Classification of Ciphers

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Pooja Maheshwari

to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

February 2001

Certificate

This is to certify that the work contained in the thesis entitled *Classification of Ciphers*, by Pooja Maheshwari, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

February 2001

(Dr. Manindra Agrawal)

Department of Computer Science & Engineering,

Indian Institute of Technology,

Kanpur.

Abstract

In cryptanalysis of an unknown cipher first step is to identify the cipher and then to break it. To identify the cipher we need to classify them. Classifying ciphers means identifying the cipher, which has resulted the given ciphertext encrypted by that unknown cipher.

In this thesis, classification of classical ciphers was done with very good accuracy. For classification of modern ciphers like DES and IDEA, several schemes have been examined. Slightly positive results were obtained for modern ciphers classification.

Acknowledgements

I would like to express deep gratitude towards my thesis supervisor Dr. Manindra Agrawal, for his excellent guidance and help during my thesis work. He was always available to help, guide, and encourage me. He has always been very patient and understanding.

I am also thankful to all the faculty members of Computer Science & Engineering Department for their encouragement, which has brought me to this competent stage. I extend my thanks to the technical staff of the department for their cooperation and help.

I thank to all my friends, specially Nameeta, Vibha, and Bhoomika, for patiently listening to my thesis problems and making my stay memorable one at IIT-Kanpur without which I would not have cheerfully completed this task.

Finally I would like to mention my beloved parents for their love and affection. It is their trust and expectations that drives me always.

Contents

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Objective

In breaking an unknown cipher first step is to identify the cipher and then to break it. By *Classification of Ciphers* we mean that given an encrypted text from an unknown cipher, identify the cipher. Very little published literature is available on this problem.

We got the idea of working on this problem from a website "www.dgsciences.com/codeclas.htm". The site includes a set of ciphers like public key euler function, public key knapsack cipher, etc. which are being classified by them and whose limited distribution is available for beta testing. But they did not include ciphers like DES and IDEA. [1]

1.2 Classical Ciphers

We started our work with a set of classical ciphers namely Substitution Cipher, Permutation Cipher, Combination of Substitution and Permutation Cipher, and Vigenere Cipher. In case of these classical Ciphers the main attack is frequency distribution. We have been able to classify these classical ciphers with very good accuracy.

1.3 Modern Ciphers

We then tried classification of modern ciphers like DES and IDEA. We chose a simpler problem of differentiating DES stream from IDEA stream. But even this task becomes very difficult, as these ciphers do not possess properties that could be analyzed easily. In our several approaches like randomness tests, use of XOR operations, use of threshold functions, etc. DES and IDEA exhibit same behavior. Although for certain threshold functions, we got encouraging results that, when developed further, may lead to classify DES from IDEA.

1.4 Organization

In chapter 2, we discuss classical ciphers, technique used for their classification and, results obtained.

In chapter 3, we discuss modern ciphers like DES and IDEA, several attempts for their classification, and results obtained.

Chapter 4, ends with the final conclusion and directions for future work.

CHAPTER 2

Classification of Classical Ciphers

2.1 Description of Classical Ciphers: [2] [3]

2.1.1 Permutation Cipher:

Permutation Cipher alters position of plaintext characters by rearranging them. Formal definition of permutation cipher is as follows:

"Let m be some fixed positive integer. Let P = C = $(Z_{26})^m$ and let K consist of all permutations of {1,…,m}. For a key (i.e. a permutation) $\prod$, we define

$$e_{\prod}(x_1,\ldots x_m) = (x_{\prod(1)},\ldots,x_{\prod(m)})$$

and

$$d_{\prod}(y_1,\ldots y_m) = (y_{\prod^{-1}(1)},\ldots,y_{\prod^{-1}(m)}),$$

where $\prod^{-1}$ is the inverse permutation to $\prod$. "

In ciphertext resulted by permutation cipher each character represents itself.

Example:

We fix a permutation $\prod$ of m letters.

Suppose m = 4 and the key is the following permutation $\prod$:

1	2	3	4
?	?	?	?
3	4	2	1

Then the inverse permutation $\prod^{-1}$ is the following:

1	2	3	4
?	?	?	?
4	3	1	2

Now, suppose plaintext is: "ENCRYPTION"

We'll group the plaintext into groups of four letters each:

ENCR | YPTI | ON

Now we'll permute each group four letters according to permutation $\prod$ resulting in ciphertext: "CRNETIPYNO".

The ciphertext can be decrypted in a similar way, using the inverse permutation $\prod^{-1}$.

2.1.2 Substitution Cipher:

In substitution cipher each letter is mapped to another letter via a fixed mapping.

Formal definition of substitution cipher is as follows:

"Let P = C = Z_{26}. K consists of all possible permutations of the 26 symbols 0, 1, …, 25. For each permutation ? e K, define

$$e_{?}(x) = ?(x),$$

and define

$$d_{?}(y) = ?^{-1}(y),$$

where $?^{-1}$ is the inverse permutation to ? . "

So, in ciphertext resulted by substitution cipher each letter is substituted by some other letter.

Example:

Suppose ∏ is as follows:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
R	S	A	E	M	G	Y	F	T	B	U	Z	K

N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
C	V	W	J	L	H	X	D	Q	I	O	N	P

Thus, $e_{\Pi}(A)=R$, $e_{\Pi}(B)=S$, etc. The decryption function is inverse permutation. That is $d_{\Pi}(R)=A$, $d_{\Pi}(s)=B$, etc.

Now, suppose plaintext is: " AN ENCRYPTED MESSAGE"

Then ciphertext will be: "RCMCALNWXMEKMHHRYM".

2.1.3 Combination of Permutation and Substitution Cipher:

Sometimes permutation and substitution ciphers are applied together. In this, both substitution and permutation ciphers are applied one by one, in any order. Formal definition of this cipher could be stated as follows:

"Let C = P = $(Z_{26})^m$. K consists of all possible permutations of the 26 symbols 0, 1, …, 25, and K1 consists of all permutations of {1, …, m}. Where m is some fixed positive integer. Then for each ? e K and ? 1 e K1, we define

$e_{?}(x) = ?(x)$

$e_{?1}(x_1, \ldots, x_m) = (e_{?} x_{?1}(1), \ldots, e_{?} x_{?1}(m))$

and

$d_{?1}(?^{-1}y^1, \ldots, ?^{-1}y_m) = (y_{?1}{}^{-1}(1), \ldots, y_{?1}{}^{-1}(m))$

$d_{?}(y) = ?^{-1}(y)$

where $?^{-1}$ and $?1^{-1}$ are inverse permutation to ? and ? 1 respectively."

So in resultant ciphertext each letter represents some other letter and at a different position.

Example:

Suppose m=4, ? is same as ? defined in Substitution Cipher example, and ? 1 is same as ? defined in Permutation Cipher example.

Suppose plaintext is: " AN ENCRYPTED MESSAGE"

Then $e_{?}(x)$ = "RCMCALNWXMEKMHHRYM"

Breaking it into groups of four letters each:

RCMC | ALNW | XMEK | MHHR | YM

And ciphertext is = $e_{?1}(ex_1, \ldots, ex_m)$

= " MCCRNWLAEKMXHRHMMY".

2.1.4 Vigenere Cipher: [4]

The Vigenere Cipher is a polyalphabetic cipher. In a polyalphabetic cipher use of two or more cipher alphabet is involved. So instead of one-to-one relationship between each letter and its substitutes, there is a one-to-many relationship between each letter and its substitutes. It operates on blocks of characters. A secret key word of length m is added to blocks of m letters. Formally Vigenere Cipher is defined as follows:

"Let m be some fixed positive integer. Define P = C = K = $(Z_{26})^m$.

For a key $K = (k_1, k_2, \ldots, k_m)$, we define

$e_K(x_1, x_2, \ldots, x_m) = (x_1+k_1, x_2+k_2, \ldots, x_m+k_m)$

and

$d_K(y_1, y_2, \ldots, y_m) = (y_1-k_1, y_2-k_2, \ldots, y_m-k_m)$

where all operations are performed in Z_{26}".

Example:

To encrypt using this cipher first we've to choose some keyword. So let m=3, and the keyword is "RED". Suppose plaintext is "ENCRYPTION". Now write down the keyword above plaintext repeatedly as follows:

Keyword: RED RED RED R

Plaintext: ENC RYP TIO N

Numerical equivalent of keyword K is = (17, 4, 3)

Numerical equivalent of plaintext symbols is:

(4, 13, 2), (17, 24, 15), (19, 8, 14), (13)

So, numerical equivalent of ciphertext is: { $c_i = (p_i + k_i) \bmod 26$ }

(21, 17, 5), (8, 2, 18), (10, 12, 17), (4)

So, the ciphertext is: "VRFICSKMRE".

2.2 Technique Used for Classification:

The program first checks for possibility of permutation cipher. If it is not permutation cipher then it finds whether the input ciphertext is encrypted using substitution cipher or combination of permutation and substitution cipher. If it doesn't pass any test till now then it checks for possibility of vigenere cipher. If it doesn't pass vigenere cipher test also then input cipher is reported as "Non-classical Cipher".

As in permutation cipher each letter represents itself, so in ciphertext single letter frequency distribution remains same as in plaintext. So this single letter frequency distribution of ciphertext is similar to single letter frequency distribution of normal English text. So the cost function for permutation cipher is as follows:

$$\text{Cost} = \sum_{i=0}^{25} | K^u(i) - D^u(i) | \qquad \ldots (1)$$

where $K^u(i)$ contains relative frequency of letter 'i' in normal English text[5], and $D^u(i)$ contains relative frequency of letter 'i' in given ciphertext. Here u is used to represent unigram (or single letter) frequencies. In $K^u(i)$ and $D^u(i)$ frequencies are stored in alphabetical order.

On given ciphertext, frequency analysis is performed and array D is computed. Then cost is computed using equation (1). If this cost comes out to be less than or equal to some tolerance value (say TOLRP) then the given ciphertext is encrypted by permutation cipher otherwise it is not encrypted by permutation cipher.

An experimentally good value of TOLRP is 40.0. This value is set by running the program on number of sample ciphertexts including both permutation and non-permutation ciphers.

If the ciphertext is not encrypted by permutation cipher then it checks for possibility of substitution or combination of permutation and substitution cipher. In substitution cipher, since each letter is substituted by another letter so single letter relative frequencies of ciphertext letters are different from that of plaintext letters. So we sort the frequencies of ciphertext letters in descending order stored in array D. And then we compute Cost1 using following formula:

$$\text{Cost1} = \sum_{i=0}^{25} KS^u(i) - DS^u(i) \qquad \ldots (2)$$

Where $KS^u(i)$ contains frequencies of normal English text letters in descending order and $DS^u(i)$ contains frequencies of ciphertext letters in descending order. If this Cost1 is less than or equal to some tolerance value then only it can be either substitution or combination of permutation and substitution cipher. In combination of permutation and substitution cipher permutation cipher is also applied on ciphertext letters but as permutation has no effect on unigram frequencies so it does not change cost1 as computed by equation (2).

Now to distinguish between substitution and combination of permutation and substitution we compute bigram cost. Bigram cost is difference between relative bigram frequencies of normal English text letters and that of ciphertext letters in descending order. In case of only substitution cipher bigram cost will come less than some tolerance value. But in combination of permutation and substitution ciphers along with substitution, permutation is also applied on plaintext letters so this disturbs normal bigram frequency distribution. Hence in case of combination of permutation and substitution cipher bigram cost is more than tolerance value. Cost2 (bigram cost) is computed using following formula:

$$\text{Cost2} = \sum_{i=1}^{BLN} | F^b(i) - BR^b(i) | \quad \ldots (3)$$

Where F stores relative frequencies of normal English text letters in descending order and BR stores relative frequencies of ciphertext letters in descending order. And ‘b’ is used to indicate bigram frequencies. BLN is number of bigrams to be considered as all bigrams need not be considered.

So unigram cost is computed and if it comes less than or equal to TOLR value than bigram cost is computed. If bigram cost is also less than or equal to TOLR value then input ciphertext is encrypted using Substitution Cipher. But if bigram cost is more than TOLR value then input ciphertext is encrypted using both permutation and substitution ciphers. And if unigram cost is more than TOLR value then given ciphertext is neither encrypted by substitution cipher nor by combination of permutation and substitution cipher.

By running the program on number of ciphertext encrypted by substitution, combination of substitution and permutation ciphers and non-classical ciphers, an experimentally good value of TOLR was found to be 21.0.

If Cost1 is greater than tolerance value then it checks for the possibility of Vigenere Cipher. For detecting vigenere cipher one must know keyword length (or block size). So we start by guessing block size. We choose a variable 'bs' and vary it from 2 to BSIZE, where BSIZE is maximum possible block size to be considered. For every iteration we divide ciphertext into blocks of size 'bs' each. In every block letters at same position are encrypted by same key letter. So we apply frequency analysis on 1^{st} letter to bs^{th} letter on every block and compute cost (1), …, cost (bs) as follows:

$$\text{Cost (i)} = \sum_{j=0}^{25} | KS^u(j) - DS^u(i, j) | \qquad \ldots (4)$$

Where KS array contains unigram frequencies of normal English text in sorted order, and DS(i,j) contains relative frequency of j^{th} letter of ciphertext at i^{th} position. If all these cost values, i.e. cost (1), …, cost(bs) are less than tolerance value then given ciphertext is encrypted by Vigenere Cipher. If atleast one of the cost values is more than tolerance value then program breaks current iteration and continues next iteration. If till last iteration vigenere cipher is not detected then program reports that given ciphertext is encrypted by some non-classical cipher.

2.3 RESULTS:

If input file size is very small then program may not give correct results. If by simply copying the same data input file size is increased then, as it does not change relative frequency of letters, program may still not give correct results. Some experiments are performed on 200 files and results are recorded. Out of these 200 files, first 100 files are large sized files. Roughly speaking in these 100 files 80 files are in the range of 2KB to 8 KB, and remaining 20 files are around 20 KB to 30 KB. Last 100 files are small sized files; all are in the range of 300 to 800 bytes.

1) When these files were encrypted by permutation cipher then the program gives 100% correct results for both large sized as well as small sized files.
2) When these files were encrypted by substitution cipher then the program gives 100% correct results for large sized files and 97% correct results for small sized files.
3) When these files were encrypted by combination of permutation and substitution cipher then the program gives 100% correct results for large sized files and 70% correct results for small sized files.

4) When these files were encrypted by vigenere cipher then the program gives correct results for all large sized files. Out of these 100 files, for first 50 files (whose sizes are varying from 2KB to 6KB) program gives correct results when keyword length is at most 15. For the next 30 files (whose sizes are varying from 7KB to 12 KB) program gives correct results when keyword length is at most 35. For remaining 20 files (whose sizes are varying from 15KB to 35KB) it gives correct results for keyword length of at most 100.

   Program when tested with small sized 100 files encrypted by vigenere cipher gives correct results for 85 files for keyword length of = 5. It gives incorrect results for 15 files.

In case of combination of permutation and substitution cipher bigram frequency analysis is involved so to get correct results input file size should be considerably high. If input file size is small then the number bigrams considered over total number of bigrams will not be much so program may not give correct results. In case of vigenere ciphers if key length increases then size of ciphertext has to be increased in order to detect the cipher correctly.

CHAPTER 3

Classification of DES and IDEA

3.1 Brief Description of DES and IDEA: [6]

Both DES and IDEA are block ciphers.

3.1.1 DES (Data Encryption Standard):

DES encrypts data in 64-bit block. The algorithm takes 64-bit block of plaintext as input and outputs a 64-bit block of ciphertext. DES is a symmetric algorithm, i.e., the same algorithm and key are used for both encryption and decryption except for key schedule. The key length used in DES is 56-bits.

DES algorithm is a combination of two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on key. This is known as a round. DES has 16 rounds; it applies the same combination of techniques on plaintext block 16 times (as shown in fig 3.1.)

After an initial permutation, the block is broken into a right half and a left half, each 32-bit long. Then there are 16 rounds of identical operations, called function f, in which the data are combined with the key. After the sixteenth round, the right and left halves are joined, and final permutation (the inverse of initial permutation) finishes off the algorithm.

In each round (as shown in fig3.2), the key bits are shifted, and then 48 bits are selected from 56 bits of the key. The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and a permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again. These four operations make up function f. The output of function f is then combined with the left half via another XOR. The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times, making 16 rounds of DES.

If B_i is the result of i^{th} iteration, L_i and R_i are left and right halves of B_i, K_i is 48-bit key for round i, and f is the function that does all the substituting and permuting and XORing with the key, then a round looks like:

$$L_i = R_{i-1}$$

& $$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

The left and right halves are not exchanged after the last round of DES; instead the concatenated block $R_{16}L_{16}$ is used as the input to the final permutation.

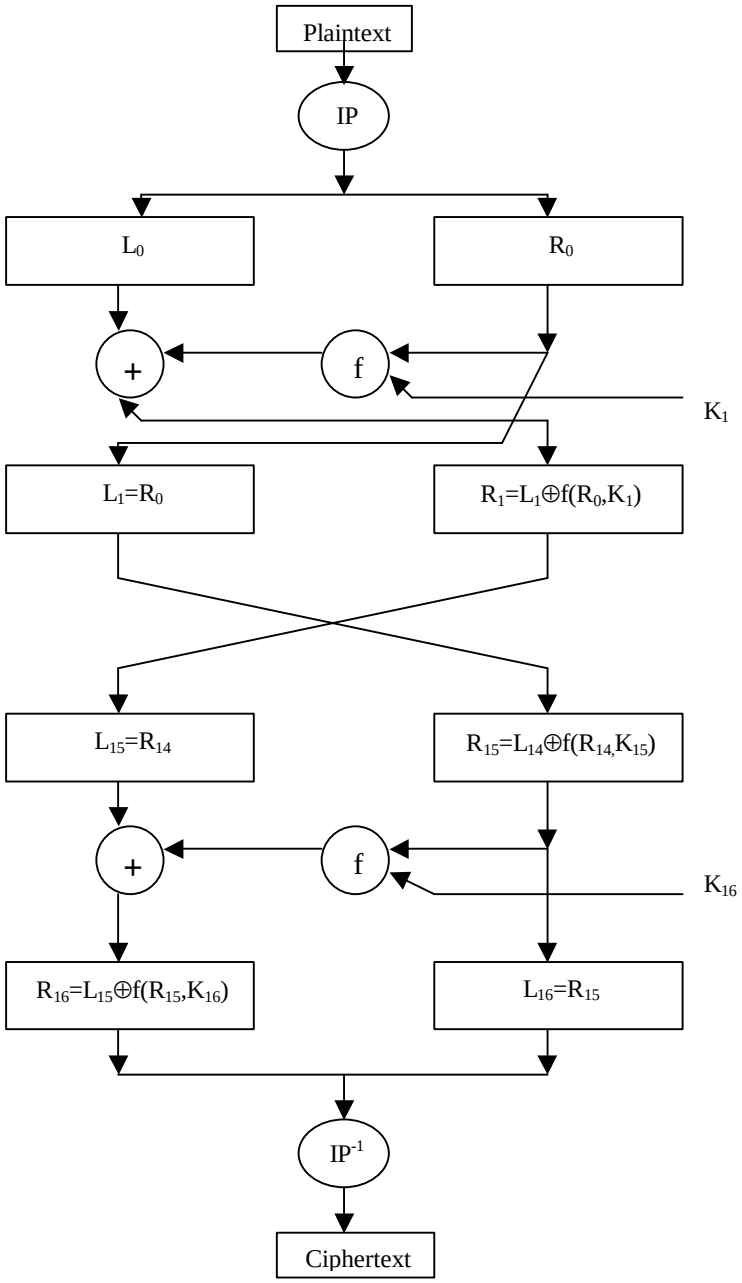


Fig. 3.1: DES

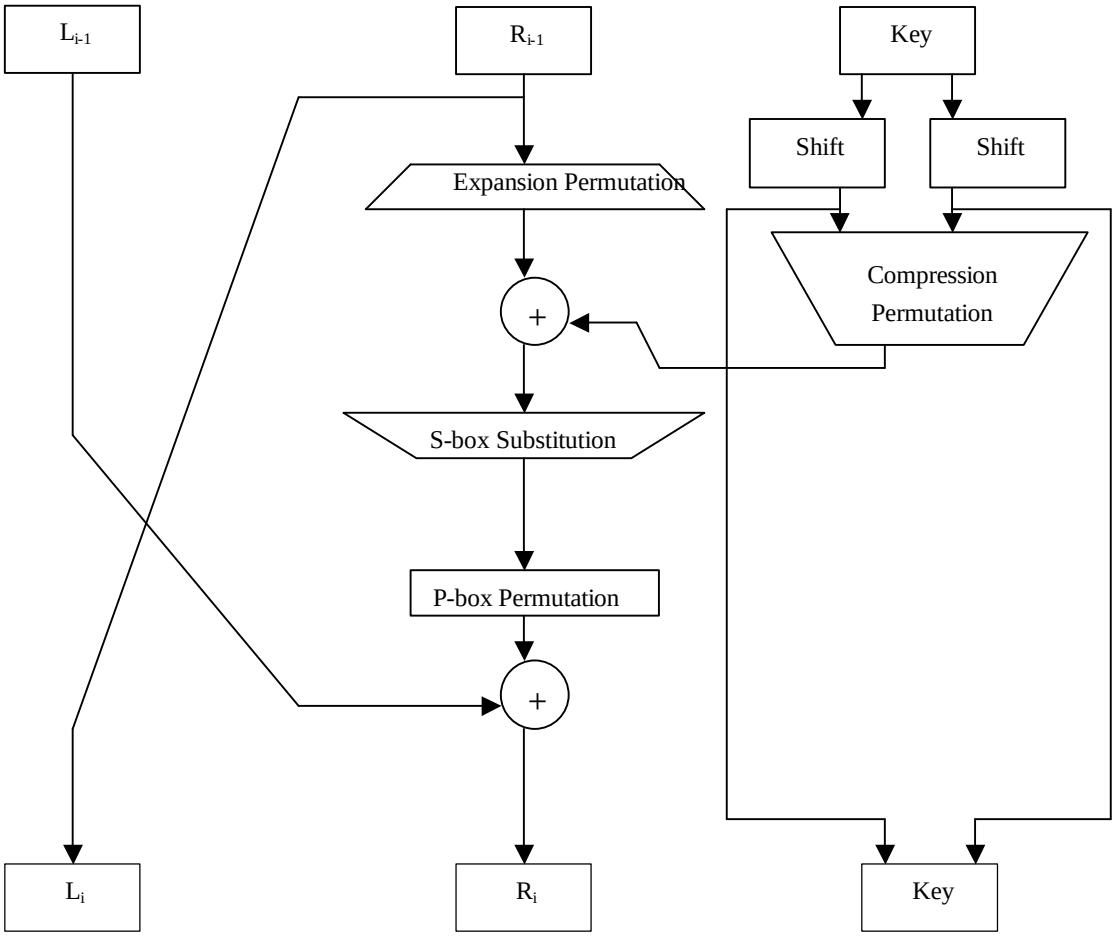


Fig 3.2: One round of DES

3.1.2 IDEA (International Data Encryption Algorithm):

IDEA also operates on 64-bit plaintext blocks. The key used in IDEA is 128-bits long. In case of IDEA also the same algorithm is used for both encryption and decryption. The algorithm is based on "mixing operations from different algebraic groups".

Fig. 3.3 shows an overview of IDEA. The 64-bit plaintext block is divided into four 16-bit sub-blocks: X_1, X_2, X_3, and X_4. These four sub-blocks become the input to the first round of the algorithm. There are eight rounds total.

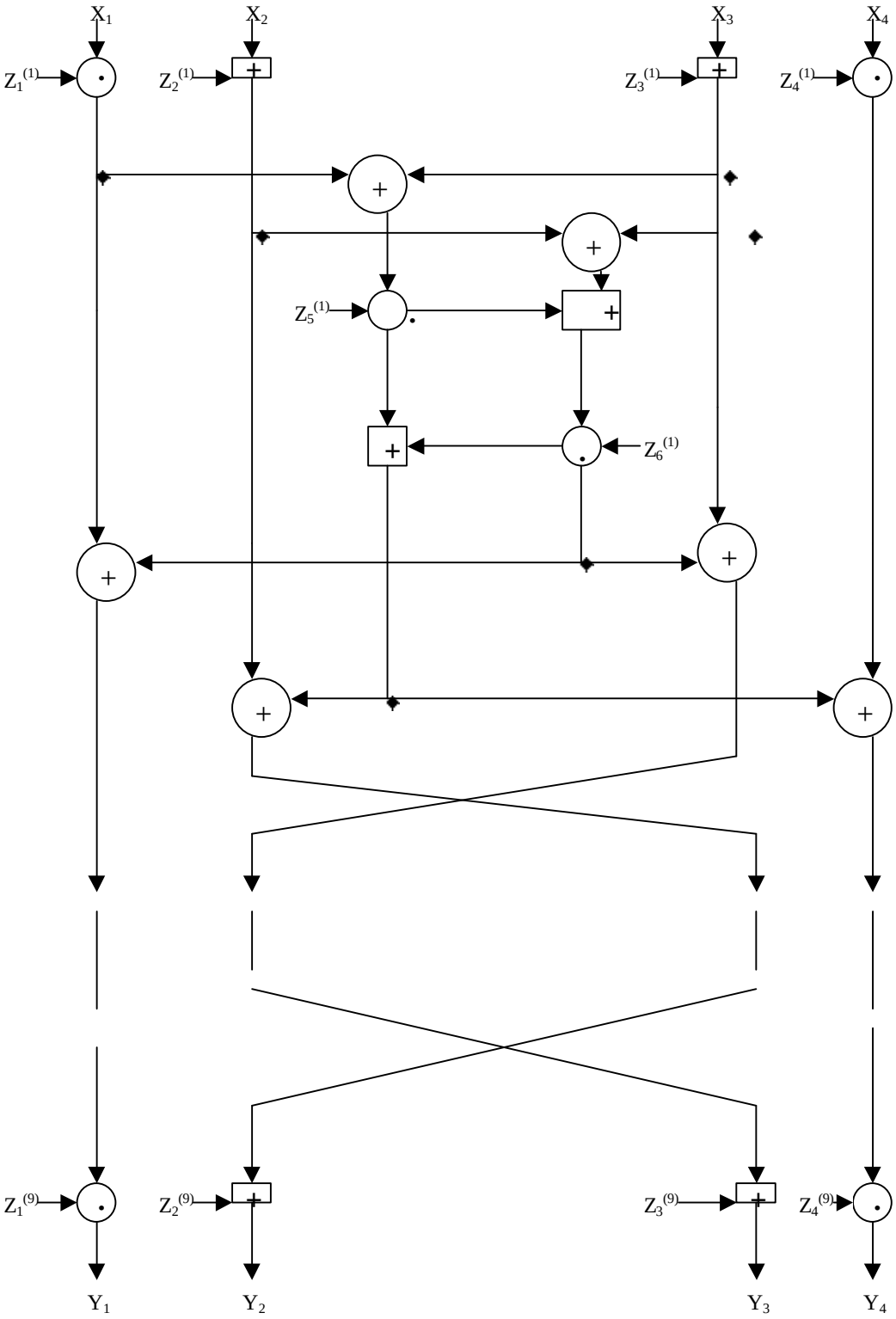
In each round, the sequence of operations is as follows:

1) Multiply X_1 and the first subkey.
2) Add X_2 and the second subkey.
3) Add X_3 and the third subkey.
4) Multiply X_4 and the fourth subkey.
5) XOR the result of steps (1) and (3).
6) XOR the results of steps (2) and (4).
7) Multiply the result of step (5) with the fifth subkey.
8) Add the results of steps (6) and (7).
9) Multiply the result of step (8) with the sixth subkey.
10) Add the result of steps (7) and (9).
11) XOR the results of steps (1) and (9).
12) XOR the results of steps (3) and (9).
13) XOR the results of steps (2) and (10).
14) XOR the results of steps (4) and (10).

The output of the round is the four sub-blocks that are the results of steps (11), (12), (13), and (14). Swap the two inner blocks (except for the last round) and that's the input to the next round. After the eight round, there is a final output transformation:

1) Multiply X_1 and the first subkey.
2) Add X_2 and the second subkey.
3) Add X_3 and the third subkey.
4) Multiply X_4 and fourth subkey.

Finally, the four sub-blocks are reattached to produce the ciphertext.



X_i:16-bit plaintext sub-block, Y_i: 16-bit ciphertext sub-block, $Z_{(i)}^r$: 16-bit key sub-block.
$\oplus$: bit-by-bit XOR of 16-bit integers. $\boxplus$: Addition modulo 2^{16} of 16-bit integers.
$\odot$: Multiplication modulo $2^{16} + 1$ of 16-bit integers with the zero sub-block corresponding to 2^{16}.

Fig 3.3: IDEA

3.2 Techniques Attempted for Classification of DES and IDEA:

We have tried several techniques to classify DES and IDEA. First we tried some randomness tests, then we used XOR operation along with randomness test, and then we tried a combination of threshold functions to classify DES and IDEA.

3.2.1 Randomness Tests: [7]

Our first approach for classifying DES and IDEA was to run several randomness tests on large number of files encrypted by DES and IDEA. Namely frequency test, run test, collision test, gap test, serial test, poker test, and permutation test were tried. Chi-square test is a basic method for studying random data in connection with many of these tests.

Chi – Square Test (χ^2 Test):

The chi-square test compares observed and expected frequencies (counts). The chi-square test statistic is basically the sum of the squares of the differences between the observed and expected frequencies, with each squared difference divided by the corresponding expected frequency. Suppose n observations are taken in an experiment and every observation can fall into one of the k categories. Let p_s be the probability that each observation falls into category s, and let y_s be the number of observations that actually do fall into category s. Then variance V is computed as:

$$V=\sum_{s=1}^{k}\frac{(y_s-np_s)^2}{np_s}$$

Frequency Test:

Frequency test looks for uniform distribution of numbers between 0 and (d-1), where d is some integer. For each integer r, $0 = r < d$, the number of times $Y_j = r$ for $0 = j < n$ is counted, and chi-square test is applied. Chi-square values for almost all the DES and IDEA files are out of range.

Run Test:

In this lengths of increasing or decreasing segments is examined by “run-up” and “run-down” tests, and then chi-square test is applied. For both “run-up” and “run-down” tests chi-square values fall out of range for almost all the DES and IDEA files.

Collision Test:

In collision test number of collisions are counted. Ratio of actual number of collisions to expected number of collisions for DES and IDEA files fall in the same range.

Gap Test:

In this length of "gaps" between occurrences of Y_j in a certain range is examined. For this also chi-square test is applied. For almost all the DES and IDEA files observations fall out of range.

Serial Test:

It looks for uniform distribution of pairs of successive numbers. The number of times that the pair $(Y_{2j}, Y_{2j+1}) = (q, r)$ occurs is counted, for $0 = j < n$; and these counts are made for each pair of integers (q, r) with 0 =q, r<d, and then chi-square test is applied. For almost all the DES and IDEA files observations fall out of range.

Poker Test:

Poker test considers five successive integers, $(Y_{5j}, Y_{5j+1}, \ldots, Y_{5j+4})$ for $0 = j < n$, and counts the number of distinct values in set of five. In general n groups of k successive numbers are considered, and number of k-tuples with r different values are counted, and then chi-square test is applied. For almost all the DES and IDEA files observations fall within the same range.

Permutation Test:

In this test input is divided into n groups containing t elements each. Each group can have t! orderings; the number of times each ordering appears is counted, and then chi-square test is applied. For almost all DES and IDEA files observations fall out of range.

3.2.2 Use of XOR operation:

We tried to use XOR function along with the randomness tests in the following way:

1) XORing the given ciphertext with the first block of ciphertext so that resulted file has one less block than the original one. Then we applied all those randomness tests (discussed in previous section) on the output file of XOR operation. But still randomness tests on DES and IDEA files give same results so this technique also fails to classify DES and IDEA.
2) In the similar way, we've XORed DES and IDEA files with random 64-bit string and then applied all the randomness tests. But even this technique fails to classify DES and IDEA.

3) We also tried XORing every two blocks of ciphertext so that resulted file has almost half the number of blocks than original one. We applied all the randomness tests on the resulted files of this XOR operation. But even this technique fails to classify DES and IDEA.

3.2.3 Use of Threshold Functions:

We tried to construct a function F in the following way:

$$F(b_0, b_1, \ldots, b_{319}) = \{ \ 1 \quad \text{if DES output,}$$
$$0 \text{ if IDEA output } \}$$

where $b_0, b_1, \ldots, b_{319}$ are first 320 bits of DES or IDEA output.

We guessed the following simple form for F :

$$F = \sum_{i=0}^{319} c_i b_i \geq T \qquad \text{for DES}$$

and,
$$F = \sum_{i=0}^{319} c_i b_i < T \qquad \text{for IDEA}$$

where c_i's and T are real numbers.

We used the following objective function:

$$Z = c_0 + c_1 + \ldots + c_{319} - T$$

to be maximized.

So the above problem can be restated as follows:

maximize

$$Z = c_0 + c_1 + \ldots + c_{319} - T$$

subject to

(DES constraints)

$$-c_0b_0 - c_1b_1 - \ldots - c_{319}b_{319} + T = 0$$

(IDEA constraints)

$$c_0b_0 + c_1b_1 + \ldots + c_{319}b_{319} - T = 0$$

Several files were encrypted by DES and IDEA separately with different keywords and first five blocks were taken from each of these files as test cases. By using different combination of constraints (like 100 DES files and 100 IDEA files, 150 DES files and 100 IDEA files, 80 DES files and 80 IDEA files, 90 DES files and 110 IDEA files, 100 DES files and 40 IDEA files, 200 DES and 200 IDEA files etc) values of c_i's and T were found. For doing this (i.e., for solving linear programming) we used matlab's optimization toolbox.

For every combination of constraints, values of c_i's are tested on first segment (first five blocks) of DES files as well as first segment of IDEA files. But no satisfactory results were obtained. Then we took several large files and encrypted them with DES and IDEA. Then for all the combinations of c_i's and T we tested all the files. We counted the number of segments (a segment is equal to 320-bits) with greater than or equal to T value. That is, for every segment

$$s_j = \sum_{i=0}^{319} c_i b_i \qquad 1 = j < n$$

where n is maximum number of segments in given file, it is checked that whether s_j is greater than threshold value or not. Ratio of number of segments with greater than threshold value is taken over total number of segments. This ratio falls within the same range for both DES and IDEA files.

We then extended our threshold gate model to two levels as explained in fig. 4.

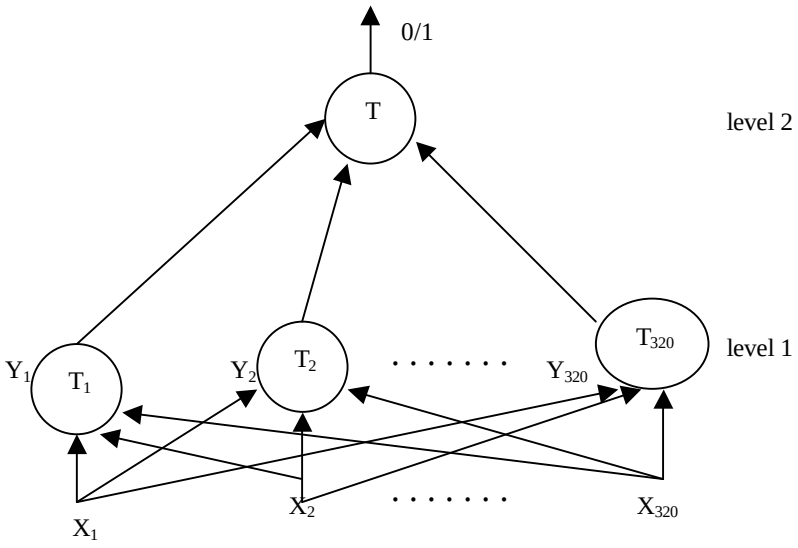


Fig. 3.4: 2 – Level Threshold Gate Model

Every threshold gate outputs 1 bit.

If $\sum_{i=0}^{319} c_i x_i > T$ then it outputs 1

otherwise it outputs 0.

We took 200 segments from each of the DES and IDEA files. Using these 400 equations and 320 different objective functions we found 320 different sets of c_i values. The objective functions used are

$$\min x_i \qquad i = 0 \text{ to } 319$$

for 320 different solutions.

In this 2 level model, 320 threshold gates were used at level1. So level1 takes 320 bits input and outputs another 320 bits. A transformation program converts given ciphertext file segment by segment to level1 file by using cval1 to cval320 files (containing 320 different solutions) and tval file (containing threshold values for each solution). At level2, one threshold gate is used. Coefficient values (c_i values) at this level were found by using level1 files.

In this technique, ciphertext file is first transformed to level1 file and then level1 file is tested with level2 solution. Ratio of segments above threshold value to total number of segments is computed at level2.

This ratio falls almost in the same range for DES and IDEA files. Though the overall range for this ratio was same but scattering of data were found to be slightly different.

We then extended our 2 level threshold gate model to 3 level threshold gate model, as explained in fig. 5.

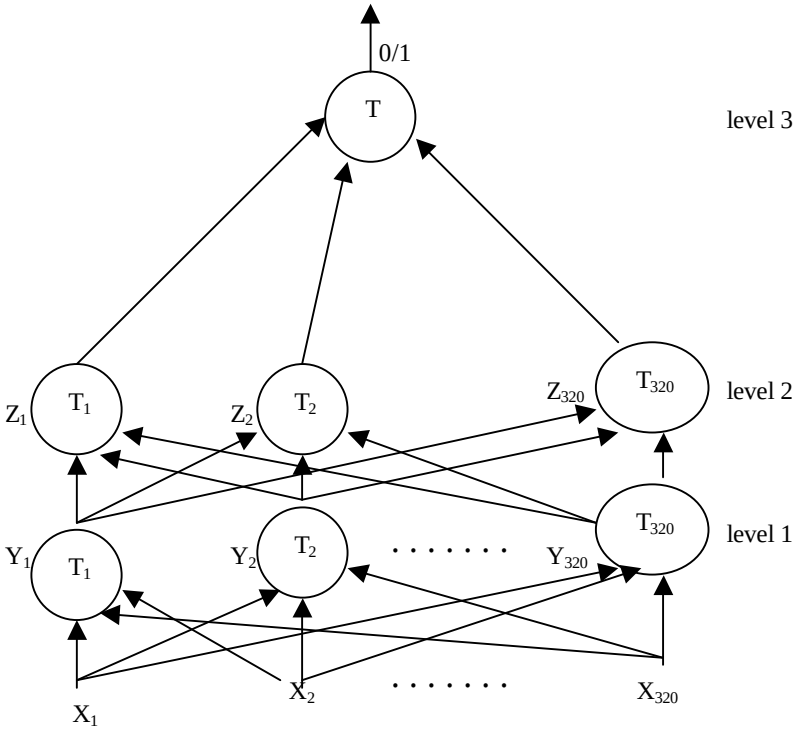


Fig. 3.5: 3 – Level Threshold Gate Model

In this, a given ciphertext file is transformed to level1 file, and a level1 file is transformed to level2 file. Then level2 file is checked with the solution found at level3. Again at level3, ratio of segments with greater than threshold value to total number of segment is computed. Here also, this ratio found to be same for DES and IDEA files.

After this, we tried several simple tricks such as complementation, combination of randomness tests with threshold gate model, ratio of ones or zeros to total number of bits, etc. We couldn't get satisfactory results by any of these techniques.

3.3 Experimental Results:

3.3.1 Randomness Tests:

All the tests were applied on 80 files, 40 DES files and 40 IDEA files. In frequency test, run test, gap test, and serial test chi-square values are in the same range. For both DES and IDEA files results fall either above range or below range of expected chi-square value. In poker test and permutation also, chi-square values fall in the same range. In permutation test values fall above expected range for both DES and IDEA, while for poker test values fall either within the expected range or above expected range. In collision test, ratio of actual number of collisions to expected number of collisions is taken, and was found to be in the range of 1.02 to 1.50 for both DES and IDEA.

Even by using XOR operation along with randomness tests in different ways as described in previous section, we get same results for both DES and IDEA files. The same range of chi-square values were obtained for both DES and IDEA files in case of combination of XOR operation with frequency test, run test, gap test, serial test, poker test, and permutation test. In XORing along with collision test also, ratio of actual number of collisions to expected number of collisions falls within the same range for both DES and IDEA files.

3.3.2 Threshold Gate Model:

For most of the cases we got negative results, i.e., almost same values for DES and IDEA files. For some of the cases results obtained loosely classify DES and IDEA files. Following are the results of those cases where we get no classification of DES and IDEA.

LEVEL1:

Tests are applied on 80 files, 40 DES and 40 IDEA level1 files. There can be two cases, either DES constraints are set to give less than threshold value & IDEA constraints are set to give greater than threshold value, or DES constraints are set to give greater than threshold value and IDEA constraints are set to give less than threshold value. These two cases are considered separately and their results are tabulated in the following tables (Table1 and Table2).

Table 3.1: DES < Threshold Value and IDEA > Threshold Value

No. of DES Constraints	No. of IDEA Constraints	Threshold Value	DES (Segments found above Threshold)	IDEA (Segments found above Threshold)
40	40	5.1111	98% to 100%	98% to 100%
50	40	5.1883	98% to 100%	98% to 100%
80	40	5.3441	92% to 95%	92% to 95%
100	40	5.3848	89% to 92%	89% to 92%
120	40	5.4244	87% to 90%	85% to 91%
100	100	5.4037	89% to 95%	91% to 95%
200	200	1.0095	51% to 54%	46% to 53%
300	100	1.5574	41% to 47%	42% to 49%

Table 3.2: DES > Threshold Value and IDEA < Threshold Value

No. of DES Constraints	No. of IDEA Constraints	Threshold Value	DES (Segments found above Threshold)	IDEA (Segments found above Threshold)
100	100	5.1732	91% to 95%	88% to 96%
50	150	5.400	85% to 91%	85% to 91%
50	200	5.2955	79% to 84%	79% to 84%
150	50	5.300	98% to 100%	98% to 100%
200	50	5.200	98% to 100%	98% to 100%
200	200	5.300	72% to 78%	72% to 79%
210	210	5.400	70% to 75%	70% to 75%
300	100	5.4037	89% to 95%	91% to 95%

The 4^{th} and 5^{th} columns of above tables show percentage of segments with greater than threshold value. For most of the files ratio falls in overlapping range.

In all these combinations objective function used was

$$\text{To maximize } \sum_{i=0}^{319} c_i - T$$

Scattering of ratio is found to be uniform for the above cases.

LEVEL2:

Tests are applied on 80 files, 40 DES and 40 IDEA level2 files. Results are as follows:

1. Ratio of absolute difference of 1s and 0s to total number of bits is taken. For DES files this ratio lies in the range of 0.70 to 5.20 and for IDEA files it is from 0.20 to 4.80.
2. Ratio of 1s to total number of bits is taken. For both DES and IDEA files this ratio lies in the range of 23.50 to 25.50

For both the cases almost all values fall in overlapping range.

LEVEL3:

Tests are applied on 80 files, 40 DES and 40 IDEA level3 files. Results are as follows:

1. 200 DES constraints and 200 IDEA constraints are taken and c_i and T values are found by solving Linear Programming problem. Threshold value was found to be 10.1. Ratio of number of segments with greater than threshold value to total number of segments is taken. For DES files this ratio falls in the range of 53% to 61%. For IDEA files this ratio falls in the range of 51% to 63%.
2. Different set of 200 DES constraints and 200 IDEA constraints are taken and same test as above was applied. Threshold value was found to be 1.00. Again ratio of number of segments with greater than threshold value to total number of segments is taken. For DES files this ratio falls in the range of 64% to 71%. For IDEA files this ratio falls in the range of 62% to 70%.
3. Sign of some of the randomly selected c_i values were changed from positive to negative or negative to positive in the solution found in point number 1. Then ratio of number of segments with greater than threshold value to total number of segments is taken. For DES files this ratio falls in the range of 52% to 58%. For IDEA files this ratio falls in the range of 51% to 62%.

4. Ratio of 1s to total number of bits is taken. For both DES and IDEA files this ratio falls in the range of 24 to 27.
5. Compliment all the test files. Compute the ratio of number of segments with greater than threshold value to total number of segments with solution found in point number 1. For DES files this ratio falls in the range of 55% to 63%. For IDEA files this ratio falls in the range of 53% to 62%.
6. Compliment all the test files. Compute the ratio of number of segments with greater than threshold value to total number of segments with solution found in point number 2. For DES files this ratio falls in the range of 75% to 81%. For IDEA files this ratio falls in the range of 74% to 82%.

For all the above cases almost values falls in overlapping range.

Following tricks have also been tried but give negative results for test files at all the three levels and also for complement of files at all these levels:

1. XORing successive bits, so that total number of bits is reduced by one and then taking ratio of 1s to total number of bits.
2. XORing each two successive bits, so that total number of bits is reduced by one half, and then taking ratio of 1s to total number of bits.

Following are the results of those cases where we are getting loose classification among DES and IDEA.

LEVEL1:

For level2, we've computed 320 different sets of c_i values using different objective functions. Using each of these values individually we've tested level1 files. For some of the cases, we found scattering of ratio of number of segments with greater than threshold value to total number of segments slightly different for DES and IDEA files. These results are as follows:

1. When objective function used is min c_{94} and T=0, then in 62.5% of DES files ratio lies above 43.0% and for 62.5% of IDEA files ratio lies below 43.0%.
2. When objective function used is min c_{207} and T=0, then in 52.5% of DES files ratio lies above 47.0% and for 80.0% of IDEA files ratio lies below 47.0%.
3. When objective function used is min c_{240} and T=0, then in 70.0% of DES files ratio lies above 46.0% and for 57.5% of IDEA files ratio lies below 46.0%.

4. When objective function used is min c_{257} and T=0, then in 55.0% of DES files ratio lies above 46.0% and for 55.0% of IDEA files ratio lies below 46.0%.
5. When objective function used is min c_{281} and T=0, then in 52.5% of DES files ratio lies above 43.0% and for 62.5% of IDEA files ratio lies below 43.0%.

Now using each of those 320 different set of values individually we've tested compliment of level1 files. For some of the cases, we found scattering of ratio of number of segments with greater than threshold value to total number of segments slightly different for DES and IDEA files. These results are as follows:

1. When objective function used is min c_{92} and T=0, then in 75.0% of DES files ratio lies below 46.5% and for 57.5% of IDEA files ratio lies above 46.5%.
2. When objective function used is min c_{181} and T=0, then in 72.5% of DES files ratio lies below 52.0% and for 60.0% of IDEA files ratio lies above 52.0%.
3. When objective function used is min c_{223} and T=0, then in 75.0% of DES files ratio lies below 44.5% and for 60.0% of IDEA files ratio lies above 44.5%.

LEVEL2:

Ratio of number of segments to total number of segments was found to be slightly different for level2 files.

1. 200 DES constraints and 200 IDEA constraints are taken and c_i and T values are found by solving Linear Programming problem. Threshold value was found to be 13.9925. Ratio of number of segments with greater than threshold value to total number of segments is taken. For 67.5% of the DES files this ratio lies below 55.5%. For 60.0% of the IDEA files this ratio lies above 55.5%.
2. Compliment all the test files. Compute the ratio of number of segments with greater than threshold value to total number of segments with solution found in point number 1. For 60.0% of the DES files this ratio lies above 65.0. For 72.5% of the IDEA files this ratio lies below 65%.

For level3, we've computed 320 different sets of c_i values using different objective functions. Using each of these values individually we've tested level2 files. For the following cases, we found scattering of ratio of number of segments with greater than threshold value to total number of segments slightly different for DES and IDEA files.

1. When objective function used is min c_{135} and T=0, then in 67.5% of DES files ratio lies above 62.0% and for 72.5% of IDEA files ratio lies below 62.0%.

2. When objective function used is min c_{275} and T=0, then in 65.0% of DES files ratio lies above 59.0% and for 55.0% of IDEA files ratio lies below 59.0%.

By increasing the number of levels there is not much improvement in the result as was expected.

CHAPTER 4

Conclusion and Further Work

Classical ciphers can be easily classified by frequency distribution technique. But this technique cannot be extended for classification of modern ciphers like DES and IDEA. The differentiation of DES from IDEA is very difficult, as these ciphers do not posses non-randomness properties. DES cannot be classified from IDEA by applying randomness tests, or XOR-operation along with randomness tests. By using threshold functions, we get slightly positive results.

In our threshold functions only linear properties are used. One may get better results by introducing non-linearity in computing these threshold functions. Secondly, one can extend this work to include other ciphers like AES(Advanced Encryption Standard), Blowfish, CAST, FEAL, SQUARE, etc.

References

[1] http://www.dgsciences.com/codeclas.htm

[2] Douglas R. Stinson, *Cryptography Theory and Practice*, CRC Press.

[3] Dorothy Elizabeth, Robling Denning, *Cryptography and Data Security*, Addison Wesley Publication.

[4] http://www.trincoll.edu/depts/cpsc/cryptography/vigenere.html

[5] http://www.math.nmsu.edu/~crypto/Frequency.html

[6] Bruce Schneier, *Applied Cryptography*, John Wiley And Sons, Inc.

[7] Pratima Gupta, *Comparison of DES and A New Cryptosystem*, M.Tech. Thesis, Department of Computer Science & Engineering, May'1998.