Cryptanalysis of Ladder-DES

Eli Biham

Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel
Email: biham@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/

Abstract. Feistel ciphers are very common and very important in the design and analysis of blockciphers, especially due to four reasons: (1) Many (DES-like) ciphers are based on Feistel's construction. (2) Luby and Rackoff proved the security of a four-round Feistel construction when the round functions are random. (3) Recently several provably secure ciphers were suggested, which use other (assumed secure) ciphers as the round function. (4) Other such ciphers use this construction as attempts to improve the security of other ciphers (e.g., to improve the security of DES).

In this paper we cryptanalyze Ladder-DES, a four-rounds Feistel cipher using DES in the round function, and show that its security is smaller than expected.

1 Introduction

Feistel ciphers are very common and well known. In particular, Feistel's construction is used in the Data Encryption Standard[8], Lucifer[12] and their many successors (such as Feal[11,7], GDES[10], and many others). This construction was studied from the theoretical point of view by Luby and Rackoff[4], who concluded that four rounds suffice to prove its security when the round function is random. Many suggested cryptosystems were designed with this construction, using another cipher as the round function: some examples are Bear and Lion[2], Beast[6], and Ladder-DES[9]. Many other works had generalized or adopted the Feistel/Luby-Rackoff construction (some of them are [5,1]).

In this paper we cryptanalyze Ladder-DES, a four-round Feistel cipher, whose aim is to increase the security of DES, using DES in the round function. We describe the attack on Ladder-DES, and show that the security of Ladder-DES is smaller than expected. This attack can be generalized to many similar Feistel ciphers whose two parts (halves) are of the same size and whose round functions are permutations. This attack uses a novel application of the birthday paradox.

2 Description Of Ladder-DES

First we describe ladder-DES. It consists of four Feistel rounds, each applies DES as the round function. The four keys of the four DES applications serve as the key of Ladder-DES. The following figure describes Ladder-DES:

```
   A=L_0          B=L_1
    |      k1      |
    v  F_1 v       |
   XOR <- DES1-----|L_1
    |              |
    |      k2      |
    |      v  F_2  v
L_2 |---- DES2 -> XOR
    |              |
    |      k3      |
    v  F_3 v       |
   XOR <- DES3 ----|L_3
    |              |
    |      k4      |
    |      v  F_4  v
L_4 |---- DES4 -> XOR
    |              |
    v              v
   C=L_4          D=L_5
```

The rounds are numbered from 1 to 4, L_i is the 64-bit input of DES in round i, and F_i is the output of DES in round i. L_0 and L_5 are the left halves of the plaintext and the ciphertext, respectively.

3 A chosen Plaintext Attack

The main tool of the attack is the birthday paradox, which is used in a very unusual way. Usually the birthday paradox is used to find a collision (two equal values) in a set of $\sqrt{n}$ random values. Our attack uses the birthday paradox to identify whether given values are calculated by a pseudo-random function or a pseudo-random permutation. In the first case, the birthday paradox predicts the existence of a collision given $\sqrt{n}$ values. In the later case, collision cannot occur even given all the n values. In our attack the key is found only when we identify that there is no collision. This is the only use of the birthday paradox in this way which we are aware of.

In the attack we choose 2^{36} plaintexts of the form (A,B) where B is your favorite (or random) 64-bit fixed constant, and A gets 2^{36} different 64-bit values.

In this context, L_1 is fixed in all the 2^{36} encryption runs, and L_0 gets 2^{36} different values in the 2^{36} encryption runs. for simplicity, we will call this property of L_0 a *permutation* (i.e., there is no collision; this property holds even in all the 2^{64} possible plaintexts with a fixed B). F_1 is L_1 encrypted under a fixed (but unknown) key, thus it is fixed in all the runs. A permutation XORed with a fixed value is also a permutation, and thus L_2 is a permutation, and F_2 is also a permutation. L_1 is fixed, and thus L_3 and F_3 are permutations as well. L_4 is not a permutation: it is a mix of two permutation, which behaves like a pseudo-random function.

Our aim is to find the permutation in L_3, given the ciphertext $(C, D) = (L_4, L_5)$.

When the 2^{36} ciphertexts are given, we try all the 2^{56} possible keys k4, one by one, using the following algorithm:

```
for each possible key k4 (in range 0 to $2^{56} - 1$)
  for each ciphertext $(C_i, D_i)$ $(i = 1, \ldots, 2^{36})$
    compute $L_3^{i,k4} = \mathrm{DES}_{k4}^{-1}(C_i) \oplus D_i$
    if a collision occurs (i.e., $L_3^{i,k4} = L_3^{j,k4}$ for some $j < i$)
      conclude that k4 is wrong, and try next k4
  end for
  – We reach here only when k4 is the right key!!
  conclude that k4 is the key
end for
```

When we decrypt the ciphertext with a wrong candidate for k4, the one-round decryption function (that computes L_3) is expected to behave like a random function. For each candidate key we decrypt the fourth round of all the 2^{36} ciphertexts, or till we get two equal values of L_3. If two equal values of L_3 are found, L_3 is not a permutation, and thus the candidate for k4 is not the key. In average about 2^{32} candidates are required to discard a wrong candidate. The real value of k4 does not imply any collision of L_3 even if all the 2^{64} possible ciphertexts are decrypted by one round, and thus it can be identified.

Later, the values of k3 can be found with the same data, because L_2 is a permutation, but if a wrong value of k3 is used during decryption, the resultant value of L_2 would not be a permutation. A simpler method to find k3 takes two of the plaintexts, compute the difference of the output of F_3 as the XOR of the differences of L_0 and of L_4. Then, it searches exhaustively for the key k3 which satisfies this difference. False alarms can be identified and discarded using a third plaintext.

The remaining key k1 and k2 can then be found by exhaustive search, which would require only one plaintext/ciphertext sample, taken from the data we already have. Each of k1 and k2 would be found with complexity 2^{56}, after k3 and k4 are known.

Some notes on the birthday paradox:

About $\sqrt{2 \cdot \log_e 2 \cdot 2^{64}} = 1.177 \cdot 2^{32} \approx 2^{32}$ random values are required to find two equal 64-bit values with probability 1/2, and $\sqrt{2 \cdot \log_e 2 \cdot 2^{64} \cdot m} = \sqrt{m} \cdot 1.177 \cdot 2^{32}$ random values are required to find such a pair with probability $1 - 2^{-m}$. In particular, in the interesting case when m=56, and we have an error probability of 2^{-56}, we need only $\sqrt{56} \cdot 1.177 \cdot 2^{32} = 8.1 \cdot 2^{32} = 2^{35}$ values to identify whether they are the result of a random function or a permutation. Thus given 2^{35} ciphertexts we can identify the key almost without mistakes, and with 2^{36} ciphertexts we can be almost ensured to have no mistakes (error probability about $e^{-128} = 2^{-185}$, which causes probability 2^{-129} for a false alarm). In average we need only 2^{32} trys, and only in a few cases we need more than 2^{34} trys for a key (except for the real key).

Complexity:

We try 2^{56} keys, for each we calculate in average 2^{32} single DES's before we discard it. Thus our complexity is about 2^{88} to find k4. The complexity to find k3 is 2^{57}. k1 and k2 can then be found with complexity 2^{56} each. Thus, the total complexity is about 2^{88}. Only 2^{35}–2^{36} chosen plaintexts are required. This complexity is much less than the expected 2^{112} complexity of a meet in the middle attack[3], which was claimed for this cryptosystem.

4 A Known Plaintext Attack

The complexity and number of required plaintexts of this known plaintext attack are about the same as of the chosen plaintext attack (2^{90} complexity, 2^{36} known plaintexts). The amount of required memory is however much larger than the chosen plaintext attack requires.

When the plaintexts/ciphertexts are given, we try all the 2^{56} possible keys k4 one by one. For each k4 we search for collisions in L_3 as in the chosen ciphertext attack, but this time collisions should occur for all the keys. We keep the first two collisions we find (in lexicographic order of the index of the plaintexts) in a table (of size $2 \cdot 2^{56}$, each keeps only the index of the pair). Similarly, we try all the values of k1 and search for collisions in F_3 ($F_3 = A \oplus C \oplus DES_{K1}(B)$). Clearly, a pair collides in L_3 *iff* it collides in F_3. We then search for pairs in the first table which have the same indices as pairs in the second table: only such pairs can suggest the right k1 and k4. It is expected that only the right k1 and k4 will collide in two same pairs (average of 2^{-16} false alarms; additional safety margins can be added by keeping three colliding pairs in the tables, which reduces the rate of false alarms to 2^{-80}).

The remaining k2 and k3 are easily found later with complexity 2^{57}.

This attack requires 2^{36} known plaintext, 2^{90} work (in average to find the first two colliding pairs for each key) and requires 2^{57} space (about 2^{60}–2^{61} bytes).

5 Acknowledgements

We are very grateful to Don Coppersmith for his various comments which improved the results of this paper. This research was supported by the fund for the promotion of research at the Technion.

References

1. William Aiello, Ramarathnam Venkatesan, *Foiling Birthday Attacks in Length-Doubling Transformations*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'96, pp. 307–320, 1996.
2. Ross Anderson, Eli Biham, *Two Practical and Provably Secure Block Ciphers: BEAR and LION*, proceedings of Fast Software Encryption, Cambridge, Lecture Notes in Computer Science, pp. 113–120, 1996.
3. W. Diffie, M. E. Hellman, *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, Computer, Vol. 10, No. 6, pp. 74–84, June 1977.
4. M. Luby, C. Rackoff, *How to construct pseudorandom permutations from pseduorandom functions*, SIAM Journal on Computing, Vol. 17, No. 2, pp. 373–386, 1988.
5. Stefan Lucks, Faster Luby-Rackoff Ciphers, proceedings of Fast Software Encryption, Cambridge, Lecture Notes in Computer Science, pp. 189–203, 1996.
6. Stefan Lucks, BEAST: A Fast Block Cipher for Arbitrary Blocksizes, 1996.
7. Shoji Miyaguchi, Akira Shiraishi, Akihiro Shimizu, *Fast Data Encryption Algorithm FEAL-8*, Review of electrical communications laboratories, Vol. 36, No. 4, pp. 433–437, 1988.
8. National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, January 1977.
9. Terry Ritter, *Ladder-DES: A Proposed Candidate to Replace DES*, appeared in the Usenet newsgroup sci.crypt, February 1994.
10. Ingrid Schaumuller-Bichl, *On the Design and Analysis of New Cipher Systems Related to the DES*, technical report, Linz university, 1983.
11. Akihiro Shimizu, Shoji Miyaguchi, *Fast Data Encryption Algorithm FEAL*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'87, pp. 267–278, 1987.
12. Arthur Sorkin, *Lucifer, a Cryptographic Algorithm*, Cryptologia, Vol. 8, No. 1, pp. 22–41, January 1984.