

M.Tech. (C.S.) Dissertation Series

**Safe Encoding: Making the encoded text of an  
instantaneous code as random as possible**

A dissertation submitted towards partial fulfilment  
of the requirements for the M.Tech (Computer Science) degree of

**Indian Statistical Institute**

By

Nagendar Gouru

Under the supervision of

**Prof. Bimal Kumar Roy**


**Indian Statistical Institute**

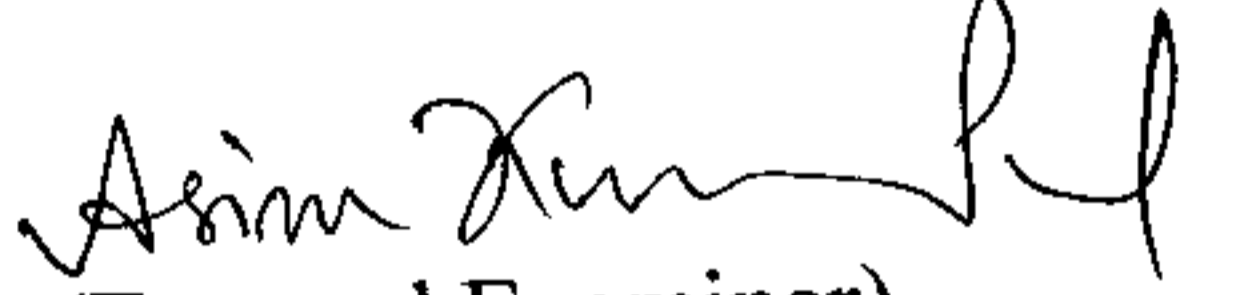
203, Barrakpore Trunk Road  
Calcutta - 700 035.

## Certificate of Approval

This is to certify that the thesis entitled *Safe Encoding: Making the encode text of an instantaneous code as random as possible* submitted by Nagendar Goulu, towards partial fulfilment of the requirement for M.Tech. in Computer Science degree of the *Indian Statistical Institute, Calcutta*, is an acceptable work for the award of the degree.

Date: July 23, 2001

  
(Supervisor) 23.7.01

  
(External Examiner)  
AIC PAL 23.7.2001  
IIMC

## Acknowledgement

My sincere gratitude goes to Prof. Bimal Kumar Roy for his guidance, advice, enthusiasm and criticisms throughout the course of this dissertation.

I am indebted to all my teachers.

Finally I would like to thank all my classmates , juniors and Mr.Sreenivasa Rao without whose co-operation and support this work would not have been a success.

Nagendar Gouru.

# Contents

## 0 Abstract

## 1 Introduction

- 1.1 Path length and Huffman tree
- 1.2 Information Theory: Channel and mutual information
- 1.3 Cipher text only attack

## 2 Padding algorithm

## 3 Case Study

## 4 Conclusion

## Bibliography

## Abstract

We know from information theory that the channel capacity is achieved when the input events are statistically independent and have the same probability distribution and the channel capacity is the maximum amount of information we can send through the channel. Here a padding algorithm is designed which takes any plain text (i.e. message) as input and outputs the corresponding encoding scheme whose encoded text will have the frequencies of strings "00", "01", "10", "11" close together. This makes the encoded text close to random, since English language can be considered to be a Markov Process of order 1. This also protects against cipher text only attack by transmitting symbols randomly giving almost zero information to the eavesdropper. Without loss of generality, Huffman code is taken as the coding scheme in initial steps to the algorithm. Since the minimum variance code ensures less degradation of the coding scheme in presence of noise, the least skewed Huffman tree is considered. About the closeness bound of the frequencies of strings "00", "01", "10", "11" is discussed .

**Safe Encoding: Making the encoded text of an  
instantaneous code as random as possible**

Nagendar Goudu  
Indian Statistical Institute  
Calcutta

July 23, 2000

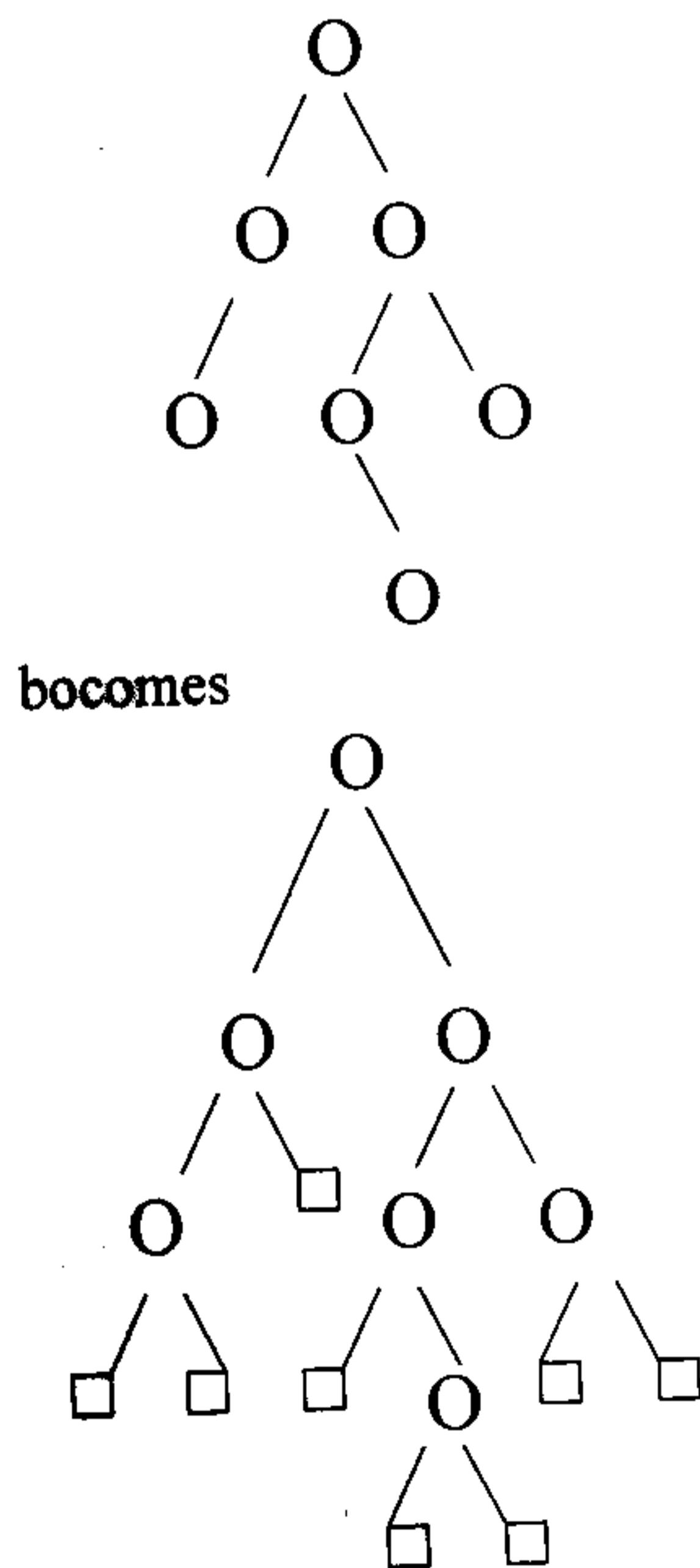
# Chapter 1

## Introduction

First we will try to get the least skewed Huffman tree. Then we will discuss the necessity to design a padding algorithm in detail. The results that are needed in our work are also provided.

### 1.1 Path length and Huffman tree

Let us extend each binary tree diagram by adding special nodes wherever a null sub tree was present in the original tree, so that



.....(fig 1.1)

The latter is called an *extended binary tree*. After the square shaped nodes have been added in this way, the structure is sometimes more convenient to deal with. It's clear that every circular node has two sons and every square node has none. If there are  $n$  circular nodes and  $s$  ~~circular~~ <sup>square shaped</sup> nodes, we have  $n+s-1$  edges (since the diagram is a connected acyclic graph), and, counting another way, by the number of sons, we see there are  $2n$  edges. Hence it is clear that

$$S = n+1; \dots\dots\dots(1.2)$$

i.e., the number of "external" nodes just added is one more than the number of "internal" nodes we have originally.

Assume that a binary tree has been extended in this way. The *external path length* of the tree, denoted by  $E$ , is defined to be the sum – taken over all external (square) nodes – of the lengths of the paths from the root to each node. The *internal path length*, denoted by  $I$ , is the same quantity summed over the internal (circular) nodes.

In fig(1.1) the external path length is  $E = 3+3+2+3+4+4+3+3 = 25$ , and the internal path length is  $I = 2+1+0+2+3+1+2 = 11$ . These two quantities are always related by the formula

$$E = I+2n, \dots\dots\dots(1.3)$$

Where  $n$  is the number of internal nodes.

To prove formula (1.3), consider deleting an internal node  $V$  at a distance  $k$  from the root, where both sons of  $V$  are external. The quantity  $E$  goes down  $2(k+1)$ , since the sons of  $V$  are removed, then it goes up  $k$ , since  $V$  becomes external, so the net change in  $E$  is  $k-2$ . The net change in  $I$  is  $-k$ , so (1.3) may be proved by induction.

It is not hard to see that the internal path length (and hence the external path length also) is highest when we have a degenerate tree with linear structure;

in that case the internal path length is

$$(n-1) + (n-2) + \dots + 1 + 0 = 0.5(n^2 - n).$$

Consider now the problem of discovering a binary tree with  $n$  nodes having minimum path length; such a tree will be important, since it will minimize the computation time for various algorithms. Clearly, only one node (the



root) can be at zero distance from the root; at most two nodes can be at distance 1 from the root, at most four can be 2 away, etc. So, we see that the internal path length is always at least as big as the sum of the first n terms of the series

$$0, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, \dots$$

This is the sum  $\sum_{1 \leq k \leq n} \lfloor \log_2 k \rfloor$ , which is same as

$$(n + 1)q - 2^{(q+1)} + 2, \text{ where } q = \lfloor \log_2 \frac{(n+1)}{2} \rfloor \dots \dots \dots (1.4)$$

(Proof follows from the following:

We know that 
$$\sum_{1 \leq k \leq n} (a_{k+1} - a_k) b_k = a_n b_n - a_1 b_1 - \sum_{1 \leq k \leq n} a_k (b_{k+1} - b_k)$$

Putting  $b_n = n$ , we get 
$$\sum_{1 \leq k \leq n} a_k = n a_n - \sum_{1 \leq k \leq n} k (a_{k+1} - a_k) \quad (n > 0)$$

hence the result is just by substitution)

The optimum value (1.4) is essentially of the form  $n \log n$ ; this optimum is clearly achieved in the complete tree with n internal nodes where we may number the nodes 1,2,3...n; this numbering has the useful property that the father of node k is node  $\lfloor k/2 \rfloor$ , the sons of node k are nodes 2k and 2k+1. The external nodes are numbered n+1 through 2n+1, inclusive.

It follows that a complete binary tree may be simply represented in sequential memory locations. These results have an important generalization if we shift our point of view slightly. Suppose that we are given m real numbers  $w_1, w_2, w_3 \dots w_m$ , the problem is to find an extended binary tree with m external nodes, and to associate  $w_1, w_2, w_3 \dots w_m$  with these nodes in such a way that the  $\sum w_j l_j$  is minimised, where  $l_j$  is the length of path from the root and the sum is taken over all external nodes. It is not hard to realise that a perfectly balanced tree does not give the minimum weighted path length when the weights are 2, 3,4, and 11. Although we have seen that it does give the minimum in the special case  $w_1 = w_2 = \dots = w_m = 1$ .

An elegant algorithm for finding a tree with minimum weighted path length has been given by D. Huffman: First find the two w's of lowest values, say  $w_1$  and  $w_2$ . Then solve the problem for m-1 weights  $w_1 + w_2 \dots w_m$ , and replace the external node of weight  $w_1 + w_2$  in this solution

by an internal node having two external nodes of weights  $w_1$  and  $w_2$  as its children.

It's not hard to show that this method does in fact minimise the weighted path length by induction on  $m$ . Suppose that  $m \geq 2$  and  $w_1 \leq w_2 \leq w_3 \leq \dots \leq w_m$ . And suppose that we are given a tree which minimises the weighted path length. (Such a tree certainly exists, since only finitely many binary trees with  $m$  terminal nodes are possible) Let  $B$  an internal node of maximum distance from root. If  $w_1$  and  $w_2$  are not the weights already attached to the sons of  $B$ , we can interchange them with the values that are already there, and not increase the weighted path length. Thus there is a tree which minimises the weighted path length and which contains the internal node having two external nodes weights  $w_1$  and  $w_2$  as its children. Now it is to see that the weighted path length of a such a tree is minimised iff the tree with the internal node having two external nodes of weights  $w_1$  and  $w_2$  as minimum path length its children replaced by the external node of weight  $w_1 + w_2$  has minimum path length for weights  $w_1+w_2, w_3 \dots w_m$ , since the numbers which appear in the circular nodes of an extended binary tree are equal to the sums of the weights in the external nodes of the corresponding sub tree and sum of all values in the circular nodes is equal to the weighted path length.

In general there are many trees, which minimise  $\sum w_j l_j$ . If the  $w$ 's are kept in order throughout the construction and if when  $w_1$  and  $w_2$  are removed, the quantity  $w_1+w_2$  is placed higher in the ordering than any of the same value, then the tree constructed by Huffman's method has the smallest value of  $\max l_j$  and of  $\sum l_j$ . Among all trees which minimize  $\sum w_j l_j$ . This can also be accomplished in another way. We need not change the Hoffman algorithm, and suppose that we got a Hoffman tree  $T$ . If this tree is unique up to the lengths and sum of the lengths, there is nothing to do. If not, there must exist three nodes of equal weights, one of which is neutral. Now interchange the position, of sub trees rooted. At the internal node and the external node (in case of ties take one that is closer to the root) of equal weight if it reduces either  $\sum l_j$  or  $\max l_j$ . Do the above step until no such interchange is possible. This will change  $T$  to the least skewed Huffman tree. We can jump into information theory from here after observing the following connective:

**Weighted path length** = length of encoded text or cipher.

## 1.2 Information theory : Channel and mutual information:

The average code length of an encoding scheme is defined to be  $L = \sum p_j l_j$ , where  $l_j$  is the length of the representation of the  $j^{\text{th}}$  symbol  $s_j$ . In the notation of a  $j^{\text{th}}$  order Markov process,  $P(s_i/s_{i1}, s_{i1} \dots s_{ij})$  is the conditional probability of seeing  $s_i$  given that we have just seen all the letters  $s_{i1}, s_{i1} \dots s_{ij}$ , in that order. The entropy of a signalling system  $S$  having symbols  $s_i$  and probabilities  $p_i$  is defined as

$$H_r(S) = \sum_{i=1}^q p_i \log_r(1/p_i)$$

Where  $s_1, s_2, \dots, s_q$  are the source symbols that are to be sent. And the code's alphabet as is symbols ( $r$  for the radix of the system).

We consider the case  $r=2$ .

Of course  $H_r(S) = H_2(S) \log_r(2)$ .

It can be proved from the two facts.

1.  $\log(x) \leq x-1$ , and
2. if  $x_i$  and  $y_i$  are two probability distributions (i.e.  $\sum x_i = \sum y_i = 1$  and each  $x_i$  and  $y_i$  is non-negative) then

$$\sum_{i=1}^v x_i \log_2 \left( \frac{y_i}{x_i} \right) \leq 0$$

That the entropy bounded above by  $\log_r(q)$  i.e.  $H_r(S) \leq \log_r(q)$ .

The relation between the average code length  $L$  and the entropies  $H(S)$  is given by  $H_r(S) \leq L$ .

### **Channel:**

An information channel is statistical model of the medium through which the signal processes. We need to formalise this idea in order to compute how much information goes through in this channel. A channel is described set of confusable probabilities  $P(b_j/a_i)$ , which are the probabilities that an input  $a_i$  from an alphabet of  $q$  letters will appear as  $b_j$  from an alphabet of  $s$

letters. The sizes of  $q$  and  $s$  of the alphabets need not be the same. In this model the channel is completely described by the matrix of conditional probabilities

$$P_{ij} = (P(b_j/a_i)).$$

A row contains all the probabilities that a particular symbol  $a_i$  becomes  $b_j$ .

We can define system entropies as follows.

Entropy of information source  $A$  is

$$H_r(A) = \sum_{i=1}^q p(a_i) \log_r(1/p(a_i))$$

The entropy function has the following properties:

$$(1) H_r(A) \geq 0$$

$$(2) H_r(A) \leq \log_r q$$

$$(3) H_r(A) = \log_r q, \text{ when all the source symbols are equally likely.}$$

Similarly the entropy of the received symbols  $H_r(B)$  can be defined as

$$H_r(B) = \sum_{j=1}^s p(b_j) \log_r(1/p(b_j))$$

This quantity measures the uncertainty of the output symbols and has the same properties as the entropy of the source.

Similar expressions for conditional entropy given a particular  $b_j$ , is given by

$$H_r(B) = \sum_{i=1}^q p(a_i/b_j) \log_r(1/p(a_i/b_j))$$

we can also define  $H_r(A/B)$  and  $H_r(B/A)$  similarly. To measure the uncertainty of the joint even for both source and receiver, we define the joint entropy in a similar way:

$$H_r(A, B) = \sum_{i=1}^q \sum_{j=1}^s p(a_i, b_j) \log_r(1/p(a_i, b_j))$$

It is easy to see that

$$H_r(A, B) = H_r(B) + H_r(A/B).$$

Consider again the afore mentioned transmission system. The input symbols are the  $a_i$  and the output symbols are the  $b_j$ , and the channel is defined by the conditional probabilities  $P_{i,j}$ .

Prior to reception, the probability of the input symbol  $a_i$  was  $p(a_i)$ . This is a priori probability of  $a_i$ . After reception of  $b_j$ , the probability that the input

symbol was  $a_i$  becomes  $p(a_i/b_j)$ , the conditional probability that we sent  $a_i$  given that we received  $b_j$ . This is an a posterior probability of  $a_i$ . The change in probability measures how much the receiver learnt from the reception of  $b_j$ . In the ideal channel with no noise, the a posterior probability is 1, since we are certain from the received  $b_j$  exactly what was sent. In practical systems there are finite non zero probabilities that errors will occur and the receiver can't be absolutely sure that what was sent. The difference between the information uncertainty before and after reception of a  $b_j$  measures gain in information due to the reception of the  $b_j$ . This information is called the mutual information and is naturally defined as

$$I(a_i; b_j) = \log_r(1/p(a_i)) - \log_r(1/p(a_i/b_j)) = \log_r(p(a_i/b_j)/p(a_i))$$

Here also we can define conditional mutual information  $I(A; b_j)$  which is the information gain provided by the reception of  $b_j$ . Finally  $I(A;B)$ , which is symmetric in the two alphabets, can also be defined naturally to be the measure of the information gain of the whole system and doesn't depend on the individual input and output symbols but only their frequencies; it is called *the system mutual information*.

We can see that

$$I(A;B) \geq 0$$

$I(A;B) = 0$  if and only if A and B are independent

$I(A;B) = I(B;A)$  from symmetry.

$$I(A;B) = H(A) + H(B) - H(A,B) \geq 0$$

Hence  $I(A;B) = H(B) - H(B/A) = H(A) - H(A/B)$ .

We define the channel capacity C as the maximum mutual information over all possible assignments of the  $p(a)$ ,

$$C = \max_{P(a)} I(A;B)$$

It can be proved that the binary symmetric channel has the channel capacity C provided that the two input symbols are chosen with equal frequency.

Hence we need to design codes, which produces random strings of code's alphabet.

Another main reason for the desire to get random encoded text is to prevent cipher text only attack.



### 1.3 Cipher text only attack.

We assume that the channel we are using is public in the sense that anybody can have access over it partly or fully. In this light, we classify the attempts of an opponent as follows: cipher text only, known plain text, and chosen cipher text in the increasing order of strength. In the cipher text only attack, the opponent possesses a string of cipher text. Since we are making it random, opponent hardly gets any information from that. The crypto analyst is assumed to have full knowledge of the encoding and decoding functions. In addition, he or she may also have a variety of side information such as language statistics, knowledge of context, etc. in cipher text only attack which is the weakest one, opponent will have some cipher text, and doesn't know the key. Perfect secrecy can be realized in one time pad, where the plain text and the key are both bit strings of a specified length, and the cipher text is constructed by making the bit wise ex-or of the plain text and key. But the key must be randomly generated. We are using this idea here by making the encoded text random. Cipher text only attack fails completely if no opponent can predict with better than average success the next bit to be emitted. Since complete randomness is not practical, we would be satisfied if the above prediction is not possible in reasonable time. Since English language is a Markov process of order 1, the encoded text would be sufficiently random if we achieve equal number of 00,01,10,11.

## 2. Padding algorithm

Input: Plain text (message)

Output: The code of each source symbol which will make the frequencies of the strings "00", "01", "10", "11" close together.

Algorithm:

( Steps 2 to 5 are considered to be Trivial Twisting steps.

Note: In the below steps from step3 to step5 there will be effect on frequencies of strings other than "ab", so in each place where the interchange takes place we have to calculate the frequencies of strings "00", "01", "10", "11".

And also note that in step3, we have to consider the symbols which is coming next to the symbols which are participating in the interchange, in step4, we have to consider the symbols which is coming previous to the symbols which are participating in the interchange, in step5, we have to consider the symbols which are coming next to the symbols and the symbols which are coming previous to the symbols which are participating in the interchange.

(In Steps 6 and 7, which are extension steps, not much calculation is needed)

(In step8 which is also extension step, but here we have to consider the symbols which are coming next to the symbol whose code we are extending and we have to calculate the changes in the frequencies of strings "10"/"11" depends upon the codes of the symbols next to this symbol and codes with first symbol 0/1 and with the help of the conditional probabilities (the number of such pairs).)

In step9 which is also extension step, but here we have to consider the symbols which are coming next to the symbol whose code we are extending and we have to calculate the changes in the frequencies of strings "00"/"01" depends upon the codes of the symbols next to this symbol and codes with first symbol 0/1, and with the help of the conditional probabilities (the number of such pairs).

Step 0: Calculate the frequencies of source alphabet and construct Huffman tree and calculate the frequencies of strings "00", "01", "10", "11" in corresponding cipher text. Repeat steps 1 to 9 until all four strings covered.

Step1: Find the Maximum frequency and minimum frequency among the frequencies of the strings "00", "01", "10", "11". The difference between the Maximum and minimum frequencies is say delta. Say the minimum frequency string is "ab".

Step2: Go for the codes of symbols with same code length and which are having same beginning and ending code symbols and the string "ab" as sub string in one or more places. Interchange the codes of symbols if there is a gain in the frequency of the string "ab" and this gain should be at most delta and the closeness should not be worse than before (means the new delta value should be less than the previous value, then only we will interchange). If the symbol is participated in this step then it is locked to stop participating further. Repeat this step until there are no symbols to be covered.

Step3: Go for the codes of symbols with same code length and which are having same beginning but not ending code symbols and the string "ab" as sub string in one or more places. Interchange the codes of symbols if there is a gain in the frequency of the string "ab" and this gain should be at most delta and the closeness should not be worse than before (means the new delta value should be less than the previous value, then only we will interchange). If the symbol is participated in this step then it is locked to stop participating further. Repeat this step until there are no symbols to be covered.

Step4: Go for the codes of symbols with same code length and which are having same ending but not beginning code symbols and the string "ab" as sub string in one or more places. Interchange the codes of symbols if there is a gain in the frequency of the string "ab" and this gain should be at most delta and the closeness should not be worse than before (means the new delta value should be less than the previous value, then only we will interchange). If the symbol is participated in this step then it is locked to stop participating further. Repeat this step until there are no symbols to be covered.

Step5: Go for the codes of symbols with same code length and which are having different beginning and ending code symbols and the string "ab" as sub string in one or more places. Interchange the codes of symbols if there is a gain in the frequency of the string "ab" and this gain should be at most



delta and the closeness should not be worse than before (means the new delta value should be less than the previous value, then only we will interchange). If the symbol is participated in this step then it is locked to stop participating further. Repeat this step until there are no symbols to be covered.

Step6: If the string with minimum frequency is "00", then go for the symbol code among the symbol codes ending with code symbol 0 and having the frequency close to delta and extend the code of the symbol with padding code symbol 0 and closeness should not be worse than before (means the new delta value should be less than the previous value). Then go for the next maximum frequency symbol code and apply this step, repeat this step until there is no further improvement in getting closeness.

Step7: If the string with minimum frequency is "11", then go for the symbol code among the symbol codes ending with code symbol 1 and having the frequency close to delta and extend the code of the symbol with padding code symbol 1 and closeness should not be worse than before (means the new delta value should be less than the previous value.) Then go for the next maximum frequency symbol code and apply this step, repeat this step until there is no further improvement in getting closeness.

Step8: If the string with minimum frequency is "01", then go for the symbol code among the symbol codes ending with code symbol 0 and having the frequency close to delta and extend the code of the symbol with padding code symbol 1, if the new value of delta is less than the old value of delta. . Then go for the next maximum frequency symbol code and apply this step, repeat this step until there is no further improvement in getting closeness.

Step9: If the string with minimum frequency is "10", then go for the symbol code among the symbol codes ending with code symbol 1 and having the frequency close to delta and extend the code of the symbol with padding code symbol 0, if the new value of delta is less than the old value of delta. Then go for the next maximum frequency symbol code and apply this step, repeat this step until there is no further improvement in getting closeness.

### 3. Case Study:

OUTPUT 1:

-----

Input message is : aabbccbc

Huffman code is: a----10, b----11, c----0.

The length of the plaintext and strings(cipher text) are: 8 13

cipher text is:1010111100110

In cipher text frequencies of strings "00", "01", "10", "11":

(0	0)	----	1
(0	1)	----	3
(1	0)	----	4
(1	1)	----	4

End of padding Algorithm Cipher text is:  
1010111100001100

End of padding Algorithm , in cipher text the frequencies of strings "00", "01", "10", "11":

(0	0)	----	4
(0	1)	----	3
(1	0)	----	4
(1	1)	----	4

End of padding Algorithm Codes of input alphabet are :

a----10,  
b----11,  
c----00

For this example in the initial stage the cipher text length is 13 and after making the frequencies of strings "00", "01", "10", "11" close together, the cipher text length is 16.

So, the average code length initially is  $13/8$  i.e. 1.624.

the average code length after applying padding algorithm is  $16/8$  i.e. 2

-----  
OUT PUT 2:  
-----

INPUT (MESSAGE) :

makingtheencodedtextofaninstantaneouscodeasrandomas possiblehereapaddingalgorithmisdesignedwhichtakesany instantaneouscodeasinputandoutputsthecorrespondingencoder's scheme which we are looking for

HUFFMAN CODE:

110-----a  
11111110-----b  
00110-----c  
1010-----d  
010-----e  
111110-----f  
00111-----g  
1110-----h  
0110-----i  
101110-----k  
101111-----l  
011111-----m  
100-----n  
0000-----o  
11110-----p  
01110-----r

```

0001-----s
0010-----t
10110-----u
011110-----w
1111110-----x
11111111-----y

```

The length of the plaintext and cipher text are initially(before applying padding algorithm): 186  
757

Average code length initially:  $757/186$  i.e. 4.067

Cipher text initially(before applying padding algorithm):

```

011111110101110011010000111001011100100101000011000
001010010101000100101111110001000001111101101000110
100000100101101000010110100010000010110000100110000
01010010110000101110110100101000000111111000011111
000000001000101101111111010111101011100100111001011
011110110101010100110100001111101011110011100000111
001100010111001111101100001101001000010110100010101
001111011100110001101110001011010111001000011101001
111111101101000001001011010000101101000100000101100
001001100000101001011000010110100111101011000101101
001010000010110001011110101100010000100101110010001
100000011100111001000011111000001001010011010000111
010100001100000101001010000111000100110111001001111
101001111011100110001101110011110010110011100101011
1100000000101110011010000111111110000001110

```

In cipher text the frequencies of strings "00", "01", "10", "11" initially:

```

(0 0)-----210
(0 1)-----185
(1 0)-----185
(1 1)-----176

```

In cipher text the frequencies of strings "00", "01", "10", "11" after applying Padding algorithm:

(0	0)	193
(0	1)	196
(1	0)	195
(1	1)	204

After applying the Padding algorithm , the cipher text is:

```
011111110111110101101001011010010111001001010011110
100001010010101000100101111110001000001011101101000
110100000110010110100001011010001000000011000011111
101000010100101100001100111110100101000000111111100
001101110000000011000110110111111101011110101110010
001110101100111011010101010011010010110111010111110
110100000011101100010111001111101100001110100100001
101101000101010011110111001101111011110001011011111
01010000111101001111111011010000011001011010000101
101000100000001100001111110100001010010110000110110
100011100011000101101001010000000110001001110001100
010000110010111001011110100000011100111010000110111
000001001010011010010110101010011110100001010010100
101101000111111011110010011111010011110111001101111
011110011110010110001110101011110000000011111010110
1001011011011100000000111
```

Average code length, after applying the Padding algorithm =  $788/186 = 4.236$

OUTPUT 3:

-----

INPUT (MESSAGE) :

thedocumentsdistributedherehavebeenprovidedasameans  
toensuretimelydisseminationofscholarlyandtechnicalw

orkonanoncommercialbasiscopyrightandallrightstherei  
naremaintainedbytheauthorsorbyothercopyrightholders  
notwithstandingthattheyhaveofferedtheirworkshereele  
ctronicallyitisunderstoodthataallpersonscopyingthisi  
nformationwilladheretothetermsandconstraintsinvoked  
byeachauthorscopyrighttheseworksmaynotberepostedwit  
houttheexplicitpermissionofthecopyrightholdercomple  
xityandalgorithmsforreasoningabouttimeagraphtheoret  
icapproachmpaperdealswithproblemsinreasoningaboutsuc  
h intervalswhentheprecisetopologicalrelationshipbet  
weenthemisunknownonlypartiallyspecifiedthisworkunif  
iesnotionsofintervalalgebrasinartificialintelligenc  
ewiththoseofintervalordersandintervalgraphsincombin  
atorics

HUFFMAN CODE:

1000-----a  
101111-----b  
11110-----c  
10110-----d  
010-----e  
011110-----f  
111110-----g  
0110-----h  
000-----i  
11111111-----k  
11010-----l  
110110-----m  
1110-----n  
1100-----o  
01110-----p  
1001-----r  
1010-----s  
001-----t  
011111-----u  
1111110-----v  
101110-----w  
11111110-----x  
110111-----y

The length of the plaintext and cipher text are initially(before applying padding algorithm): 772  
3225

Average code length initially:  $3225/772$  i.e. 4.177

In cipher text the frequencies of strings "00", "01", "10", "11" initially:

(0 0)-----666  
(0 1)-----843  
(1 0)-----843  
(1 1)-----872

In cipher text the frequencies of strings "00", "01", "10", "11" after applying Padding algorithm:

(0 0)-----811  
(0 1)-----865  
(1 0)-----865  
(1 1)-----870

Average code length after applying padding algorithm:  $3412/772$  i.e. 4.42

## **4. Conclusion**

We discussed the padding algorithm achieves the closeness of strings "00", "01", "10", "11". Observe that steps 1 to 9 in padding algorithm are applied to each string. After applying to all once, we can apply again the steps 1 to 9 to get more closeness if possible. This sets trade off between randomness and cost afforded. we could stop the algorithm anywhere when we are satisfied with the achieved closeness about the strings or there is no further improvement in getting closeness about the strings.

The above algorithm is in greedy approach, so there may be effect on further gain in getting closeness by the steps we have taken before.

As we have seen in case study, the average code length is increasing by 5% of the initial average code length on approximate, after applying this padding algorithm.



## Bibliography

- [1] Knuth, Fundamental Algorithms (The Art of Computer Programming Volume 1) Second Edition: Narosa Publishing House.
- [2] Richard Wesley Hamming, Coding and information theory: Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632
- [3] F.J.Mac Williams and N.J.A.Sloane, The Theory of Error Correcting Codes Part I: North Holland publishing company, Amsterdam, New York, Oxford.
- [4] Douglas R.Stinson, Cryptography: Theory and Practice: CRC Press Boca Raton London Tokyo
- [5] Steven Roman, Coding and Information theory: Springer-Verlag.
- [6] Robert M.Fano, Transmission of Information: M.I.T. Press and John Wiley and Sons.