So one plaintext-ciphertext pair provides one bit of information about the subkeys. Still this cipher is a good model, since it shares several main properties with the original RC5 (in particular data dependent rotations). The carry propagation due to $+$ is very slow. We will show, that good pair probabilities of RC5$\oplus$ and RC5 differ approximately by a factor of $2^{\frac{2(r-1)}{3}}$.

Space Oracles In this section we define a new notion that we call a "space oracle". We show that a good pair can be used as a space oracle into the cipher's behavior under specific key. The idea in [4] can be interpreted as using such an oracle. In this paper we construct more efficient oracles for RC5⊕ (being slightly modified they work for original RC5).

Consider a partition of the space of all possible plaintext pairs into n subsets $M_1, M_2, \ldots, M_n$, and suppose we know that one subset M_j is $k * n$ times more probable for a good pair than any other subset M_i. We can easily gain a factor of k in data if we check a fraction of $\frac{1}{nk}$ data from each subset. Even if one subset is "only" n times better than the others, a good pair from this subset will be a "space oracle", which helps to reduce the plaintext space to a subset M_j, where the probability of a good pair is n times higher. More formally[3]:

Definition 5 *Let $M(n) = \{M_1, M_2, \ldots, M_n\}$ be a partition of the space of all plaintext pairs into n disjoint subsets, let P_i be a probability for M_i to contain a good pair ($P_i = Prob(g$ is a good pair $\mid g \in M_i)$), and let $1 \leq j \leq n$. Then a triple $(M(n), j, G)$ is a differential* ***space oracle*** *for partition $M(n)$ with gain $G > 1$, if $\forall i \neq j$, $1 \leq i \leq n$: $P_j > G \cdot P_i$.*

We will also call a good pair that belongs to a particular subset — a space oracle for that subset.

Question 1 *Knowing the structure of an oracle, what is the most efficient way to use it? In other words: Good pairs are distributed very non-uniformly in the space of all possible pairs, so how to locate the high density spot with the smallest number of steps?*

Knudsen and Meier [4] noticed, that knowledge of proper (key dependent) five least significant bits of both halves of the plaintext in effect peels off one full round of RC5 and thus increases the probability of finding differentials for Kaliski and Yin's attack [3]. If we start with a 64-bit plaintext (L_0, R_0), such that $R_0 \equiv S_1 \pmod{32}$ and $L_0 \equiv S_0 \oplus S_2 \pmod{32}$, then we gain first round of RC5 for free, since there is no rotation in the first two half-rounds. Being translated into our oracle terminology: there exists a partition M of the space of all plaintext pairs, such that:

$$M_{(i,j)} = \{(L, R), (L^*, R^*) \in \{0,1\}^{64} \mid L \equiv L^* \text{ (mod 32)} = i, R \equiv R^* \text{ (mod 32)} = j\}.$$

The partition subset used in [4] for differential attack has index (i, j) with $i = S_0 \oplus S_2 \pmod{32}$ and $j = S_1 \pmod{32}$. The gain G is between 2^3 and 2^5 depending on the input difference being used.

Let us describe two oracles of a more complex structure. For example if we start with input difference (e_w, e_w), then we can gain five half-rounds (2.5 full rounds) by the following 10-bit oracle (actually it is a 32+5+10 bit oracle but 37 bits are under our control). Denote $x \equiv R_0 \pmod{32}$, then the result of the

[3] We define oracles for differential chosen plaintext attacks, but it is possible to define oracles for known plaintext attacks as well.

first half-round will be $R_2 = ((L_0 \oplus R_0 \oplus S_0 \oplus S_1) \lll (x \oplus S_1)) \oplus S_2$. Denote $y \equiv R_2 \pmod{32}$. If we fix x and $L_0 \oplus R_0$, then R_2 and thus y is fixed too. If we set now five bits of R_0 at location $w - y$ to be equal to the five lsb bits of $S_3 \oplus ((R_2 \oplus S_1) \lll y)$, we cause no rotation in the third half-round. Thus we gain five half-rounds (plus the zeroth half-round) with probability of $(\frac{w-\lg w}{w})^2$ (again, this can be formulated in our oracle terminology).

The second oracle requires input difference $(0, e_w)$ and helps us to gain three full rounds of RC5$\oplus$. We require no rotation on the first and fourth half-rounds, so that $R_0 \equiv S_1 \pmod{32}$. Then $R_2 = L_0 \oplus R_0 \oplus S_0 \oplus S_1 \oplus S_2$. Denote by $a \equiv R_2 \pmod{32}$, then $R_3 = ((L_0 \oplus S_0 \oplus S_2) \lll a) \oplus S_3$. Denote by $b \equiv R_3 \pmod{32}$, then in order to have no rotation at the fourth round $(R_2 \oplus R_3) \lll b \equiv S_4 \pmod{32}$ must hold. If we require that $R_0 \equiv L_0 \pmod{32}$ then a is constant. If we require that L_0 is fixed we get that all R_3 is fixed, and so is b. Thus knowledge of location b in R_0 at which five constant bits should be placed provides us with six half-rounds with probability $(\frac{w-\lg w}{w})^2$. This is a 15-bit oracle with 32+5 bits under control.

We estimate that the probability to find one good pair bounds the data complexity of differential attack on RC5. This means that the amount of data required in order to detect one good pair, is much larger than the amount of data required to find 10 or even 100 good pairs, after one good pair has been identified. Note that oracles discussed above work for RC5-$w/r/b$ for any value of w.

If one wants to see examples of a space oracles in other block-ciphers, he does not need to go far. "Enhanced characteristic's probability" for DES [7] was noted in [1]. It is shown, that one can gain a factor of four in probability of a pattern and in S/N ratio, knowing three bits of the key. This can be reformulated as a three-bit oracle with gain four in our terminology. In the case of DES, however, oracles are not as helpful as in our case, since differential cryptanalysis of DES uses only one good pair.

Probability of a Good Pair In this section we develop a method to determine a theoretical estimate for good pairs probability for RC5$\oplus$, based on the properties of the underlying Feistel structure.

In [4] it was noticed that Hamming weights of the differences in half-rounds propagate roughly like a Fibonacci sequence $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$[4]. We noticed that in a good pair Hamming weights of the differences behave more like: $0, 1, 1, 0, 1, 1, 2, 3, 5, 8, 13, \ldots$ or $0, 1, 1, 2, 1, 3, 4, 1, 5, 6, 11, \ldots$. Which is a "corrected" Fibonacci sequence with one or two "corrections" respectively. Therefore we define:

Definition 6 *A sequence of numbers $\mathcal{F}(n)$ is called a* ***corrected Fibonacci sequence*** *with k corrections if addition $\mathcal{F}(l) = \mathcal{F}(l-1) + \mathcal{F}(l-2)$ is exchanged for subtraction $\mathcal{F}(l) = \mid \mathcal{F}(l-1) - \mathcal{F}(l-2) \mid$ exactly k times.*

[4] These numbers correspond to the weights of R_i. The last two are impossible as differences since they are bigger than 32, they are present to compare speeds of growth.

Since difference weight sequence in RC5 constitutes a Fibonacci sequence with corrections a question that one may ask is: "Given n and k how many corrected Fibonacci sequences with $\mathcal{F}(i) < W$, $i = 1, \ldots, n$ exist[5]. Correction in the Fibonacci weight sequence occurs at location j if rotation bits of half-round $j-1$ are zero (corresponding to no rotation), thus each correction costs us a factor of 2^{-5} in probability of a pattern (under assumption that all round subkeys are uniformly and independently chosen). If a difference with weight h is rotated, then with probability $\frac{C^h_{w-\lg w}}{C^h_w}$ there will be no difference in the $\lg w$ crucial rotating bits after the rotation. It turns out that in some cases it is worth to pay the price of another correction, instead of pushing forward too heavy differences.

We generated all possible Fibonacci sequences for all "reasonable" numbers of corrections, and calculated the probabilities of patterns, corresponding to such sequences (see Appendix A). We assumed that the starting difference is (e_w, e_w). As a result Table 1 gives the upper bounds for the probability of a good pair for RC5⊕ (with 32 and 64-bit words), which were calculated as a sum of probabilities of all possible corrected Fibonacci sequences of length $n = 2r$ (the last $(2r+1)$st half-round is not relevant to probability calculations). This table

Rounds	5	6	7	8	9	10	11	12	13	14	15	16
32-bit	$2^{-7.1}$	$2^{-10.5}$	$2^{-13.9}$	$2^{-17.2}$	$2^{-20.7}$	$2^{-24.1}$	$2^{-27.4}$	$2^{-30.9}$	$2^{-34.3}$	$2^{-37.7}$	$2^{-41.2}$	$2^{-44.6}$
64-bit	$2^{-6.2}$	$2^{-9.9}$	$2^{-13.9}$	$2^{-18.0}$	$2^{-22.0}$	$2^{-26.1}$	$2^{-30.2}$	$2^{-34.2}$	$2^{-38.2}$	$2^{-42.3}$	$2^{-46.3}$	$2^{-50.4}$

Table 1. Theoretical estimate of good pair probability for RC5⊕ .

gives a rather low upper bound which may be lowered further.

¿From experiments with the program shown in Appendix A we observed that the structure of the sequences changes with increasing number of rounds. For less than eight rounds most of the sequences are non-iterative, and are based only on bit cancelations and randomly placed corrections (1-2 for 6 rounds, 2-3-4 for eight rounds). For a bigger number of rounds such as 10 or 12 the structure of sequences looks as concatenation of four iterative sequences (0,1,1,0,1,1,0,1,1,0,1,1 is a Fibonacci sequence with corrections on every third round) with a more corrupted Fibonacci sequence. We use this feature in Section 3.1 in order to build a very efficient filter of good pairs for RC5⊕. Though the pattern corresponding to the iterative sequence has the highest probability, the fact that corrections may be randomly located and not fixed to every third half-round as in an iterative pattern increases the total probability of such Fibonacci sequences considerably.

Another issue that we noticed while studying RC5⊕ is a strong key dependence of differentials. For example: the probability of a good pair after 10

[5] If $\mathcal{F}(i)$ is greater than $w - \lg w$, where w is the word size of the cipher, the sequence cannot produce a good pair.

full rounds of RC5⊕ is about 2^{-24} according to the Table 1, but in reality it ranges from 2^{-20} to 2^{-24} for different keys. Thus RC5 key can be weak in many unpredictable ways. Consider the following example of weak keys for RC5: Non-negligible fraction of all keys produce subkeys S_i with $S_i \equiv 0 \pmod{w}$ for some values of i (among 26 subkeys five or six "zero" subkeys is not a rare event). This fact increases probability of some differentials (for example with rotation patterns $x, x, 0, x$, where the last x is gained for free if S_i corresponding to zero rotation amount is zero modulo 32. Then the probability of this pattern is the same as for two concatenated iterative patterns (the corresponding Hamming weights look like 1,1,2,1,1,0 instead of 1,1,0,1,1,0). In any case if zero rotation corresponds to zero subkey modulo 32, the rotation in the next round is preserved which reduces the difference after that round.

The Attack on RC5⊕-32/12/16 In this section we present a very efficient differential attack on RC5⊕-32/12/16 which uses only 2^{28} chosen plaintexts.

The optimal input difference in our case is (e_w, e_w), since it peels off one round of RC5 with probability $\frac{w - \lg w}{w}$. This beginning is four times better than $(e_w, 0)$ and about 25 times better than $(0, e_w)$ for most of the Keys[6].

The probability p of a good pair of type (e_w, e_w) for 12 rounds is about $2^{-30.9}$ (see Table 1). Thus for a straightforward approach we will need a pool of $1/p = 2^{30.9}$ chosen pairs with input differences (e_w, e_w). Then the probability to find at least one good pair in such a pool will be about $0.63 \approx 1 - \frac{1}{e}$. In order to reach probability of 0.86 we will need twice as much data.

Since pairs with input differences $(e_i, e_i), i = 5, \ldots, 31$ will be as good as (e_w, e_w) in case of RC5⊕ , we will use the following idea of packing pairs into structures suggested in [1]. Suppose our attack can use successfully several linearly independent input differences δ_i, $i = 1, \ldots, k$, then for some plaintext A we will require the ciphertexts of $A, A \oplus \delta_1, A \oplus \delta_2, A \oplus \delta_3, \ldots, A \oplus \delta_1 \oplus \delta_2, A \oplus \delta_1 \oplus \delta_3, A \oplus \delta_2 \oplus \delta_3, \ldots, A \oplus \delta_1 \oplus \delta_2 \oplus \delta_3, \ldots$. Then a pool of 2^k such ciphertexts contains $k \cdot 2^{k-1}$ pairs with differences from the set $\{\delta_1, \ldots, \delta_k\}$. In our case using, the set of differences $(e_i, e_i), i = 5, \ldots, 31$ we need the encryptions of only 2^{27} chosen plaintexts packed into structures described above. Then this pool of encryptions contains $27 \cdot 2^{26} = 2^{30.8}$ chosen plaintext pairs suitable for our attack and the probability to find at least one good pair in it is about 0.63.

Once we have estimated the probabilities of good pairs the question is, how to detect them among all other pairs. As it was noticed in [4], in the good pair the five least significant bits of the ciphertext must agree $L_{2r+1} \equiv L^*_{2r+1} \pmod{32}$, since there should be no difference in rotation amounts. We noticed that the same must hold for the right halves of the ciphertext at the location,

[6] Previous attacks [3, 4] studied only iterative differentials that end by three or five bit differences, the starting patterns were predetermined to be (e_w, e_w), $(e_w, 0)$ or $(0, e_w)$ depending on the $2r + 1 \pmod{3}$. The key-detection algorithm used in [4] in order to define ten bits of the plaintext, corresponding to the gaining oracle subset, requires to start from the plaintext difference $(0, e_w)$ and proceed to $(e_w, 0)$ difference.

defined by rotation bits of the last round s_{2r+1}:

$$R_{2r+1} \ggg s_{2r+1} \equiv R^*_{2r+1} \ggg s_{2r+1} \pmod{32}.$$

These considerations help to filter 2^{-10} of all pairs as soon as they are generated.

¿From our definition of a good pair, and the fact that almost all good pairs still have one-bit differences after eight rounds we build a new very efficient filter. For each pair that passed our first ten-bit filter, we check if the ciphertext difference of this pair can be a result of some low Hamming weight difference in the flow of four rounds of RC5⊕. We do this by trying all possible rotation amounts that do not contradict the good pair assumption. Suppose, that $\triangle L_{16}$, $\triangle R_{16}$ (the left and the right halves of the difference after eight rounds) are one-bit and two-bit differences respectively (which is correct for most of the good pairs). In the case of a good pair the knowledge of the rotation amounts gives us the ability to calculate the differences back and forth. Thus starting with a particular difference after eight rounds we have to check all values for seven rotation amounts (35 bits), that do not contradict the good pair assumption (which holds with probability 2^{-10} for this difference during four rounds of RC5). Thus we arrive at about 2^{25} possible output differences after 12 rounds of RC5. However 2^{54} output differences are possible, passing our ten-bit filter. Thus the Signal/Noise ratio of our new "go up" filter is about: $S/N \approx \frac{2^{54}}{2^{25}2^{14}} = 2^{15}$ (here a factor of 2^{14} corresponds to the number of possible differences after eight rounds). Note that most of the patterns among 2^{35} coincide, and thus the ratio S/N should be even higher. This filter proved to be very efficient, exact and fast[7]. The fraction of random pairs that pass it is negligible and probability that it will reject a good pair can be made arbitrarily small. Implementation of this filter is given in Appendix C.

When a few good pairs are successfully detected we run a key-derivation algorithm. As in previous attacks, given a good pair it is possible to predict the value of the rotation amount at the $2r$-th round (several variants of this value in our case) and using this information to derive several key bits of the last subkey S_{2r+1}. We need between 30 and 60 good pairs in order to derive most of the bits of the subkey with high success probability. We defer the details of our modified key-derivation algorithm to the final version, since we consider it to be a less significant contribution of ours (and due to some similarity to previous results).

In our experiments we used Alpha Server 8400 Model 5/300. On this 64-bit machine with 300 MHz processor, the attack on RC5⊕ with 12 rounds uses about 2^{28} chosen plaintexts (which provide us with 2^{32} pairs) and takes 10–15 minutes (most of the time spent on encrypting the data, since we use the non-optimized reference implementation from [8]). The success rate of this attack is 86%.

[7] We use Fibonacci sequence to cut the search-tree, otherwise the search is exponential and may take hours for each pair. This hint does not degrade qualities of the filter.

3.2 The Attack on RC5-32/12/16

In this section we use methods that were developed in the previous attack, against the standard RC5-32/12/16 version. As for RC5$\oplus$ we encrypt a pool of plaintext pairs with the difference (e_w, e_w). The idea to use structures as in Section 3.1 also works, but we get a factor of 7 in chosen pairs instead of 27 since addition of the subkey in the first round may corrupt the differences (this happens with probability 1/4)[8]. The filtering procedure is more complicated due to subkey addition. If we suppose that a good pair has relatively small Hamming weight we can combine previous filter with Hamming weight filter. Probability of noise, i.e. for a random pair to have weight h, when particular ten bits of the difference are required to be zero, is expressed by: $\frac{C_{54}^h}{2^{64}}$, see Table 2. The ratio of

Probability of Noise ($Prob = C_{54}^h/2^{64}$)									
1	2	3	4	5	6	7	8	9	10
$2^{-58.2}$	$2^{-53.5}$	$2^{-49.4}$	$2^{-45.7}$	$2^{-42.4}$	$2^{-39.4}$	$2^{-36.6}$	$2^{-34.0}$	$2^{-31.7}$	$2^{-29.5}$
11	12	13	14	15	16	17	18	19	20
$2^{-27.5}$	$2^{-25.7}$	$2^{-24.0}$	$2^{-22.4}$	$2^{-21.0}$	$2^{-19.7}$	$2^{-18.6}$	$2^{-17.5}$	$2^{-16.6}$	$2^{-15.8}$

Table 2. Hamming weight filter for RC5-32/r/b.

the number of good pairs to the number of non-good pairs that may pass our filters (signal to noise ratio) for big numbers of rounds can be increased if we get several candidate pairs and compare to which space oracle they belong. Good pairs will have many consecutive bits in particular locations of the plaintext in common. The guess about a good pair can be tested with a small number of pairs, using the oracle suggested by this pair. Several successfully detected good pairs, help to detect another 30–60 good pairs which are required by our key derivation algorithm.

We performed experiments in order to determine the probability of a good pair. This probability for six rounds of RC5 is about $2^{-12.6}$, for eight rounds – $2^{-20.4}$, for ten rounds – 2^{-30}. We estimate that for 12 rounds this number is 2^{-38}. Although the probability of iterative three-round pattern for RC5 is four times smaller than for RC5$\oplus$ (because of carry after subkey addition), the results of our experiments show that the probability of a good pair for RC5 is only about $\frac{2(r-1)}{3}$ times smaller than probability of a good pair for RC5$\oplus$. This factor can be used to extrapolate our results to RC5-32/r/16 with $r > 12$. We performed successful attacks on RC5 with four, six, eight and ten rounds. Our calculations show that our attack on RC5-32/12/16 will require 2^{44} chosen

[8] Input differences, containing two or even three bits of the difference for each half of the plaintext may be worth studying, since a pool of structures contains them for free.

plaintexts (probably overestimate). In Table 3 we compare complexities of our attack with Knudsen-Meier results [4].

We suspect that as in the case of RC5⊕ one or two good pairs may suffice to start an attack and thus the data complexity will decrease to about 2^{38} at the cost of more thorough and time-consuming filtration of good pairs.

r	Our attack	Knudsen-Meier
4	2^7	2^{17}
6	2^{16}	2^{24}
8	2^{28}	2^{38}
10	2^{36}	2^{46}
12	2^{44}	2^{54}

Table 3. Number of chosen plaintexts for differential attacks on RC5-32/r/16.

3.3 Some Considerations on RC5-64/16/16

The RC5-64/16/16 version was suggested in [8] to be used with new generation of 64 bit computers. The best previously known attack is [4]. Their attack uses 2^{83} chosen plaintexts.

We suggest to study more complex differentials in the case of RC5-64/16/16, as we did for the case of RC5-32/12/16. Our hypothesis is that 64-bit version is much weaker than it was expected. The main reasons for weaknesses are:

- The ratio of the number of good pairs to the number of random pairs that may pass our filters is higher. It is $\frac{C^h_{116}}{2^{128}}$ for Hamming weight the difference h. Thus differences with much higher weights can be used (42 for 6 rounds, 35 for 8 rounds).
- Patterns similar to those that worked for 32 bit version work for 64 bit version as well with addition of a factor 1/2 for each three half-rounds (3-round iterative pattern). This factor is probably reduced due to extended locality properties.
- The fraction $\frac{w-lg(w)}{w}$ is bigger for bigger w.
- We expect that non-uniformity that we noticed in RC5-32/12/16 is even more influential in 64-bit version, since there is more space for strange "oracle" constructions.

The complete analysis of RC5-64/16/16 is yet to come, but based on the Table 1 our estimate is that this attack will require less than 2^{63} chosen plaintexts. This number though highly impractical is still by a factor of 2^{20} better than previous results.

4 Summary

In this paper we present an improved differential cryptanalysis of the RC5 block cipher. We study more complex differentials than in previous works and define a general notion of a "good pair" with respect to data dependent rotations. We analyze all pairs that escape differences in rotation amounts, and not only pairs that follow specific, easy to analyze, patterns as in previous attacks. We show that only one or two successfully detected good pairs suffice to start an attack on RC5 with high success probability. Thus the data complexity of cryptanalysis of RC5 is bounded by the probability of a good pair and our capability to detect good pairs. RC5 is heavily based on data dependent rotations. We show that probability for a pair to escape differences in rotations amounts is much higher than it was expected by the designers of RC5. This causes weak avalanche properties, and high key dependence of the cipher's properties.

We start by analyzing a simplified version of RC5, with all additions changed by XORs. We successfully attack this RC5$\oplus$-32/12/16 version with only 2^{28} chosen plaintexts. We show that XOR version serves as a good approximation to original RC5. Finally we present an attack, capable of breaking RC5-32/12/16 with 2^{44} chosen plaintexts. This is 1024 times less than the best previous attack. The complexity of the data analysis phase in our attack is negligible compared to the data collection complexity. We suspect that it may be possible to attack RC5-32/12/16 with only 2^{38} chosen plaintexts, using more efficient filtering and subkey detection algorithms (trading data requirements for complexity of analysis). We also estimate that RC5-64/16/16 is 2^{20} times weaker than it was expected.

We conclude that RC5-32/12/16 is not secure against chosen plaintext attacks. Though RC5 has already 12 rounds (24 half-rounds) we suggest to increase this number to at least 16 rounds in order to increase security against differential cryptanalysis. We estimate that RC5-32/r/16 reaches the level of theoretical security against partial differential attacks at 18–20 rounds (probability of a good pair is estimated to be lower than 2^{-64}).

References

1. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
2. X. Lai, J. L. Massey, S. Murphy, *Markov Ciphers and Differential Cryptanalysis.*, Lecture Notes in Computer Science 547, Advances in Cryptology – EUROCRYPT'91, pp.17–38, Springer-Verlag, 1992.
3. B. S. Kaliski, Y. L. Yin, *On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm*, Lecture Notes in Computer Science 963, Advances in Cryptology – CRYPTO'95, pp.171–184, Springer-Verlag, 1995.
4. L. R. Knudsen, W. Meier, *Differential Cryptanalysis of RC5*, European Transactions on Telecommunications, Vol. 8, N. 5, pp.445–454, 1997.
5. L. R. Knudsen, *Truncated and Higher Order Differentials*, Lecture Notes in Computer Science 1008, Advances in Cryptology – Fast Software Encryption, pp.196–211, Springer-Verlag, 1994.

6. M. Matsui, *Linear Cryptanalysis Method of DES Cipher*, Lecture Notes in Computer Science 765, Advances in Cryptology – EUROCRYPT'93, pp.386–397, Springer-Verlag, 1994.
7. National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, 1977.
8. R. L. Rivest, *The RC5 Encryption Algorithm*, Lecture Notes in Computer Science 1008, Fast Software Encryption, pp.86–96, Springer-Verlag, 1994.
9. A. A. Selçuk, *New Results in Linear Cryptanalysis of RC5*, to appear, proceedings of Fast Software Encryption, 1998.
10. Y. L. Yin, *The RC5 Encryption Algorithm: Two Years On*, RSA Laboratories' *CryptoBytes*, Vol. 2, No. 3, pp.14–16, 1997.

A Probabilities of Differentials for RC5$\oplus$

In this appendix we present a program which calculates probabilities of good pair differentials for RC5$\oplus$-$r/w/b$ with plaintext difference (e_w, e_w). It generates all Fibonacci sequences with corrections and calculates their total probability. Output of the program corresponds to one entry of Table 1. Note, that weights loaded in `F[0]` and `F[1]` correspond to symmetric differences (like `80000000 80000000`), `F[i]` being the weight of R_{i+1}. Last round does not influence probability calculations, since its result cannot corrupt the good pair property.

```
/* Differentials for RC5-XOR (Corrected Fibonacci Sequences)        */
#include <stdlib.h>
#include <math.h>
#define mc       20              /* Max number of corrections    */
#define w        32              /* word size in bits            */
#define lgw       5              /* lg w                         */
#define R        24              /* number of half-rounds,R=2*r  */
#define BND    w-lgw             /* w-lg(w) weight boundary      */

long int F[R]  = {R*0};          /* Fibonacci sequence           */
int      cp[R] = {R*0};          /* Corrections places           */
double   P     =    0.;          /* Probability of a sequence    */
double   acc   =    0.;          /* Accumulated probability      */
double Cm[BND+1];
void construct (int, int);
double Com(int);

/* Log ( C(BND,f) / C(w,f) )                                         */
double Com(f)
int f;
{ int j;
  double L=0.;

  for(j=0; j < f; j++)
      L=L+(log((float)BND-j)-log((float)w-j))/M_LN2;
  return(L);
}
```

```
/* Construct all Fibonacci sequences with up to mc corrections      */
void construct(i,c)
int i,c;
{ double tmp;

  if(c > 0){
      F[i]=abs( F[i-1] - F[i-2]);    /* Corrected Fibonacci behavior */
      cp[i]=1; tmp=P; P-=lgw; P=P+Cm[F[i]]-Cm[F[i-1]];
      if(i+1 < R) construct(i+1,c-1);/* Recursive call               */
      else acc+=pow(2.,P);           /* Accumulate probability       */
      cp[i]=0; P=tmp;
      F[i]=F[i-1] + F[i-2];          /* Regular Fibonacci behavior   */
      if(F[i] > BND) return;  P+=Cm[F[i]];
      if(i+1 < R) construct(i+1,c);
      else acc+=pow(2.,P);
  }
  else { while(i < R){               /* No more corrections          */
           F[i] = F[i-1] + F[i-2];
           if(F[i] > BND) return; P+=Cm[F[i]];
           i++;
         }
         acc+=pow(2.,P);             /* Accumulate probability       */
  }
}

main(){
  int i;

  for(i=0; i<=BND; i++) Cm[i]=Com(i);/* Precompute  coefficients     */
  F[0]=1; F[1]=0;                    /* Initial weights              */
  P+=Cm[F[1]];
  construct(2,mc);                   /* Construct Fibonacci sequences*/
  printf("%f\n",log(acc)/M_LN2);     /* Prob. of a differential      */
}
```

B Example of Difference Propagation

In this appendix we demonstrate an example of difference propagation in RC5 in the case of RC5-32/8/16 (eight rounds). The first two columns display the encryption process of the first plaintext block (results after each full round of RC5). The second two columns display the encryption process of the second plaintext. Rightmost columns provide Hamming weight (HW) of the halves of the difference. As one may see, the pattern in this case is not iterative, but the avalanche is slow. Our attack uses such ciphertext pairs in order to derive several bits of the last subkey.

Plaintext 1		Plaintext 2		Difference		HW	
D0F51C51	D0F51C51	50F51C51	50F51C51	80000000	80000000	1	1
CC747C0C	33C28840	CC747C0C	33C28040	00000000	00000800	0	1
229D964D	1CF30F80	229D9E4D	1CF30F80	00000800	00000000	1	0
8820939F	BEBDD31F	88208B9F	BEBDC71F	00001800	00001400	2	2
5DB4D400	093A8A00	5DB4DA00	093AA000	00000E00	00002A00	3	3
F3A47DDF	75D6EC04	F3A499DF	75D68D04	0000E400	00006100	4	3
0CD84724	61C22EFC	0CD07724	61471EFC	00083000	00853000	3	5
BF9D3541	D46209FB	BFA50541	D56809FB	00383000	010A0000	5	3
44E087D4	31FD103D	44D6F654	49FD2404	00367180	78003439	9	11

C Filter for RC5⊕

In this appendix we present an efficient implementation of the filter described in Section 3.1. One must place ciphertext differences after $(2r-1)$th round (one half-round before the last half-round) into $D[Level+1]$ and $D[Level]$ before calling the filter. These differences can be calculated (for RC5⊕) from the known ciphertext differences and the known rotation amount of the last round. A call to the filter function looks like: `GoUP(Level)`.

```
typedef unsigned long int WORD;                /* 32-bit for RC5-XOR-32/12/16  */
#define Level         7                        /* Depth of search              */
int     Fib[Level+1] = {1,2,3,5,8,13,13,13};  /* Fib. cuts for a search tree  */
WORD    D[Level+1];                            /* Array of differences         */
int     count        = 0;                      /* Number of variants           */
unsigned int HAM();                            /* Returns Hamming weight       */
WORD         ROTR();                           /* Rotation to the right        */
/* This function carries out filtration of good pairs for RC5-XOR by checking*/
/* the structure of the Fibonacci weight sequence of the particular pair.    */
int     GoUP(depth)                            /* Returns 1 if the pair is good*/
int     depth;                                 /* Depth of recursion           */
{       int i,s, flag= 0;
        if (HAM(D[depth]) > Fib[depth]) return(0); /* Cut the search tree    */
        if(depth!=0)
              for(s=0; s<32; s++)              /* Try all rotations 0..31      */
                  if( ((ROTR(D[depth+1],s)) &0x1F)==0){
                       D[depth-1] = ROTR(D[depth+1],s)^D[depth];
                       if(GoUP(depth-1))       /* Recursive call               */
                               flag=1;         /* For correct count of variants*/
                  }
                  else   {  count++;           /* Accumulate num. of variants  */
                            return(1);
                  }
        return(flag);
}
```

This article was processed using the LaTeX macro package with LLNCS style