IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 5, May 2015.
www.ijiset.com
ISSN 2348 – 7968

Reliable Transmission of Information Over Channels Using Cryptography and Convolutional Codes

Mahejabeen.Ilkal[1]
Electrical & Electronics dept,Bldea's college of enggineering & technology,Vijaypur, India 586101

ABSTRACT:**"Cryptography is about concealing information, and coding theory is about revealing it" Despite these apparently conflicting goals, the two fields have common origins and many interesting relationships.Shannon was the first to give cryptography a formal, mathematical treatment, focusing primarily on the task of encryption. In his model of cryptosystem, Shannon defined secrecy in terms of the amount of information a cipher-text reveals about its underlying message. To obtain perfect secrecy, a cryptosystem would need to reveal zero information about an encrypted message. A central problem of coding theory is reliable communication over an unreliable channel. All solutions to this problem, in some form or another, rely on the basic idea of encoding message with some redundancy, allowing the receiver to detect and correct whatever errors may arise during transmission through the channel. The main goal is to minimize the amount of redundancy while maximizing the quantity of errors that can be corrected.**

Key words: Plain text, Cipher text, S-DEC, BEC, Viterbi, Trellis and Convolutional Code.

1.INTRODUCTION

1.1Overview

The requirements of information security within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipments, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An example of the former is the use of rugged filling cabinets with the combination lock for storing sensitive documents. An example of the latter is personnel screening procedures used during the hiring process. With the introduction of computers, the need for automated tools for protecting files and other information stored on the computer became evident. The generic name for the collection of tools designed to protect data and to thwart hackers is computer security. Second major change that affected security is the introduction of distributed systems and the use networks and communication facilities for carrying data between terminal user and the computer and between computer and computer. Network security measures are needed protect data during their transmissions[1][2][3]. It is in this quest for a simple solution is to provide secure transmission of messages, that this project was under taken, its main objective is to allow two or more persons can communicate in a way that

guarantees that thedesired subset of the following four primitives,
1. Confidentiality
2. Data integrity
3. Authentication
4. Non-repudiation

1.2 Block diagram

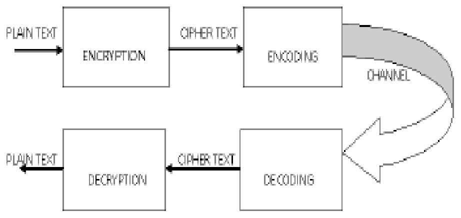


Fig 1: Flow diagram

1.3 Proposed Project Work

In this age of universal electronic connectivity, of the viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security does not matter. It is in this quest for a simple solution to this problem, that this project was undertaken. The Proposed text to Conventional crypto-coding has following features,
1.The data cannot be accessed forunauthorized use.
2.The content of the data frames is hidden.
3.The authenticity of the data can be established.
4.The undetected modification of the data is avoided.
5.The data cannot be disowned by the originator of the message.

2. SIMPLIFIED DATA ENCRYPTION STANDARD

2.1(S-DES Scheme)

The overall structure of the Simplified Data Encryption Standard is as shown in figure 1, which we will refer to as S-DES. The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of cipher-text as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that cipher text as input and produces the original 8-bit block of plaintext.

$$IP^{-1} \circ f_{k2} \circ SW \circ f_{k1} \circ IP$$

this can also be written as:

$$Ciphertext = IP^{-1}(f_{k2}(SW(f_{k1}(IP(\text{plaintext})))))$$

Where
K_1= P8 (Shift (P10 (key)))
K_2= P8 (Shift (Shift (P10 (key))))
Decryption is also shown in Figure 1 and is essentially the reverse of encryption:

$$plaintext = IP^{-1}(f_{k2}(SW(f_{k1}(IP(cipher\text{text})))))$$

First, permute the key in the following fashion. Let the 10-bit key be designed as(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10)
Then the permutation P10 is defined as
P10(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10)=(k2,k4,k6,k8,k10,k1,k3,k5,k7)

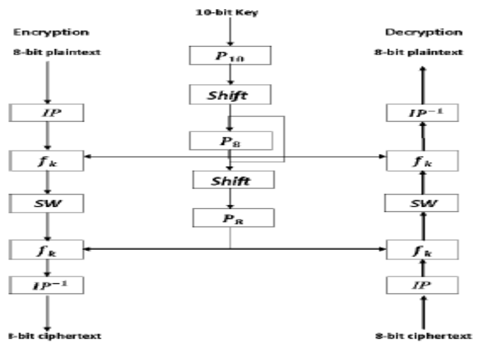


Fig 2:Simplified DES Scheme

2.2 S-DES Key generation

S-DES depends on the use of a 10-bit key shared between sender and receiver. From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm. Figure depicts the stages followed to produce the subkeys[5].

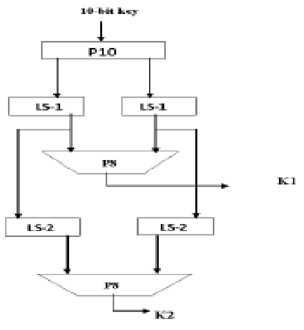


Fig 3:SDES key generation

2.3 S-DES Encryption

The S-DES encryption algorithm takes an 8- bit block of plaintext and a 10-bit key as input and produces an 8bit block of cipher-text as output. Encryption involves the sequential application of five functions.

1. An initial permutation (IP).
2. A complex function labeled f_k, which involves both permutation and substitution operations and depends on a key input.
3. A simple permutation function that switches (SW) the two halves of the data.
4. The function f_k again.
5. A permutation function that is the inverse of the initial permutation IP^{-1}

2.3.1 Initial and Final Permutations

The input to the algorithm is an 8-bit block of plaintext, which we first permute using the IP function:

IP							
2	4	6	8	1	3	5	7

This retains all 8 bits of the plaintext but mixes them up. At the end of the algorithm, the inverse permutation is used:

IP^{-1}							
5	1	6	2	7	3	8	4

It is easy to show by example that the second permutation is indeed the reverse of the first; that is,

$$IP^{-1}(\mathrm{IP}(X)) = X.$$

2.3.2 The Function f_k:

The most complex component of S-DES is the function f_k, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows. Let *L* and *R* be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to f_k,, and let F be a mapping (not necessarily one toone) from 4-bit strings to 4-bit strings. Then

$$f_k(L,R) = (L \oplus F(R,SK)R)$$

Where, *SK* is a subkey and _ is the bit-by-bit exclusive-OR function.

The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2- bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

$$\mathbf{S0} = \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix}$$

$$S1 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix}$$

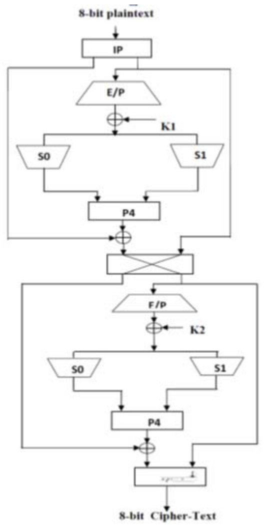
The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the S box. The entry in that row and column, in base 2, is the 2-bit output.

2.3.3 The Switch Function

The function fkonly alters the leftmost 4 bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of fkoperates on a different 4 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same.

2.3.3 The Switch Function

The function f_k only alters the leftmost 4bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of f_k operates on a different 4 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same.



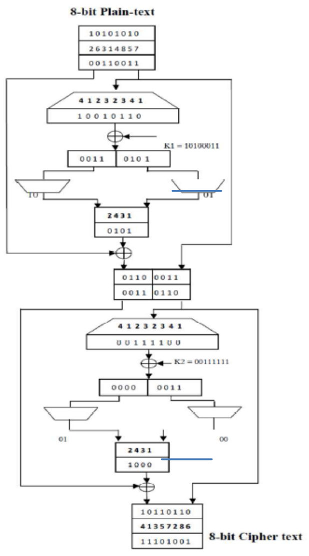


Fig 4:Example for encryption

2.4 S-DES Decryption

The S-DES decryption algorithm takes an 8- bit block of cipher-text and a 10-bit key as input and produces an 8-bit block of plain-text as output. Like Encryption, Decryption also involves the sequential application of five functions.

1. An initial permutation (IP).
2. A complex function labeled fk, which involves both permutation and substitution operations and depends on a key input.
3. A simple permutation function that switches (SW) the two halves of the data.
4. The function fk again.
5. A permutation function that is the inverse of the initial permutation (IP-1).

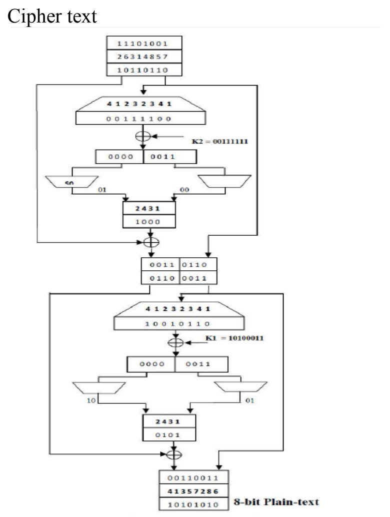
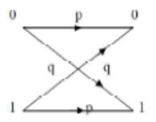


Fig 5:Example of S-DES Decryption

3. BINARY SYMMETRIC CHANNEL

The BSC is a binary channel; that is, it can transmit only one of two symbols (usually called 0 and 1). (A non-binary channel would be capable of transmitting more than 2 symbols, possibly even an infinite number of choices) The transmission is not perfect, and occasionally the receiver gets the wrong bit. This channel is often used by theorists because it is one of the simplest noisy channels to analyze. Many problems in communication can be reduced due to a BSC[10]. On the other hand, being able to transmit effectively over the BSC can give rise to solutions for more complicated channels. A binary symmetric channel with crossover probability p is a channel with binary input and binary output and probability of error p; that is, if X is the transmitted random variable and Y the received variable, then the channel is characterized by the conditional probabilities as shown in the channel diagram.



$$P(Y \mid X) = \begin{bmatrix} p & q \\ q & p \end{bmatrix}$$

Fig 6:Channel diagram and channel matrix

It is assumed that $0 \leq p \leq 1/2$. If $p>1/2$, then the receiver can swap the output (interpret 1 when it sees 0, and visa versa) and obtain an equivalent channel with crossover probability $1-p \leq 1/2$.

4. CONVOLUTIONAL CODES

Convolutional codes were first introduced by by Elias in 1955 as an alternative to block codes. Shorty there after Woaencraft proposed sequential decoding as an efficient decoding scheme for convolutional codes and experimental studies soon began to appear. In 1963, Massey proposed a less efficient but simpler to implement decoding method called threshold decoding. Then in 1967 Viterbi proposed a maximum likelihood decoding scheme that was relatively easy to implement for codes with small memory orders. This scheme called Viterbi decoding.

In “convolutional codes”, a block of n code digits generated by the encoder in a time unit depends not only the block of ‘k’ message digits within that time unit, but also on the preceding (m-1) blocks of message digits (m>1). Usually the values of k and n will be small.Like block codes, convolutional codes can be designed to either detect or correct errors. However, block codes are better suited for error detection and convolutional codes for error correction.Encoding for convolutional codes can be accomplished using simple shift registers and several practical procedures have been developed for decoding[6][9].

4.1 Encoder for Convolutional Codes

A convolutional encoder takes sequences of message digits and generates sequences of code digits. It is a finite state machine (FSM), processing information bits in a serial manner. Thus the generated code is a function of input and the states of the FSM.

1. ‘k’ information bits are encoded to a block of ‘n’ bits (n>k)
2. The n-digit code block depends on
• ‘k’ information bits, and
• previous (m-1) information bits
3. The generated code is called an (n, k, m) convolutional code of constraint length “nm” digits and the rate efficiency will be “k/n”.
4. The encoder consists of shift registers and mod-2 adders.

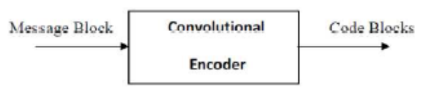


Fig 7:Encoder

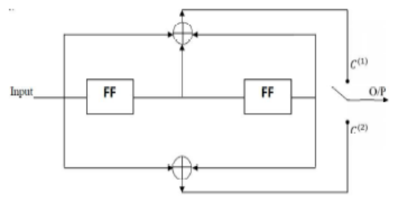


Fig 8:(2,1,2) convolutional encoder

4.2 Encoding of a Convolutional Code

Encoding of convolutional code is done by two approaches
1. Time domain approach
    1. Convolutional method
    2. Matrix method
2. Transfer domain approach

4.2.1 Time domain approach by Matrix method

The time-domain behaviour of a binary convolutional encoder may be defined in terms of a set of “n impulse responses”. Let the sequence $[\ g_1^{(1)}\ \ g_2^{(1)}\ g_3^{(1)} \ldots\ldots\ g_{m+1}^{(1)}]$ denote the “impulse response”also called “generator sequences” for the top adder and the sequence $[\ g_1^{(2)}\ \ g_2^{(2)}\ g_3^{(2)} \ldots\ldots g_{m+1}^{(2)}]$ for thebottom adder, can be interlaced and arranged in a matrix form with ,Number of rows = number of digitsin the message sequence = L rows ,Number ofcolumns = n (L + m).Such a matrix of order [L] x [n(L + m)] is called “generator matrix” of theconvolutional encoder.

$$G = \begin{bmatrix} g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & \cdots & g_{m+1}^{(1)} g_{m+1}^{(2)} & 0\,0 & 0\,0 & \cdots & 0\,0 \\ 0\,0 & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \cdots & g_m^{(1)} g_m^{(2)} & g_{m+1}^{(1)} g_{m+1}^{(2)} & 0\,0 & \cdots & 0\,0 \\ 0\,0 & 0\,0 & g_1^{(1)} g_1^{(2)} & \cdots & \cdots & g_m^{(1)} g_m^{(2)} & g_{m+1}^{(1)} g_{m+1}^{(2)} & \cdots & 0\,0 \\ .. & .. & .. & .. & .. & .. & .. & \cdots & .. \\ .. & .. & .. & .. & .. & .. & .. & \cdots & g_{m+1}^{(1)} g_{m+1}^{(2)} \end{bmatrix}$$

In the second row, the number of 0’s is equal to the number of modulo-2 adders. Since the generator matrix G has n (L + m) number of columns, the encoder output will have n (L + m) number of bits given by
C = d G
In the encoder above,
d = 1 0 1 1 1
$g^{(1)} = 1\ 0\ 1\ 1$
$g^{(2)} = 1\ 1\ 1\ 1$

The generator matrix has L=5 rows and n(L + m) = 2 (5 + 3) = 16 columns given by

$$G=\begin{bmatrix} 11 & 01 & 11 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 01 & 11 & 11 \end{bmatrix}$$

Encoder output is given by,
C = d G

$$C=[10111]\begin{bmatrix} 11 & 01 & 11 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 01 & 11 & 11 \end{bmatrix}$$

C = [11, 01, 00, 01, 01, 01, 00, 11]

State Transition Table

Present state	Binary Description	Input	Next State	Binary Description	d_l	d_{l-1}	d_{l-2}	Output $C^{(1)}\ C^{(2)}$
S0	0 0	0	S0	00	0	0	0	0 0
		1	S1	10	1	0	0	1 1
S1	1 0	0	S2	01	0	1	0	1 0
		1	S3	11	1	1	0	0 1
S2	0 1	0	S0	00	0	0	1	1 1
		1	S1	10	1	0	1	0 0
S3	1 1	0	S2	01	0	1	1	0 1
		1	S3	11	1	1	1	1 0

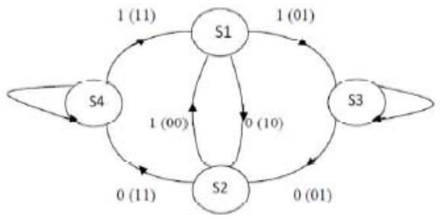


Fig 9:state diagram

5. VITERBI DECODING

A Viterbi decoder uses the Viterbi algorithm for decoding a bitstream that has been encoded using Forward error correction based on a Convolutional code[7][9]. The Viterbi algorithm is the most resource-consuming, but it does the maximum likelihood decoding.

The algorithm is based on the nearest neighbour decoding scheme and, like the other algorithms we have looked at, it relies on the assumption that the probability of t errors is much greater than the probability of t+1 errors and it thus selects or chooses and retains only the paths which have fewer errors. The Viterbi algorithm was conceived by Andrew Viterbi in 1967 as a decoding algorithm for convolutional codes over noisy digital communication links. The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, keyword spotting, computational linguistics, and bioinformatics.

5.1 Parts of Viterbi algorithm

A Viterbi algorithm consists of the following three major parts:
1. Branch metric calculation – calculation of a distance between the input pair of bits and the four possible "ideal" pairs ("00", "01","10", "11").
2. Path metric calculation – for every encoder state, calculate a metric for the survivor path ending in this state (a survivor path is a path with the minimum metric).
3. Traceback – this step is necessary for hardware implementations that don't store full information about the survivor paths, but store only one bit decision every time when one survivor path is selected from the two.

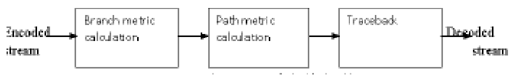


Fig 10: parts of viterbi algorithm

5.1.1 Branch Metric Calculation

Methods of branch metric calculation are different for hard decision and soft decision decoders.For a hard decision decoder: a branch metric is a Hamming distance between the received pair of bits and the "ideal" pair. Therefore, a branch metric can take values of 0, 1 and 2. Thus for every input pair we have 4 branch metrics (one for each pair of "ideal" values).For a soft decision decoder: a branch metric is measured using the Euclidean distance. Let x be the first received bit in the pair, y – the second,x0 and y0 – the "ideal" values. Then branch metric is

$$M_b = (x - x_0)^2 + (y - y_0)^2$$

Furthermore, when we calculate 4 branch metric for a soft decision decoder, we don't actually need to know absolute metric values – only the difference between them makes sense.

5.1.2 Path Metric Calculation

Path metrics are calculated using a procedure called ACS (Add-Compare-Select). This procedure is repeated for every encoder state.
1. Add – for a given state, we know two states on the previous step which can move to this state, and the output bit pairs that correspond to these transitions. To calculate new path metrics, we add the previous path metrics with the corresponding branch metrics.
2. Compare, select – we now have two paths, ending in a given state. One of them (with greater metric) is dropped.

As there are 2^{k-1} encoder states, we have 2^{k-1} survivor paths at any given time.It is important

that the difference between two survivor path metrics cannot exceed Δ log(k-1) , where δ is a difference between maximum and minimum possible branch metrics.The problem with path metrics is that they tend to grow constantly and will eventually overflow. But, since the absolute values of path metric don't actually matter, and the difference between them is limited, a data type with a certain number of bits will be sufficient.
There are two ways of dealing with this problem:
1. Since the absolute values of path metric don't actually matter, we can at any time subtract an identical value from the metric of every path. It is usually done when all path metrics exceed a chosen threshold (in this case the threshold value is subtracted from every path metric). This method is simple, but not very efficient when implemented in hardware.
2. The second approach allows overflow, but uses a sufficient number of bits to be able to detect whether the overflow took place or not. The compare procedure must be modified in this case.

5.1.3 Traceback

It has been proven that all survivor paths merge after decoding a sufficiently large block of data, i.e. they differ only in their endings and have the common beginning.If we decode a continuous stream of data, we want our decoder to have finite latency. It is obvious that when some part of path at the beginning of the graph belongs to every survivor path, the decoded bits corresponding to this part can be sent to the output. Given the above statement, we can perform the decoding as follows:
1. Find the survivor paths for N+D input pairs of bits.
2. Trace back from the end of any survivor paths to the beginning.
3. Send N bits to the output.
4. Find the survivor paths for another N pairs of input bits.
5. Go to step 2

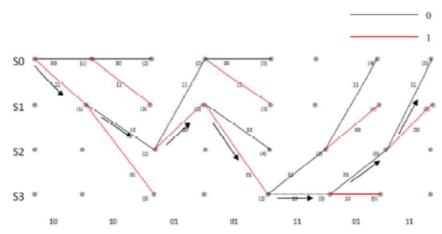
In these procedure D is an important parameter called decoding depth. A decoding depth should be considerably large for quality decoding, no less then 5K. Increasing D decreases the probability of a decoding error, but also increases latency.

5.2 The Algorithm

Step 1: Starting at level (i.e. Time unit) j = m, compute the partial metric for the single path entering 8each node (state). Store the path (the survivor) and its metric for each state.
Step 2: Increment the level j by 1. Compute the partial metric for all the paths entering a state by adding the branch metric entering that state to the metric of the connecting survivor at the preceding time unit. For each state, store the path with the largest metric (the survivor) , together with its metric and eliminate all the other paths.
Step 3: If j < (L + m), repeat step 2. Otherwise stop. The number of nodes at any level of trellis does not continue to grow as the number of incoming message bits increases, instead, it remains a constant at__$. There are __survivors from time unit 'm' upto the time unit L, one for each of the __ states. After L time units, there are fewer survivors, since there are fewer states while the encoder is returning to the allzero states. Finally, at time unit (L + m) there is only one state, the all-zero state and hence only one survivor and the algorithm terminates.

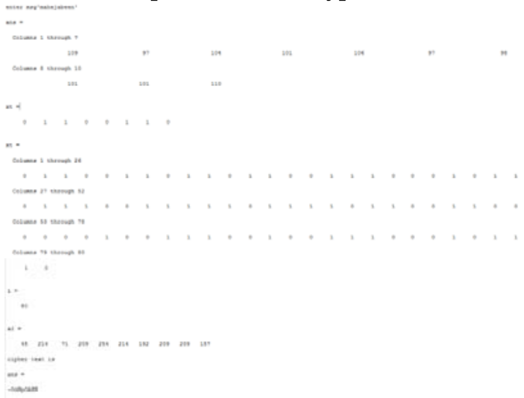
5.3 Trellis

A convolutional encoder is often seen as a finite state machine. Each state corresponds to some value of the encoder's register. Given the input bit value, from a certain state the encoder can move to two other states. These state transitions constitute a diagram which is called a trellis diagram. Trellis is a tree like structure with remerging branches[8].
A trellis diagram for the code on the Figure 2 is depicted on the Figure 3. A solid line corresponds to input 0, a dotted line – to input 1 (note that encoder states are designated in such a way that the rightmost bit is the newest one). Each path on the trellis diagram corresponds to a valid sequence from the encoder's output. Conversely, any valid sequence from the encoder's output can be represented as a path on the trellis diagram. Note that each state transition on the diagram corresponds to a pair of output bits. There are only two allowed transitions for every state, so there are two allowed pairs of output bits, and the two other pairs are forbidden. If an error occurs, it is very likely that the receiver will get a set of forbidden pairs, which don't constitute a path on the trellis diagram. So, the task of the decoder is to find a path on the trellis diagram which is the closest match to the received sequence.
Trellis Diagram
Message Sequence = [10111]
Encoded Sequence = [11100001100111]
Received Sequence from channel = [10100101110111]



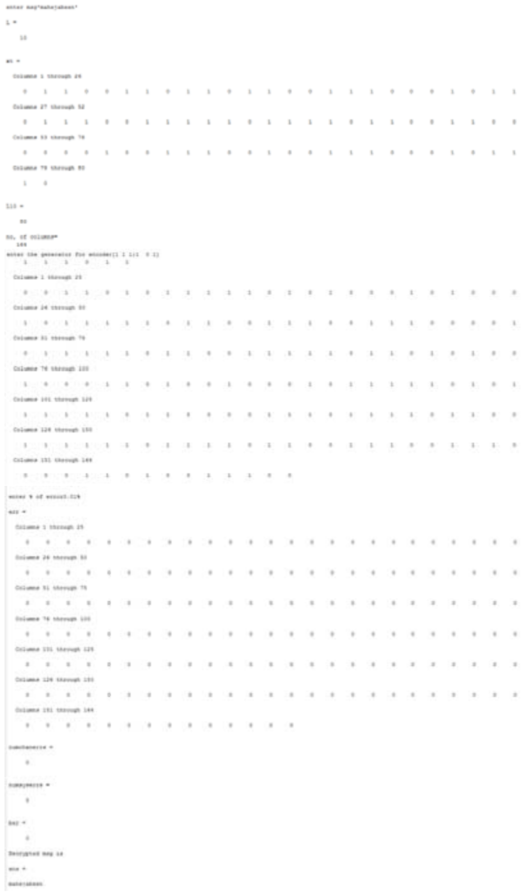
6. EXPERIMENTAL RESULTS

6.1 codede sequence with encrypted data



```
enter msg'mahejabeen'

ans =

  Columns 1 through 7

         109          97         104         101         106          97          98

  Columns 8 through 10

         101         101         110

xt =

     0     1     1     0     0     1     1     0

xt =

  Columns 1 through 26

     0     1     1     0     0     1     1     0     1     1     0     1     1     0     0     1     1     1     0     0     0     1     0     1     1

  Columns 27 through 52

     0     1     1     1     0     0     1     1     1     1     1     0     1     1     1     1     0     1     1     0     0     1     1     0     0

  Columns 53 through 78

     0     0     0     0     1     0     0     1     1     1     0     0     1     0     0     1     1     1     0     0     0     1     0     1     1

  Columns 79 through 80

     1     0

l =

    80

xl =

    65   214    71   209   254   214   192   209   209   157

cipher text is

ans =

[illegible]
```

6.2 Decoder result



```
enter msg'mahejabeen'

l =

    10

xt =

  Columns 1 through 26

     0     1     1     0     0     1     1     0     1     1     0     1     1     0     0     1     1     1     0     0     0     1     0     1     1

  Columns 27 through 52

     0     1     1     1     0     0     1     1     1     1     1     0     1     1     1     1     0     1     1     0     0     1     1     0     0

  Columns 53 through 78

     0     0     0     0     1     0     0     1     1     1     0     0     1     0     0     1     1     1     0     0     0     1     0     1     1

  Columns 79 through 80

     1     0

l11 =

    80

no. of columns=
   164
enter the generator for encoder[1 1 1;1  0 1]
   1    1    1    0    1    1

  Columns 1 through 25

     0     0     1     1     0     1     0     1     1     1     1     1     0     1     0     1     0     0     0     1     0     1     0     0     0

  Columns 26 through 50

     1     0     1     1     1     1     1     0     1     1     0     0     1     1     1     0     0     1     1     1     0     0     0     0     1

  Columns 51 through 75

     0     1     1     1     1     1     0     1     1     0     0     1     1     1     1     1     0     1     1     0     1     0     1     0     0

  Columns 76 through 100

     1     0     0     0     1     1     0     1     0     0     1     0     0     0     1     0     1     1     1     1     1     0     1     0     1

  Columns 101 through 125

     1     1     1     1     1     0     1     1     0     0     0     0     1     1     1     0     0     1     1     1     0     1     1     0     0

  Columns 126 through 150

     1     1     1     1     1     1     0     1     1     1     1     0     1     1     0     0     1     1     1     0     0     1     1     1     0

  Columns 151 through 164

     0     0     0     1     1     0     1     0     0     1     1     1     0     0

enter % of error0.01%

err =

  Columns 1 through 25

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 26 through 50

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 51 through 75

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 76 through 100

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 101 through 125

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 126 through 150

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 151 through 164

     0     0     0     0     0     0     0     0     0     0     0     0     0     0

numchanerrs =

     0

numsyserrs =

     0

ber =

     0

Decrypted msg is

ans =

mahejabeen
```

7. CONCLUSION

The project relies on the concept of cryptography and coding theory, together which provides secure and reliable communication between the systems. The main idea is to enhance the confidentiality and integrity of the data being transmitted. Cryptography is the science of protecting data, which provides means and methods of converting data into unreadable form. Security related transformation is done using S-DES encryption and decryption algorithm so as to convert the message into cipher-text and vice-versa. Coding theory is the process of adding redundant information to the data being transmitted. The reason why we are using channel coding is because the transmitted signal are often corrupted by channel noise and we can use channel coding to correct these errors so as to achieve the reliable transmission of information over a large varieties of channels. Using convolutional codes together with Viterbi algorithm can provide a large coding gain and good performance. As the percentage of error introduced is increased, the BER becomes nonzero and the decrypted message is not exactly same as the input. In such cases, number of trials are to be made to decode the input message.

References

[1] Dr.P.S.Sathyanarayana,"probability,Information and Coding Theory",First Edition,1996.

[2] William stallings,"Crytography and Network security", Third Edition,Pearson Education,2003.

[3] WWW.1-core.com

[4] Ling Bin; Liu Lichen; Zhang Jan," Image encryption algorithm based on chaotic map and S-DES ",Advanced Computer Control (ICACC), 2010 2nd International Conference on, Print ISBN: 978-1-4244-5845-5, Issue Date: 27-29 March 2010

[5] Fu Li; Pan Ming," A Simplified FPGA Implementation Based on an Improved DES Algorithm , Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, Print ISBN: 978-0-7695-3899-0, Issue Date: 14-17 Oct. 2009

[6] Rosenthal, J. Schumacher, J.M. York, E.V, On behaviors and convolutional codes , Information Theory, IEEE Transactions on, Issue Date: Nov 1996 Volume: 42 Issue: 6 On page(s): 1881 – 189

[7] Onyszchuk, I.M," Truncation length for Viterbi decoding ",Communications, IEEE Transactions on Issue Date: Jul 1991 ,Volume: 39 Issue: 7 ,On page(s): 1023 – 1026 ,ISSN: 0090-6778

[8]Filler, T.; Judas, J.; Fridrich, J.," Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes"Information Forensics and Security, IEEE Transactions on ,Issue Date:Sept. 2011 ,Volume: 6 Issue:3 ,On page(s) 920 935 ,ISSN:1556-6013

[9]Forney, G.D., Jr," The viterbi algorithm" Proceedings of the IEEE Issue Date: March 1973 Volume: 61 Issue:3 On page(s): 268- 278

[10]Liveris A.D.; Zixiang Xiong; Georghiades,

C.N.; “Distributed compression of binary sources using conventional parallel and serial concatenated convolutional codes “,Data Compression Conference, 2003. Proceedings. DCC 2003 ,Issue Date: 25-27 March 2003 ,On page(s): 193 – 202 ,ISSN: 1068-0314 ,Print ISBN: 0-7695-1896-6

[11]Chilappagari, S.K.; Sankaranarayanan, S.; Vasic, B.; “Error Floors of LDPC Codes on the Binary Symmetric Channel, Communications, 2006. ICC '06. IEEE International Conference on ,Issue Date: June 2006 On page(s): 1089 - 1094 ,Location: Istanbul ,ISSN: 8164-9547