From Differential Cryptanalysis to Ciphertext-Only Attacks

Alex Biryukov[1] and Eyal Kushilevitz[2]

[1] Applied Mathematics Department,
Technion - Israel Institute of Technology, Haifa, Israel 32000.
albi@cs.technion.ac.il
[2] Computer Science Department,
Technion - Israel Institute of Technology, Haifa, Israel 32000.
eyalk@cs.technion.ac.il

Abstract. We present a method for efficient conversion of differential (chosen plaintext) attacks into the more practical known plaintext and ciphertext-only attacks. Our observation may save up to a factor of 2^{20} in data over the known methods, assuming that plaintext is ASCII encoded English (or some other types of highly redundant data). We demonstrate the effectiveness of our method by practical attacks on the block-cipher Madryga and on round-reduced versions of RC5 and DES.

Keywords: block-ciphers, Madryga, RC5, DES, ciphertext-only attack, differential cryptanalysis, differential-linear attack.

1 Introduction

Differential cryptanalysis [1,12] is a very powerful technique for the analysis of block-ciphers. It has been used with success against many block-ciphers, e.g. [1,2,3,18,4]. One weakness of differential cryptanalysis is that it finds *chosen plaintext* attacks; these are much less practical than known-plaintext and certainly than ciphertext-only attacks. Ciphertext-only attacks are the most useful attacks on cryptosystems, since they require only passive eavesdropping from the attacker. Such attacks are usually hard to find, since the assumptions on the knowledge of the attacker are minimal. Exceptions include the most basic ciphers, like simple substitution or Vigenère [11].

Although there exists a general method for converting any differential chosen plaintext attack into the more favorable known plaintext attack [1], this conversion becomes (almost) impractical due to the huge increase in the data requirements. If a differential attack uses m chosen-plaintext pairs, the corresponding known-plaintext attack will need about $2^{\frac{w}{2}}\sqrt{2m}$ known plaintexts, where w is the block size (in bits) of the analyzed cryptosystem. For example, if a differential attack on a cryptosystem with 64-bit block uses only eight chosen-plaintext pairs, the corresponding known-plaintext attack will require 2^{34} known plaintext-ciphertext pairs, an increase which makes this attack much less practical.

H. Krawczyk (Ed.): CRYPTO'98, LNCS 1462, pp. 72–88, 1998.

In this paper we show a method of converting successful differential chosen-plaintext attacks into known-plaintext and even ciphertext-only attacks without loosing as much efficiency as the above mentioned method, and under a reasonable assumption that plaintext comes from a redundant source. Notice that due to plaintext redundancy, the probability of some input differences increases, and the probability of other input differences decreases or even becomes negligible. If the probability of the input differences which are useful for the differential attack is increased (depending on the type of input redundancy and the type of input differences required for the attack on the particular cipher), then the cipher is weaker against the differential attack combined with the redundancy assumption. We show, for example, that under the assumption that plaintext comes from ASCII encoded English encrypted in ECB (Electronic CodeBook) mode, the probability of input differences with small Hamming weight (which are the differences needed for most of the known attacks) increases significantly. Therefore, only about 2^{14} known-plaintexts are needed for a known-plaintext attack of the previous example, saving a factor of 2^{20} in data. Moreover, our observation helps to turn differential attacks into much more desirable ciphertext-only attacks, with modest increase in data. Our efficient conversion method applies also for the combined differential-linear attacks [13], which can be converted into efficient known-plaintext attacks.

This paper is organized as follows: In section 3 we outline the principles of our method. Then, we demonstrate its applicability for various ciphers; we start by presenting a new differential attack on Madryga [14,21] with only sixteen chosen plaintext pairs. We use this cipher as a testing ground for the development of our ideas. Then, we proceed to a ciphertext-only attack on Madryga with only several thousand ciphertexts. We continue demonstrating the effectiveness of our approach with a ciphertext-only attack on 4-round RC5 using only 2^{17} ciphertexts, and a known plaintext attack on 6-round RC5 (as of today this is the first known plaintext attack on this cipher) with about 2^{18} plaintext/ciphertext pairs (the previous known-plaintext attack on this cipher [8] required 2^{57} for 6-round RC5 but it was found erroneous [22]). We show a new known-plaintext attack on seven round DES [19] with about 2^{17} known plaintexts.[1] Finally we show, that our attacks are applicable not only to ECB mode, but also to the first block of the CBC (Cipher Block Chaining) mode if the initial vector (IV) is unchanged for several datagrams or incremented sequentially, starting from a random value (as is usually the practice on the Internet), and to the counter mode [21].

To conclude, we show that differential attacks are very subjective to the underlying plaintext redundancies. We mark the importance of studying differential attacks on ciphers together with the underlying redundancies of the protocols, they are used in. We also suggest methods, that may help to prevent attacks of the kind described in this paper.

[1] For FEAL [16,17] there exist several very efficient known-plaintext attacks which use specific features of this cipher. Our analysis is applicable to FEAL as well but yields inferior results.

2 Differential Cryptanalysis

Differential cryptanalysis is a very efficient chosen plaintext attack on block-ciphers [1]. The idea of differential cryptanalysis is to analyze pairs of plaintexts instead of single plaintexts. An attacker chooses the difference ΔP between plaintexts (P, P^*) and studies the propagation (avalanche) of the changes in the encryption process. During the attack he searches and then studies the ciphertexts pairs (C, C^*), which exhibit difference ΔC, predicted by his analysis.

Let us introduce some terminology related to differential cryptanalysis. The difference between two bit-strings X and X^* of equal length is defined as $X \oplus X^* = \Delta X$, where $\oplus$ is a bitwise XOR operation. We call a pair of plaintexts (P, P^*) a **good pair** with respect to differential analysis of a cipher, if it exhibits the difference propagation and the output difference ΔC, predicted by the analysis of a cipher. We call **noise** all pairs that are suspected to be good pairs (i.e. pass all our criteria for good pairs (which we call **filters**)), but which do not exhibit the difference propagation, predicted by the analysis.

It is well known, that unlike other chosen plaintext attacks, differential cryptanalytic attacks can be easily converted to known plaintext attacks [1]. The idea is similar to the Birthday paradox. Denote the length of the blocks in bits by w. Suppose that differential chosen plaintext attack needs m pairs to succeed, and that we are given $n \approx 2^{\frac{w}{2}}\sqrt{2m}$ random known plaintexts and their ciphertexts. These plaintexts can form about $2^w \cdot m$ pairs. Since the block size is w bits, there are only 2^w possible different XOR values and thus, due to the uniform distribution of the plaintexts, there are about $\frac{2^w \cdot m}{2^w} = m$ pairs for each XOR value (and thus, we expect m pairs with the specific differences, used by the attack). In order to find pairs with useful input differences, one can sort the array of n known plaintext/ciphertexts by plaintexts, and then search for pairs with particular differences; the total complexity of this process is $O(n \log n)$. Once these pairs are discovered a regular differential attack on the cipher may begin.

3 On Ciphertext-Only Attacks

In this section we describe our method for converting successful differential attacks into ciphertext-only attacks (and known-plaintext attacks) with huge savings over the method described in the previous section. We show practical cases, where this conversion can be applied very efficiently.

The essence of differential cryptanalysis is in studying the *differences* between plaintexts, without using the plaintexts themselves (although there are many ways of helping the analysis by adding the information about plaintexts). Thus, one can perform a ciphertext-only attack on a cipher as soon as he is able to detect ciphertext pairs that come from good plaintext pairs. Suppose that we are in a ciphertext-only attack scenario and that we are given only a pool of n ciphertexts, without the knowledge of the corresponding plaintexts. It may seem that we will have to check all the $\frac{n^2}{2}$ pairs, which is usually infeasible, and that we will not be able to detect pairs that exhibit useful input difference (as we show

in the previous section). However, two observations may help in our case. First, the structure of the ciphertext difference in a good pair may be very restricted, and thus the search in a sorted pool of ciphertexts will still have $O(n \log n)$ complexity. The second observation is related to the possible redundancy of the encrypted plaintexts. Usually not all 2^w blocks are "legal" plaintexts, and the probabilities of possible plaintexts may be very non-uniform. Thus, among the pairs taken from a pool of redundant plaintexts, some differences will be very frequent, and some will never occur.

Let us proceed with an important but simple to analyze example of plaintext redundancy. Suppose that the plaintext source, produces blocks of w bits, and has entropy of e bits (so there are $w-e$ redundant bits per block). Suppose also that the entropy is "bit-local": a fixed subset of e bits out of w is chosen and may get all 2^e possible values uniformly. The other $w-e=r$ redundant bits are fixed to arbitrary values. This sort of redundancy describes many types of computerized information (database records, network packets, etc.). Denote the set of all such blocks as S_e, and denote the set of all differences, produced by elements of S_e as ΔS_e. Let there exist a fast differential attack on the cipher, which succeeds with m chosen pairs, using differences from the set ΔS_e. If the differences used by the differential attack are not in ΔS_e, then the cipher (taken together with the redundancy assumption) is more secure against this differential attack, than in the general (uniformly random plaintext) case[2]. For the described type of redundancy, we can use only $n_e \approx 2^{\frac{e}{2}}\sqrt{2m}$ ciphertexts, which form about $2^e m$ pairs. This pool contains about m pairs with useful input differences for a differential attack. If the corresponding plaintexts are known to the attacker, he can simply sort the table of given plaintexts and search in the sorted table for pairs that exhibit the necessary input differences and proceed to analysis of the pairs as in a regular differential attack.

If the corresponding plaintexts are not known, we are in a ciphertext-only scenario. If we expect (due to differential analysis) a ciphertext difference in a good pair to have a definite structure (for example, ΔC should be a particular constant), then the probability for a random pair to have similar ciphertext difference may be as small as 2^{-w}. Suppose for simplicity, that the probability of a good pair is $p \approx \frac{1}{m}$ (i.e., one good pair is enough to start a differential attack, which is true in many cases). Since in a ciphertext-only scenario we do not know which m pairs exhibit useful input differences, we can only hope that about one pair in a pool of $2^e m$ pairs of ciphertexts will be good for the differential attack[3]. The Signal/Noise ratio (ratio of the probability of a good pair to the probability

[2] This may lead to a method of strengthening for differentially-weak ciphers. Add redundancy to the plaintext, in a way, that prohibits successful input differences. For example one can use error correcting code (plaintexts be codewords), in order to avoid input differences of low Hamming weight.

[3] However, in order to find it we do not need to check all $2^e m$ pairs. If we are looking for a well defined ciphertext difference, search in a sorted pool of n_e ciphertexts will have $O(n_e \log n_e)$ complexity.

of noise) for the ciphertext-only attack in this case is $S/N = \frac{2^{-e}m^{-1}}{2^{-w}} = \frac{2^r}{m}$. The attack will be successful if $S/N > 1$.

We can generalize the description above in two ways. First, it may be more useful to consider a set of possible "good" output differences Δ_G of size 2^k, rather than one output difference. If differences from the set Δ_G are equally likely to appear in a good pair, then the Signal/Noise ratio will decrease to $\frac{2^{r-k}}{m}$, but the probability of a good pair will increase (since we relax the conditions on the differential propagation pattern), and thus m will decrease. In this case we have to solve a more complex search problem. For example, we would like to find all pairs with differences of low Hamming weight in reasonable time. We can state this problem as follows:

Problem 1 *Find all pairs $(s_i, s_j), i \neq j$, in a given set $\mathcal{S}$ of n binary words from $\{0,1\}^w$, such that $d_H(s_i, s_j) \leq k$ (here d_H denotes Hamming distance). Let us call such pairs k-neighbors.*

We can reduce this problem to a well studied approximate string matching (approximate dictionary queries) problem. However most of the algorithms for this problem are linear in the document size n. Since we have to call this algorithm n times (to check neighbors of each element of the set), this results in a complexity of $O(n^2)$, which is the complexity of checking all pairs in the set. In [23] an efficient algorithm, based on tries, which runs in $O(k|\Sigma|^k)$ expected worst case time (here $|\Sigma|$ denotes the alphabet size), independently of the document size (n) is presented. Trie indices combine suffixes and so are compact in storage. Applying this algorithm to our problem, we get $O(nk2^k)$ complexity. Though exponential in k, it still provides a better algorithm than exhaustive check of all pairs if $\log n > k + 1 + \log k$. Thus, for a set of size 2^{20}, the search of all 15-neighbors can be performed faster than 2^{39} using tries.

Since we have seen that redundancy in some cases helps for differential attacks, our second observation concerns another useful type of redundancy — the natural language redundancy. For a natural language L over alphabet A, denote by A^n the set of all n-grams of the language. Then the n-gram entropy of the language is:

$$H(X_1, \ldots, X_n) = \sum_{s \in A^n} -P(\chi_n = s) \log P(\chi_n = s),$$

where $\chi_n = (X_1, \ldots, X_n) \in A^n$. We say that the language has entropy H_L if:

$$H_L = \lim_{n \to \infty} \frac{H(X_1, \ldots, X_n)}{n}$$

In the case of the English language successive approximations of H_L go as: $\log_2 26 \approx 4.7$, the first order approximation (English letter frequencies) gives ≈ 4.2, digram frequencies give ≈ 3.9. Shannon [24] suggests a value of 2.3 bits per letter for eight letter blocks. By various experiments, for large n the entropy decreases into the interval $1.0 \leq H_L \leq 1.5$. By a gambling technique Cover and King [6] give an estimate of 1.3 bits of information per letter. In [5] an upper

bound of 1.75 bit per letter is estimated from a large sample of ASCII encoded English text.

Our experiments with large English files show, that some differences are very frequently encountered, even in a small quantities of English plaintext. For example, differences with low Hamming weights (especially one-bit differences at the beginning and at the end of the block) are very frequent. For a more detailed study of these differences see Appendix A. This fact can be used in a differential known-plaintext and even in differential ciphertext-only attacks on block-ciphers that are weak with respect to these differences. In the following sections we demonstrate new attacks on Madryga, RC5 and DES which follow the ideas expressed in this section[4].

As we explained here and as we will show in the further sections, differential attacks are very subjective to the underlying plaintext redundancies. We stress the importance of studying differential attacks on ciphers together with underlying redundancies of the protocols, they are used in.

4 Attacks on Madryga

We used the Madryga block-cipher [14,21] as a testing ground for the development of our ideas. In the following subsections we describe this cipher and our attacks on it. We first find a very fast differential attack on Madryga, which uses negligible amount of data, and then proceed to a differential ciphertext-only attack on this cipher, which is also very efficient.

4.1 Description of Madryga

Madryga is a blockcipher proposed in 1984 by W. E. Madryga [14]. It was designed for efficient software implementation. It consists of data-dependent rotations and exclusive or's with the bytes of the key. Madryga was designed as an alternative to DES (with larger key size – 64 bits) in order to permit efficient implementation both in software and hardware.

Here is a description of the encryption algorithm. Block size and key size in Madryga may vary, but 64-bit block size was suggested for compatibility with DES. The key size in this case is also 64 bits. The encryption process consists of two nested cycles. The out-most cycle consists of eight iterations of the inner cycle. The inner cycle consists of eight local operations on the block. A work frame (`Frame`) of three consecutive bytes $b_1b_2b_3$ is chosen in the plaintext block (`Text`), starting from the second last byte (the block is treated as a cyclic entity). The 64-bit key (`Key`) is rotated by three bit positions to the right and exclusive or'ed with the 64-bit constant (`KeyHash`). Rotation amount is extracted from the three least significant bits of b_3. Then the least significant byte of the key is exclusive or'ed with b_3. The concatenation of b_1b_2 is rotated by

[4] For the rest of the paper, all attacks are described for the English language model of redundancy, but they work even better in the "bit-local" model of redundancy, when the entropy e is the English language entropy.

the rotation amount to the left. Then the working frame is shifted one byte to the right and the process continues. The working frame moves to the right from the starting second-last byte to the starting third-last byte. Here is a Madryga implementation (WORD is 64 bits):

```
/* MADRYGA encryption engine, 64-bit implementation.            */
WORD EncKey;                            /* Secret Key           */
WORD KeyHash =  0x0F1E2D3C4B5A6978;     /* Key Hash Constant    */
WORD Key;                               /* Work Key             */
WORD Text;                              /* Plaintext block      */
WORD Frame;                             /* Work Frame           */
#define         FrameMask       0xFFFF
#define         TextMask        0xFFFFFFFFFFFF0000

Key = EncKey;
for(i=0; i < 8; i++){                           /* Eight rounds     */
       for(j=0; j < 8; j++){                    /* Eight bytes/block*/
                Frame   = ROTL(Text,8*j)&FrameMask;/* Copy to Frame    */
                Key     = ROTR(Key,3)^KeyHash;  /* Rotate & XOR     */
                                                /* with Hash        */
                rotation_count = (Text >> (56-8*j)) & 0x7;
                Text   ^= (Key & 0xFF) << (56-8*j);
                RotateFrame( rotation_count);
                                                /* Copy from Frame  */
                Text  = (Text & ROTR(TextMask,8*j)) | ROTL(Frame,64-8*j);
       }
}
```

4.2 Differential Attack on Madryga

In this subsection we present a very efficient differential attack on Madryga, with 82% probability of success, that uses only sixteen chosen plaintext pairs. Another differential attack, using 5000 chosen plaintexts was developed by Shirriff [25].

We start with two observations. The first observation is that keyscheduling in Madryga is very simple: the initial key is at least as long as the block-size (64 bits in our case). In every iteration (every time the work frame is shifted) the key is rotated to the right 3 bits, and exclusive or'ed with the key hash constant. That means that simple equations relate bits of the key at each stage to the bits of the initial key. The second observation is that block-sizes larger than 64 bits may only weaken Madryga. Data-dependent rotations in Madryga are very local, and are the only tool to provide avalanche. It will take much longer for small differences to propagate in a larger block.

We note that rotation amounts of the first round are known to an attacker. They are saved before the last byte of the working frame is exclusive or'ed with the corresponding byte of the key. Thus in a chosen plaintext attack, starting with one-bit difference in plaintexts, and setting to zero (or other appropriate value) two rotation amounts that may move this bit, one can assure that one-bit difference is preserved after the first round (after eight frame operations)

and after the second round, if difference is placed in the upper five bits of the byte. With probability $(0.57)^4 \approx \frac{1}{10}$ (the probability for a difference bit not to influence the rotation amounts after one round is intuitively $\frac{5}{8}$, but due to weak randomizing properties of Madryga it is slightly lower – 0.57; exact analysis was done by [25] and is confirmed by our experiments) this one-bit difference will propagate to the end of the sixth round (we number the rounds of Madryga $1, 2, \ldots, 8$). At the 7th and 8th rounds, we allow the difference to propagate arbitrarily (and it is even better for us if the one-bit difference will not be preserved).

Due to the very slow avalanche in Madryga, it is clear that starting with chosen plaintext pair differing by one bit, we will see many zeroes in the ciphertext difference. The question is: what information about the secret key can be derived from such a pair. In order to learn more from each pair, we do not care for the behavior of the difference in the last two rounds (similar to the 2R attack in [1]). This may cause many ciphertext differences, but we use a "go up" idea similar to [4] to trace guesses about the key bits and the difference to a one-bit difference after the sixth round. This way we get more bits of the key from each pair (at least four and up to thirty bits, depending on the difference locations in the last round), and at the same time increase the probability of a good pair. Since each found key bit can be traced exactly to a bit of the initial key (due to a simple keyscheduling algorithm in Madryga), our aim is to find about 30 key bits. The rest can be found by a reduced exhaustive search.

Putting together the ideas expressed in this section, we need only about sixteen chosen pairs with one-bit differences in order to find partial information about the secret key. The rest can be done by a reduced exhaustive search, which becomes feasible. Success probability of this attack is $1 - (1 - \frac{1}{10})^{16} \approx 0.82$.

4.3 Ciphertext-Only Attack on Madryga

In this section we demonstrate how our method can be used in order to derive ciphertext-only attacks. The differential attack described in the previous section succeeds with only about sixteen chosen pairs. Therefore the corresponding known plaintext attack, due to the conventional technique [1], will require about 2^{35} known plaintexts. Although this number can be reduced, since any one-bit, and even two-bit input difference will suffice for our attack, the pool of required known plaintexts is still too big.

Under a reasonable assumption of highly redundant plaintext (for example English text in ASCII format) the amount of data can be significantly reduced. Let us present an example of a ciphertext-only attack on Madryga with only $\approx 2^{12}$ ciphertexts, corresponding to English plaintexts (95 printable ASCII characters), taken from the beginning of "The Hound of the Baskervilles" by Sir Arthur Conan Doyle. In the pool of $\binom{2^{12}}{2} \approx 2^{23}$ English pairs, on the average about $2^{6.6}$ pairs have one-bit differences. The probability that a one-bit difference will be preserved during six rounds of Madryga is about $(0.57)^6 \approx 2^{-5}$ (this is lower than in the chosen plaintext case, since we cannot control the location of the difference bit and the surrounding rotation counts before the first

round). Thus, about four good pairs are contained in the pool (even more if we consider other small input differences). Some of those can be detected due to low Hamming weight of the ciphertext difference. For any pair suspected to be a good pair, we "go up" two rounds (using an idea from [4]), performing the necessary guessing, and check if we arrive at a one-bit difference. This idea is a very efficient filter, which helps us to discard almost all wrong pairs. If the pair is good, it provides us with 10–30 bits of the secret key. Combining the guesses from several good pairs we find about 30 bits of the key. The rest can be found by a simplified exhaustive search, or by further analysis of the given ciphertexts.

Below, we provide an example of the attack discussed above. Note that plaintexts and `Plaintext Differences` are unknown to the attacker, and they appear here only for the sake of completeness. We printed only ciphertext pairs with Hamming weight (`HW`) less than 12 (the probability for a random pair to have such a low Hamming weight is $\approx 2^{-25}$). The probability for a random pair to have one-bit ciphertext difference is $\frac{64}{2^{64}} = 2^{-58}$ and in this example there are two such differences[5]!

```
Plain1     Plain2     Plaintext Difference   Ciphertext Difference  HW
-------------------------------------------------------------------------
'e have b' 'u have b'  1000000000000000        0000008000000000       1
't such a' 'd such a'  1000000000000000        0010000000000000       1
'      fr' '      br'  0000000000000400        000000D1E01C0100      11
```

5 Known Plaintext and Ciphertext-Only Analysis of RC5

RC5 is a fast block-cipher designed by Rivest in 1994 [20]. RC5 has an attractively simple 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 variable number of rounds r and a variable-length cryptographic key b (so that the user can choose the level of security appropriate for his application). The "nominal" choice of parameters proposed in [20] 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.

We use a description of RC5 as a so-called Feistel cipher from [8]. 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:

[5] Note that one-bit ciphertext differences are less useful to us in this scenario, since they disclose less bits of the key.

$L_1 = L_0 + S_0$
$R_1 = R_0 + S_1$
`for` $i = 2$ `to` $2r + 1$ `do`
$L_i = R_{i-1}$
$R_i = ((L_{i-1} \oplus R_{i-1}) \lll R_{i-1}) + S_i,$

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 $w = 32$, then the 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 are called *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.

Several attacks on RC5 were published [8,10,4]. Currently the best known attack on RC5-32/12/16 is [4], a differential attack which uses 2^{44} chosen plaintexts. As of today no known-plaintext attack on RC5, even with reduced number of rounds, exists due to recent result [22] which found gaps in the linear attack on RC5 described in [8]. In this section we show a ciphertext-only attack on RC5 up to four rounds using only 2^{17} ciphertexts and a known plaintext attack on RC5 up to six rounds, which requires about 2^{18} known plaintext-ciphertexts.

The attack described in [4] uses chosen plaintext pairs with small input differences[6]. Our main observation is that, as in the previous section, assuming that plaintext comes from an ASCII encoded English (all 95 printable characters are possible), the probability of a small input difference is very high. This is true due to high redundancy of the English language, and due to the special properties of ASCII encoding (for example, upper and lower case letters differ in exactly one bit, etc.). See Appendix A for a study and examples of one-bit differences in ASCII encoded English. Our experiments show that if the differential attack on RC5 uses m pairs, the corresponding known-plaintext attack by our method will use $\approx 2^{10} \cdot \sqrt{2m}$ known plaintexts. For example, since the differential attack on six rounds of RC5 requires 2^{16} chosen plaintexts [4] our method provides a known plaintext attack on six-round RC5 with about 2^{18} known plaintexts. However there are several important differences between the two approaches. First of all, note that the actual probability of a good pair for six rounds of RC5 is estimated in [4] to be about $2^{-12.6}$, and 2^{16} chosen plaintexts are required in order to detect several good pairs. Then, the differential attack generates new pairs according to the "space oracle" suggested by several good pairs. For example, pairs with the same five least significant bits of L_0 and R_0 as in a good pair. This increases the probability of finding additional good pairs. In our case, we can also detect the oracle structure, but we cannot use it to a full extent by generating pairs according to it. However, we can still use it in order to refine the filtration process, since many good pairs are missed by the attack at the first stage. In our experiments we used 2^{18} different known plaintexts (taken from books by Charles Dickens) and they resulted in about $2^{16.6}$ pairs with one-bit

[6] This attack uses mainly symmetric two-bit differences like `80000000 80000000`, however symmetric differences are more rare than one-bit differences in our case, and thus we use one-bit differences, which are still good enough.

difference. Among these we were able to detect six pairs with ciphertext difference Hamming weights less than 20 (the probability for a random pair to pass this test is about $2^{-16.6}$), ranging from 14 to 19. They belonged to three space oracles: two pairs to one oracle, three to another and one pair to another one. Having detected the space oracles, it becomes possible to detect more good pairs missed by the first step, due to the increase of the Signal/Noise ratio by a factor of $\approx 2^{10}$. In fact, the pool of $2^{16.6}$ plaintexts with one-bit differences contains 8–16 good pairs. This amount of good pairs is enough in order to find about ten bits of the last round subkey of RC5. Although the full attack on RC5 according to [4] will require about 30 good pairs, it seems that a more thorough analysis of the given pool of pairs, assisted by a partial knowledge of the key will defeat six-round RC5 (two-bit and three-bit differences, that we omitted for simplicity of the present description will help).

Furthermore, the pool of 2^{17} ciphertexts can be used for a ciphertext-only attack on RC5 with four rounds, and detect several good pairs even for five rounds of RC5. The decrease in the number of rounds is due to more complicated filtration, required by the ciphertext-only attack. The idea of this attack is as follows. We know, that the set of all 2^{33} pairs produced by a given pool contains lots of good pairs (more than 2^7 good pairs). The problem is that without knowing the corresponding plaintexts we have to check all the possible pairs, which considerably complicates the filtration process. The solution is to search for the differences with lower Hamming weight than in a known plaintext case (less than 10). Note, that oracle information from the plaintexts is unavailable in the ciphertext-only case. However, we are still able to detect about 16 good pairs with low Hamming weights (from 4 to 8 in our experiment) and derive the partial key information from them exactly as in the known plaintext case. Using the partial key information, we can proceed to check the pairs with heavier differences and thus find almost all bits of the last subkey. The attack then continues with the same data, but with RC5 reduced by one-half round. Note, that given a pool of the same size, encrypted by RC5 with five rounds, we still can detect several good pairs with high probability.

A natural question that arises here is why we do not continue our attack for more rounds of RC5. For example, due to our calculation the known plaintext attack on eight-round RC5 by our method will require about $2^{10} \cdot \sqrt{2 \cdot 2^{28}} = 2^{24.5}$ known plaintexts which still is a very practical amount of data[7]. The problem is that we need *different* plaintexts. In the case of a redundant source of plaintexts we can easily get about 2^{nH} blocks (where n denotes the size of the block in bytes, and H denotes the entropy a single byte (letter)). For the English language, organized in eight-letter blocks, we can get about 2^{18}–2^{19} different plaintext blocks in a file of similar size (about 30% blocks appear several times in the file of this size). Although it is possible to exceed the 2^{nH} bound due to many rare blocks that there are in English, one will have to request much more data in the search for new blocks. Notice, however, that for 128-bit blocks this bound is

[7] A pool of only about 2^{20} different English pairs contains one good pair for eight-round RC5 with high probability, but how to detect it?

about 2^{30} (due to the larger block size, the entropy is lower — between 1.5 and 1.7, and the distribution of the blocks is much more uniform). Thus, our method is able to penetrate RC5-64/r/16 (a 128-bit version of RC5) for more rounds. We leave the details of this attack for the full version of this paper.

6 Known Plaintext Analysis of DES

The Data Encryption Standard [19] is one of the most important block-ciphers nowadays. In the 90-s two powerful methods of cryptanalysis were developed in attempt to break DES: differential cryptanalysis [1] and linear cryptanalysis [15]. The first attack uses 2^{47} chosen plaintexts, the second uses 2^{43} known plaintexts. A mixed differential-linear approach was developed in [13] and successfully demonstrated on DES reduced to eight rounds. This combined approach starts as a differential attack for four rounds, preserving parity of particular bit subsets, which are then used by linear relations for the last four rounds. Their attack is a chosen plaintext attack with about 512 chosen plaintexts.

In this section we show that due to differential behavior of the first rounds of the attack of [13], it is perfectly suitable for our method of converting differential attacks into known-plaintext and ciphertext-only attacks. Note however that we cannot afford structures, that were used in [13] to gain the first round for free and to decrease the data requirements from 1400 to 900 pairs. Thus, our attack works against DES, reduced to seven rounds (and not eight rounds as in [13]).

We checked the probability of the input difference used in [13], assuming that plaintext is ASCII encoded English[8]. In our experiment we used a file with the book "David Copperfield" by Charles Dickens. This file contains 2^{18} eight-byte blocks, among these about $2^{17.3}$ blocks are different. This file contains more than 2^{15} different pairs with one-bit input differences. However these differences are distributed very non-uniformly between the 64 possible bit-locations (see Appendix A). About two thirds of the differences occur in the four least significant bits of the first byte and in the four least significant bits of the last byte. The probability of a one-bit difference is relatively low for the middle bits of the block. Taking into account the initial permutation (IP) of DES (which was irrelevant to the analysis in [13]), the two bit locations toggled by the differential-linear attack occupy bits 42,50 (corresponds to original 2,3 bits) or bits 44,52 (corresponds to original 10,11 bits). In our experiment only $2 + 26 = 28$ pairs toggled the first group of bits, but $388 + 1348 = 1736$ toggled the second group. This data is more than sufficient, since 1400 pairs can be used by [13]'s attack in order to find six bits of the key with success probability more than 95%.

Note that our aim here is to demonstrate how our ideas work in practice and not to provide the most efficient attack in a specific case. This task would require a thorough study of the underlying redundancy together with the cryptanalysis of the cipher. For example, one may find new attacks, which are optimal only in this special case. See table 1 for a summary of our results.

[8] The input difference toggles bits 2 and/or 3, or 10 and/or 11, the bits are counted from 1 to 64 from msb to lsb.

Cipher	Data for the Attack
Madryga	16 chosen plaintext pairs or 4000 ciphertexts
RC5 (4 rounds)	2^{17} ciphertexts
RC5 (6 rounds)	2^{18} known plaintexts
DES (7 rounds)	2^{17} known plaintexts

Table 1. Summary of our attacks for English plaintexts or bit-local redundancies with English entropies.

7 Discussion

In this section we discuss modes of operation of block-ciphers in the context of our attacks. See [21] for a full account on available modes of operation for block-ciphers.

The attacks described in this paper are applicable first and foremost to redundant data encrypted in ECB mode or to the case, when by an error CBC *decryption* is performed instead of CBC encryption. However, data for these attacks can be derived from the CBC encryption mode as well in the case, when the initial vector (IV) (which is added to the first block of the plaintext before the encryption) is not frequently changed. Another case is the case of sequential IV increments (for example, a counter starting from a random value). These are common practice on the Internet, and are suggested in [9] and in several other Internet-drafts. This method is used to prevent repetitions of the first block, assuming that the first block of the datagram is constant in many applications. However, this method of IV choice provides many pairs, all with small input differences, which can be used by our attack (given the first blocks of 2^k such datagrams we get $k \cdot 2^{k-1}$ pairs with one-bit differences). Another method of IV choice is the encryption of thee datagram sequence numbers or other incremented entities, and sending IV in clear (explicit IV method) in order to avoid problems with loss, duplication or re-ordering of datagrams. This method is also very vulnerable to our analysis, and so is the case with other cryptographicaly weak pseudo-random IV generators (even having relatively long periods). Note that compression option, available in several protocols, does not influence this attack, since it uses only the first block of the plaintext. A good method of IV choice, that is suggested in several recent Internet-drafts, is the last ciphertext block of the previous datagram. This method seems to provide much better resistance to the attacks described in this paper.

Another application of our attack is to the block-cipher counter mode [21]. In this case a sequence of numbers is used as the input to a cipher in order to generate a pseudo-random stream. However, due to the sequential nature of the numbers being encrypted, the attacker obtains lots of data with small input differences, so this mode of operation is particularly vulnerable to our attack.

There is a simple and efficient method that may increase the complexity of the present attacks (at least for the types of redundancies, used in our examples).

Seeing that a combination of a cipher with ASCII encoded English is dangerous, one can perform a fixed random permutation of the ASCII table. Before the encryption, each plaintext byte is to be substituted for a new, permuted value. Although due to the structure of the language, there still will be many blocks differing only in one symbol, the probability of one-bit differences will decrease considerably. Another method is to use error-correcting code, and let codewords represent possible plaintexts. This way one can eliminate all the low Hamming weight differences in the plaintext. Note however, that adding redundancy to the plaintext must be checked with the other existing attacks.

Acknowledgments

We would like to thank the Project Gutenberg [7], which was our source for printed texts in English and Eli Biham for helpful discussions.

References

1. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993. 72, 72, 72, 74, 74, 79, 79, 83
2. E. Biham, A. Shamir, *Differential Cryptanalysis of Feal and N-Hash*, Lecture Notes in Computer Science 547, Advances in Cryptology – EUROCRYPT'91, pp.1–17, Springer-Verlag, 1991. 72
3. E. Biham, A. Shamir, *Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer*, Lecture Notes in Computer Science 576, Advances in Cryptology – CRYPTO'91, pp.156–171, Springer-Verlag, 1992. 72
4. A. Biryukov, E. Kushilevitz, *Improved Cryptanalysis of RC5*, to appear, proceedings of EUROCRYPT'98. 72, 79, 80, 81, 81, 81, 81, 81, 82
5. P. F. Brown, V. J. Della Pietra, R. L. Mercer, S. A. Della Pietra, *An Estimate of an Upper Bound for the Entropy of English*, Computational Linguistics, Vol. 18, N. 1, pp.31–40, 1992. 76
6. T. M. Cover, R. King, *A Convergent Gambling Estimate of the Entropy of English*, IEEE Transactions on Information Theory, Vol. 24, N. 4, pp.413–421, 1978. 76
7. Project Gutenberg, http://www.promo.net/pg 85
8. 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. 73, 80, 81, 81
9. P. Kharn, P. Metzger, W. Simpson, *The ESP DES-CBC Transform*, ftp://ftp.isi.edu/in-notes/rfc1829.txt, 1995. 84
10. L. R. Knudsen, W. Meier, *Improved Differential Attacks on RC5*, Lecture Notes in Computer Science 1109, Advances in Cryptology – CRYPTO'96, pp.216–228, Springer-Verlag, 1996. 81
11. A. G. Konheim, *Cryptography: A Primer*, New York: John Wiley & Sons, 1981. 72
12. 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. 72

13. S. K. Langford, M. E. Hellman, *Differential-Linear Cryptanalysis*, Lecture Notes in Computer Science 839, Advances in Cryptology – CRYPTO'94, pp.17–25, Springer-Verlag, 1994. 73, 83, 83, 83, 83, 83, 83, 83
14. W. E. Madryga, *A High Performance Encryption Algorithm*, Computer Security: A Global Challenge, Elsevier Science Publishers, pp. 557–570, 1984. 73, 77, 77
15. 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. 83
16. S. Miyaguchi, A. Shiraishi, A. Shimizu, *Fast Data Encryption Algorithm Feal-8*, Review of Electrical Communications Laboratories, Vol. 36, N. 4, pp.433-437, 1988. 73
17. S. Miyaguchi, *Feal-N specifications*, NTT, 1989. 73
18. S. Murphy, *The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts*, Journal of Cryptology, pp.145–154, 1990. 72
19. National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, 1977. 73, 83
20. R. L. Rivest, *The RC5 Encryption Algorithm*, Lecture Notes in Computer Science 1008, Fast Software Encryption, pp.86–96, Springer-Verlag, 1994. 80, 80
21. B. Schneier, *Applied Cryptography Second Edition*, John Wiley & Sons, New York, NY, 1996. 73, 73, 77, 84, 84
22. A. A. Selçuk, *New Results in Linear Cryptanalysis of RC5*, to appear, proceedings of Fast Software Encryption 5, 1998. 73, 81
23. H. Shang, T. H. Merrettal, *Tries for Approximate String Matching*, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, N. 4, 1996. 76
24. C. Shannon, *Prediction and Entropy in Printed English*, Bell Systems Technical Journal, Vol. 30, N. 1, pp.50–64, 1951. 76
25. K. Shirriff, *Differential Cryptanalysis of Madryga*, unpublished manuscript, http://ftp.cs.berkeley.edu/ucb/sprite/www/papers/madryga.ps, October 1995. 78, 79

A An Example of one-bit English plaintext differences

In this Appendix we present a short study of ASCII encoded English difference behavior. We feel however, that this subject is worth a more thorough study.

In the left part of Figure 1 we present the distribution of weights of the differences in a sample of 10000 blocks. We compare several English texts of the same length: a sample from Dickens, a sample from Conan Doyle, a sample from Merkle's description of Khufu and Khafre block-ciphers and the 1st-order approximation to English (correct letter frequencies) with a sample of 10000 uniformly random blocks. Axis x marks the weights of the difference from 0 to 64. Axis y marks the number of pairs with difference weight x. One sees that all samples behave close to binomial distribution, however the mean of the random sample is around 32 while the mean of all English samples is about 20. This is explained by the fact, that English text consists mainly from ASCII encoded letters, and thus only five least significant bits of each byte (40 bits altogether) may vary, while the 24 most significant bits are constant most of the time. Thus, even before starting the analysis of the language itself, one sees that differences in ASCII encoded English are shifted to the low Hamming weight end. However in

the 10000-block Random sample there were no differences with Hamming weight less than 11. The English samples behave very similar to each other, with an exception of the 1st-order English, which is the highest of them. This is at the cost of reduced probability of differences in the low Hamming weight tail of the distribution. However in the main part of this paper we are interested exactly in these low Hamming weight differences (e.g. one, two and three-bit differences). This shows, that simple 1st-order English is a bad approximation, for the needs of our analysis. We observed that first order *word* approximation and especially second order word approximation, behave very close to the real English in the area of low Hamming weight differences.

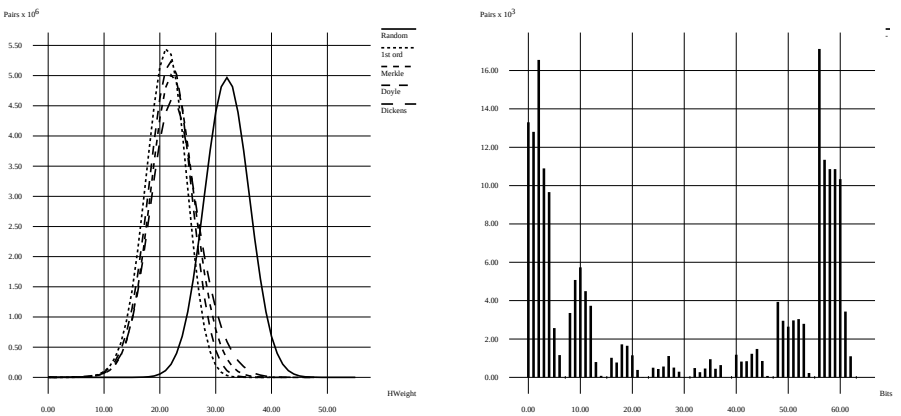

Fig. 1. Distribution of differences: (a) left figure compares English samples vs. Random sample; (b) right figure shows distribution of one-bit differences.

In the right part of Figure 1 we show the distribution of one-bit differences in English, calculated from a sample of about 2^{17} different blocks, taken from "David Copperfield" by Charles Dickens. Here we number the bit locations from 0 to 63 (from lsb to msb). As mentioned above, one sees that only about 40 out of 64 bit locations account for significant amounts of one-bit differences (due to ASCII encoding of English letters). One may see that $\frac{2}{3}$ of the differences occur in the four least significant bits of the first byte, and in the four least significant bits of the last byte of the block. The probability of the one-bit differences in the middle bits of the block is relatively low.

Below we present an example of English one-bit differences. These are the 85 one-bit differences obtained from a file of 5000 blocks, taken from the beginning of "The Hound of the Baskervilles" by Sir Arthur Conan Doyle. The same data was used in our ciphertext-only attack on Madryga in section 4.3.

```
' there h' ' there i' 0000000000000001  'by the l' 'by the m' 0000000000000001
'd left h' 'd left i' 0000000000000001  'er and r' 'er and s' 0000000000000001
'       ' '       "' 0000000000000002  ' health,' ' health.' 0000000000000002
' to be a' ' to be c' 0000000000000002  'hapter 1' 'hapter 3' 0000000000000002
'ntation,' 'ntation.' 0000000000000002  'of the m' 'of the o' 0000000000000002
'rles's d' 'rles's f' 0000000000000002  'ted.    ' 'ted.   "' 0000000000000002
'uld be m' 'uld be o' 0000000000000002  '     tha' '     the' 0000000000000004
' examina' ' examine' 0000000000000004  ' to my a' ' to my e' 0000000000000004
' which s' ' which w' 0000000000000004  ', with a' ', with e' 0000000000000004
'nding or' 'nding ov' 0000000000000004  '     tha' '     thi' 0000000000000008
' examina' ' examini' 0000000000000008  ' the sta' ' the sti' 0000000000000008
'e been a' 'e been i' 0000000000000008  'e last f' 'e last n' 0000000000000008
'e moor a' 'e moor i' 0000000000000008  ' to be c' ' to be s' 0000000000000010
' which c' ' which s' 0000000000000010  '"   "I d' '"   "I t' 0000000000000010
' have re' ' have se' 0000000000000100  'e the re' 'e the se' 0000000000000100
' stick, ' ' stick. ' 0000000000000200  'Holmes, ' 'Holmes. ' 0000000000000200
'      br' '      fr' 0000000000000400  '      do' '      lo' 0000000000000800
' was of ' ' was on ' 0000000000000800  '      do' '      to' 0000000000001000
' have be' ' have re' 0000000000001000  ' my infe' ' my inve' 0000000000001000
'?"   "Do' '?"   "To' 0000000000001000  ' the far' ' the nar' 0000000000080000
'"   "And' '"   "Ind' 0000000000080000  ' the Med' ' the med' 0000000000200000
'    The ' '    the ' 0000000020000000  'y of the' 'y on the' 0000000800000000
'mer.   "' 'mes.   "' 0000010000000000  ' on     ' ' of     ' 0000080000000000
'had you ' 'hat you ' 0000100000000000  'wed, and' 'wet, and' 0000100000000000
'er and s' 'es and s' 0001000000000000  ' as the ' ' is the ' 0008000000000000
'act     ' 'ast     ' 0010000000000000  'ed     ' 'et      ' 0010000000000000
'nd that ' 'nt that ' 0010000000000000  ' it may ' ' It may ' 0020000000000000
' the moo' ' The moo' 0020000000000000  ',      d' '-      d' 0100000000000000
'd at the' 'e at the' 0100000000000000  'ds with ' 'es with ' 0100000000000000
'e that t' 'd that t' 0100000000000000  'ht that ' 'it that ' 0100000000000000
'n that t' 'o that t' 0100000000000000  's.      ' 'r.      ' 0100000000000000
't have b' 'u have b' 0100000000000000  'u have b' 't have b' 0100000000000000
' Baskerv' '"Baskerv' 0200000000000000  ' that Dr' '"that Dr' 0200000000000000
'"I have ' ' I have ' 0200000000000000  'd in    ' 'f in    ' 0200000000000000
'e upon t' 'g upon t' 0200000000000000  'f the ch' 'd the ch' 0200000000000000
'ld count' 'nd count' 0200000000000000  'an      ' 'en      ' 0400000000000000
'en      ' 'an      ' 0400000000000000  'p upon t' 't upon t' 0400000000000000
's.      ' 'w.      ' 0400000000000000  've      ' 're      ' 0400000000000000
'and     ' 'ind     ' 0800000000000000  'f which ' 'n which ' 0800000000000000
'd such a' 't such a' 1000000000000000  'e have b' 'u have b' 1000000000000000
'He was a' 'he was a' 2000000000000000  'Were the' 'were the' 2000000000000000
'., from ' 'n, from ' 4000000000000000
```