Available online at www.elixirpublishers.com (Elixir International Journal)

Applied Mathematics

Elixir Appl. Math. 38 (2011) 4414-4417

ISSN: 2229-712X

Linear and differential cryptanalysis of DES

C. Sajeev and C. Suyambulingom
Sathyabama University, Chennai.

ARTICLE INFO

Article history:
Received: 22 August 2011;
Received in revised form:
26 August 2011;
Accepted: 31 August 2011;

Keywords
Cryptography,
DES,
AES,
Symmetric Key.

ABSTRACT
The Data Encryption Standard (DES), a symmetric-key cryptosystem, developed for United States government was intended for use by the general public. It has been officially accepted as a cryptographic standard in United States and other countries. The DES is also known as the Data Encryption Algorithm (DEA) by ANSI and DEA-1 by the ISO. It has been a worldwide standard for 30 years. Many hardware and software system have been designed with the DES. Although it is showing signs of old age, it has hold up remarkably well against years of cryptanalysis and it is still secure against all but possibly the most powerful adversaries. In this paper we begin by describing DES then describe and analyze attacks against DES.

Introduction

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption. The DES enjoys widespread use. It has also been the subject of much controversy concerning how secure the DES is. To appreciate the nature of the controversy, let us quickly review the history of the DES.

In the late 1960s, IBM set up a research project in computer cryptography led by Horst Feistel. The project concluded in 1971 with the development of an algorithm with the designation LUCIFER, which was sold to Lloyd's of London for use in a cash-dispensing system, also developed by IBM. LUCIFER is a Feistel block cipher that operates on blocks of 64 bits, using a key size of 128 bits. Because of the promising results produced by the LUCIFER project, IBM embarked on an effort to develop a marketable commercial encryption product that ideally could be implemented on a single chip. The effort was headed by Walter Tuchman and Carl Meyer, and it involved not only IBM researchers but also outside consultants and technical advice from NSA.

The outcome of this effort was a refined version of LUCIFER that was more resistant to cryptanalysis but that had a reduced key size of 56 bits, to fit on a single chip.

In 1973, the National Bureau of Standards (NBS) issued a request for proposals for a national cipher standard. IBM submitted the results of its Tuchman Meyer project. This was by far the best algorithm proposed and was adopted in 1977 as the Data Encryption Standard.

Before its adoption as a standard, the proposed DES was subjected to intense criticism, which has not subsided to this day. Two areas drew the critics' fire. First, the key length in IBM's original LUCIFER algorithm was 128 bits, but that of the proposed system was only 56 bits, an enormous reduction in key size of 72 bits. Critics feared that this key length was too short to withstand brute-force attacks. The second area of concern was that the design criteria for the internal structure of DES, the S-boxes, were classified. Thus, users could not be sure that the internal structure of DES was free of any hidden weak points that would enable NSA to decipher messages without benefit of the key. Subsequent events, particularly the recent work on differential cryptanalysis, seem to indicate that DES has a very strong internal structure. Furthermore, according to IBM participants, the only changes that were made to the proposal were changes to the S-boxes, suggested by NSA, that removed vulnerabilities identified in the course of the evaluation process.

Whatever the merits of the case, DES has flourished and is widely used, especially in financial applications. In 1994, NIST reaffirmed DES for federal use for another five years; NIST recommended the use of DES for applications other than the protection of classified information. In 1999, NIST issued a new version of its standard (FIPS PUB 46-3) that indicated that DES should only be used for legacy systems and that triple DES (which in essence involves repeating the DES algorithm three times on the plaintext using two or three different keys to produce the ciphertext) be used. Because the underlying encryption and decryption algorithms are the same for DES and triple DES, it remains important to understand the DES cipher.

Description of DES

We give here a brief description of DES, primarily to establish terminology. We do not provide the various tables that are necessary for a full description of the standard; for those, see [2] or [3].

We wish to encipher a 64-bit plaintext *message block* m under the 56-bit *key k*, to produce a 64-bit ciphertext message block $c = E_k(m)$. (The sizes of message blocks and keys, 64 bits and 56 bits respectively, are specified in the standard.) Decipherment or recovering plaintext from ciphertext, is denoted $m = D_k(c)$.

The plaintext message block m is subjected to an initial *permutation IP*, and the result is broken into two 32-bit *message halves*, m_0, and m_1. Intermediate message halves m_2 …m_{17} are

Tele:
E-mail addresses: painkulamcsajeev@yahoo.co.in

then created in sixteen *rounds*, according to the procedure described below. Finally, the 64-bit ciphertext c is generated by applying the inverse permutation IP^{-1} to the two message halves m_{17}, m_{16}.

The plaintext message halves and intermediate message halves m_0, m_1, m_2, … m_{I7} are related as follows:

$m_{i+l} = m_{i-l} \oplus f(k_{(i)}, m_i)$ i = 1, 2, , 16.

Here *k* is the secret 56-bit key, and *i* is the number of the round (from 1 through 16). Also, $k_{(i)}$ is a selection of 48 bits from the 56 bits of *k;* this selection, or *key schedule* (Described in [2]), depends on the round number, *i*. The symbol ⊕ denotes bit-by-bit "exclusive OR" (addition modulo 2), which we call "XOR" in the text.

Now we describe the function *f*. There are eight *S-boxes*, S_1, …, S_8, described in the standard. Each S-box is a table lookup, using six bits as input and providing four bits as output. For each S-box, say S_j, six consecutive bits are selected from the 48 bits of namely bits 6j - 5, 6j - 4, … , 6j. *Also*, six consecutive bits are selected from m_i, namely bits 4j - 4, 4j - 3, … , 4j + 1 (mod 32). The "mod 32" is shorthand for the convention that for j = 1 the bits are 32, 1, 2, 3, 4, 5, and for j = 8 the bits are 28, 29, 30, 31, 32, 1. Two adjacent S-boxes share two message bits; for instance, S_1 uses message bits 32, 1, 2, 3, 4, 5, while S_2 uses message bits 4, 5, 6, 7, 8, 9, and they share bits 4 and 5. (Key bits are not shared among S-boxes on one round.) S_8 and S_1 are considered to be "adjacent" because they share message bits 32 and 1.

The six key bits and the six message bits are XORed together bitwise, and the resulting six bits are used as input for a table lookup. That is, the six inputs to S-box S_j at round *i* are

$m_i[4j - 4] \oplus k_{(i)}[6j - 5]$,

$m_i[4j - 3] \oplus k_{(i)}[6j - 4]$,

…

$m_i[4j + 1] \oplus k_{(i)}[6j]$,

or, written another way,

$m_i[4j - 4, 4j - 3, 4j - 2, 4j - 1, 4j, 4j + 1]$

$\oplus k_{(i)}[6j - 5, 6j - 4, 6j - 3, 6j - 2, 6j - 1, 6j]$.

Each of the eight S-boxes implements a different table, each with 2^6 entries of four bits each. These tables are described in the standard.

The eight S-boxes together put out 8 × 4 = 32 bits. These bits are permuted according to a permutation P that is fixed for all rounds *i*. The resulting 32-bit quantity is the value of $f(k_{(i)}, m_i)$.

In summary, the 64-bit message undergoes a permutation IP to produce two 32-bit message halves m_0 and m_1. Then we compute the 32-bit quantity $f(k_{(1)}, m_1)$, and XOR that quantity with m_0 to produce m_2. We use this new quantity m_2 to compute $f(k_{(2)}, m_2)$, and XOR that quantity with m_1 to produce m_3. We continue in a like fashion until m_{16} and m_{17} have been computed. These *two* message halves are interchanged and then subjected to the permutation IP^{-1}, to produce the ciphertext c.

Decryption is easily accomplished by a user in possession of the same key *k*. First, one applies the permutation IP to c to produce the message halves m_{17}, m_{16}. Next, one computes f(k(16), m16) and XORs that quantity with m_{17} to recover m_{16}. Recalling that

$m_{17} = m_{15} \oplus f(k_{(16)}, m_{16})$,

we have

$m_{17} \oplus f(k_{(16)}, m_{16})$

$= [m_{15} \oplus f(k_{(16)}, m_{16})] \oplus f(k_{(16)}, m_{16})$

$= m_{15}$,

because of the identity $(A \oplus B) \oplus B = A$. Similarly, one computes $m_{14} = m_{16} \oplus f(k_{(15)}, m_{15})$ and continues in like fashion until one has computed m_1 and m_0. Applying IP^{-1} to the pair (m_0, m_1), one recovers the plaintext message m.

Any function could be used in place of f, and we would still have a reversible encryption method. Different choices of f, however, yield different levels of security in the overall algorithm. The function f used in DES was designed to provide a high level of security.

The strength of DES

The Use of 56-Bit Keys

With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16}. Thus, on the face of it, a brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years (see Table 1) to break the cipher.

However, the assumption of one encryption per microsecond is overly conservative. As far back as 1977, Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. This would bring the average search time down to about 10 hours. The authors estimated that the cost would be about $20 million in 1977 dollars.

It is important to note that there is more to a key-search attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. If the message is just plain text in English, then the result pops out easily, although the task of recognizing English would have to be automated. If the text message has been compressed before encryption, then recognition is more difficult. And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate. Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed. The EFF approach addresses this issue as well and introduces some automated techniques that would be effective in many contexts.

DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker. And, of course, hardware prices will continue to drop as speeds increase, making DES virtually worthless.

Differential cryptanalysis

Differential cryptanalysis was not reported in the open literature until 1990. The first published effort appears to have been the cryptanalysis of a block cipher called FEAL by Murphy. This was followed by a number of papers by Biham and Shamir, who demonstrated this form of attack on a variety of encryption algorithms and hash functions; their results are summarized in "Biham, E., and Shamir, A. Differential Cryptanalysis of the Data Encryption Standard. New York: Springer-Verlag, 1993."

The most publicized results for this approach have been those that have application to DES. Differential cryptanalysis is the first published attack that is capable of breaking DES in less

than 2^{55} complexity. The scheme can successfully cryptanalyze DES with an effort on the order of 2^{47} encryptions, requiring 2^{47} chosen plaintexts. Although 2^{47} is certainly significantly less than 2^{55} the need for the adversary to find 2^{47} chosen plaintexts makes this attack of only theoretical interest.

Although differential cryptanalysis is a powerful tool, it does not do very well against DES. The reason, according to a member of the IBM team that designed DES "Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." IBM Journal of Research and Development, May 1994", is that differential cryptanalysis was known to the team as early as 1974. The need to strengthen DES against attacks using differential cryptanalysis played a large part in the design of the S-boxes and the permutation P. As evidence of the impact of these changes, consider these comparable results reported in "Biham, E., and Shamir, A. Differential Cryptanalysis of the Data Encryption Standard. New York: Springer-Verlag, 1993." Differential cryptanalysis of an eight-round LUCIFER algorithm requires only 256 chosen plaintexts, whereas an attack on an eight-round version of DES requires 2^{14} chosen plaintexts.

Differential Cryptanalysis Attack

The differential cryptanalysis attack is complex. The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block. Here, we provide a brief overview so that you can get the flavor of the attack.

We begin with a change in notation for DES. Consider the original plaintext block m to consist of two halves m_0, m_1. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round. So, at each round, only one new 32-bit block is created. If we label each new block m_1($2 \le i \le 17$), then the intermediate message halves are related as follows:

$$m_{i+1} = m_{i-1} \oplus f(K_{(i)}, m_i), \quad i = 1, 2, ..., 16$$

In differential cryptanalysis, we start with two messages, m and m', with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $m_i = m_i \oplus m_i'$ Then we have:

$$\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1}$$
$$= [m_{i-1} \oplus f(k_{(i)}, m_i)] \oplus [m'_{i-1} \oplus f(k_{(i)}, m'_i)]$$
$$= \Delta m_{i-1} \oplus [f(k_{(i)}, m_i) \oplus f(k_{(i)}, m'_i)]$$

Now, suppose that many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used. To put this more precisely, let us say that X may cause Y with probability p, if for a fraction p of the pairs in which the input XOR is X, the output XOR equals Y. We want to suppose that there are a number of values of X that have high probability of causing a particular output difference. Therefore, if we know Δm_{i-1} and Δm_i with high probability, then we know Δm_{i+1} with high probability. Furthermore, if a number of such differences are determined, it is feasible to determine the subkey used in the function f.

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages m and m' with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the ciphertext. Actually, there are two probable patterns of differences for the two 32-bit halves: ($\Delta m_{17} \| m_{16}$). Next, we submit m and m' for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match,

$E(k, m) \oplus E(k, m') = (\Delta m_{17} \| m_{16})$ then we suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, we can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

Figure 1, based on a figure in "Biham, E., and Shamir, A. Differential Cryptanalysis of the Data Encryption Standard. New York: Springer-Verlag, 1993", illustrates the propagation of differences through three rounds of DES. The probabilities shown on the right refer to the probability that a given set of intermediate differences will appear as a function of the input differences. Overall, after three rounds the probability that the output difference is as shown is equal to $0.25 \times 1 \times 0.25 = 0.0625$.

Linear Cryptanalysis

Another development is linear cryptanalysis, described in "Matsui, M. "Linear Cryptanalysis Method for DES Cipher." Proceedings, EUROCRYPT '93, 1993; published by Springer-Verlag" This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given 2^{43} known plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by other groups to validate the linear cryptanalytic approach.

For a cipher with n-bit plaintext and ciphertext blocks and an m-bit key, let the plaintext block be labeled P[1], ... P[n], the cipher text block C[1], ... C[n], and the key K[1], ... K[m]. Then define

A[i, j, ..., k] = A[i] ⊕ A[j] ⊕ ... ⊕ A[k]

The objective of linear cryptanalysis is to find an effective linear equation of the form:

P[□1, □2, ..., □a] ⊕ C[□1, □2, ..., □b] = K[□1, □2, ..., □c]

(where x = 0 or 1; $1 \le a, b \le n$, $1 \le c \le m$, and where the □, □ and □ terms represent fixed, unique bit locations) that holds with probability $p \ne 0.5$. The further p is from 0.5, the more effective the equation.

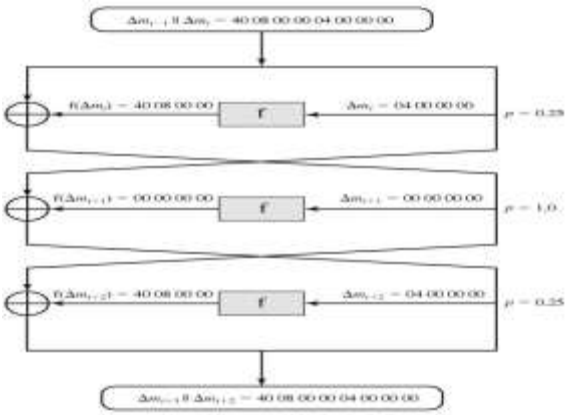

Figure 1. Differential Propagation through Three Round of DES (numbers in hexadecimal)

Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext-ciphertext pairs. If the result is 0 more than half the time, assume K[□1, □2, ..., □c] = 0. If it is 1 most of the time, assume K[□1, □2, ..., □c] = 1. This

gives us a linear equation on the key bits. Try to get more such relations so that we can solve for the key bits. Because we are dealing with linear equations, the problem can be approached one round of the cipher at a time, with the results combined.

Conclusion

The prime concern with DES has been its vulnerability to brute-force attack because of its relatively short (56 bits) key length. However, there has also been interest in finding cryptanalytic attacks on DES. With the increasing popularity of block ciphers with longer key lengths, including triple DES, brute-force attacks have become increasingly impractical. Thus, there has been increased emphasis on cryptanalytic attacks on DES and other symmetric block ciphers. In this paper, we analyzed the two most powerful and promising approaches: differential cryptanalysis and linear cryptanalysis.

Reference

[1] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," Lecture Notes in Computer Science: Advances in Cryptology-Proceedings of CRYPTO '90, Springer-Verlag, 1990, pp. 2-21.

[2] "Data Encryption Standard," Federal InformationProcessing *Standards Publication No. 46*, National Bureau of Standards, January 15, 1977.

[3] C. H. Meyer and S. M. Matyas, Cryptography: A New Dimension in Computer Data Security, John Wiley & Sons, Inc., New York, 1982.

[4] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," J. Cryptol. 4, 3-72 (1991).

[5] Electronic Frontier Foundation. Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design. Sebastopol, CA: O'Reilly, 1998.

[6] Diffie, W., and Hellman, M. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard." Computer, June 1977.

[7] Murphy, S. "The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts." Journal of Cryptology, No. 3, 1990.

[8] Biham, E., and Shamir, A. Differential Cryptanalysis of the Data Encryption Standard. New York: Springer-Verlag, 1993.

[9] Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." IBM Journal of Research and Development, May 1994.

[10] Matsui, M. "Linear Cryptanalysis Method for DES Cipher." Proceedings, EUROCRYPT '93, 1993; published by Springer-Verlag.

[11] Sinkov, A. Elementary Cryptanalysis: A Mathematical Approach. Washington, DC: The Mathematical Association of America, 1966.

[12] Feistel, H. "Cryptography and Computer Privacy." Scientific American, May 1973.

[13] Shannon, C. "Communication Theory of Secrecy Systems." Bell Systems Technical Journal, No. 4, 1949.

[14] Szor, P. The Art of Computer Virus Research and Defense. Reading, MA: Addison-Wesley, 2005.

[15] Konheim, A. Cryptography: A Primer. New York: Wiley, 1981.

[16] Lewand, R. Cryptological Mathematics. Washington, DC: Mathematical Association of America, 2000.

[17] Schaefer, E.; and Wedig, S. "A Simplified AES Algorithm and Its Linear and Differential Cryptanalyses." Cryptologia, April 2003.

[18] Nechvatal, J., et al. Report on the Development of the Advanced Encryption Standard. National Institute of Standards and Technology. October 2, 2000.

Table 1. Average Time Required for Exhaustive Key Search

Key size (bits)	Number of alternative keys		Time required at 1 decryption/ms		Time required at 10^6 decryption/ms
32	2^{32}	$= 4.3\times10^{9}$	2^{31} ms	= 35.8 minutes	2.15 milliseconds
56	2^{56}	$= 7.2\times10^{16}$	2^{55} ms	= 1142 years	10.01 hours
128	2^{128}	$= 3.4\times10^{38}$	2^{127} ms	$= 5.4\times10^{24}$ years	5.4×10^{18} years
168	2^{168}	$= 3.7\times10^{50}$	2^{167} ms	$= 5.9\times10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	26!	$= 4\times10^{26}$	2×10^{26} ms	$= 6.4\times10^{12}$ years	6.4×10^{6} years