ISSN:2231-0711
www.ijcset.net

Encryption, Decryption and Error Detection using Finite State Transducers and their Inverses

Bhaskara Rao N
Dayananda Sagar College of Engineering
Bangalore-560078, India
bhaskararao.nadahalli@gmail.com

Abstract—**In this paper, we define the Inverse of a Finite State Transducer. The inverse and the corresponding Finite State Transducer are used for encryption and decryption of sequential data. The encryption process is designed for error detection capability during the decryption phase.**

Keywords-error detectiont capability;finite state transducer; Translational inverse;

1. INTRODUCTION

Finite State Transducers [1], [2] convert a given input sequence into another output sequence using a Finite State Machine. In this paper we are using the Moore model where the present output symbol depends only on the present state. Also, the Finite State Transducer (FST) is designed for invertibility. That is, the input sequence of the FST can be recovered from its output sequence using the inversion of the FST.

II. BASIC MODEL, SYMBOLS AND NOTATIONS

A. *Finite State Transducer Model*

The finite states of the FST model used in this paper are designated as integers 1,2,…,N instead of the conventional notation Q_0, Q_1,…, etc. Here, N is the number of available states. The temporal state of the system at i th iteration is represented by the discrete finite state variable, Q(i) for i=1,2,3,…etc. That is, the successive states are denoted by the variables Q(1), Q(2), …etc. The starting state is Q(1), the next state is Q(2) and so on. The domain of Q(i) is,

$$Q(i) \in \{1, 2, ..., N\} \qquad (1)$$

We adopt this convention so that the state transition rule for the FST can be represented by a state transition matrix instead of a state transition table and the domain of Q(i) can represent the rows of the State Transition Matrix.

The input sequence to this Finite State Transducer is represented by vector y, where,

$$y = [y(1) \quad y(2) \quad ... \quad y(L)] \qquad (2)$$

where, L is the length of the input sequence.

Here, y(i) belongs to the input set Σ which is made up of integers 1 to N instead of say alphabets as,

$$y(i) \in \{1, 2, ..., N\} \qquad (3)$$

Again, we choose the integer values for the y(i) domain so that it can represent the columns of the State Transition Matrix**.** Here the number of input symbols is N. In those cases where the input symbols to the FST are not integers, they are mapped into integers in the range {1:N}.

B. *State Transition Matrix T*

Let the state transition be represented by the matrix T of size NxN. The present state of the system (FST) depends on the previous state and the present input as,

$$Q(i) = T\left(Q(i-1), y(i)\right) \qquad (4)$$

for i=2,3,…,L, where L is the number of elements in the sequence y. For the sake of convenience the initial (starting) state Q(1) is taken as 1. That is,

$$Q(1) = 1 \qquad (5)$$

In Eq.(4), the previous state Q(i-1) is the row index and the present input y(i) is the column index of the matrix T. The corresponding matrix element T(Q(i-1),y(i)) gives the present state in terms of the previous state and the present input. Thus T is basically a mapping table.

C. *FST output sequence*

In the Moore model, the present output depends on the present state. Here, the output sequence x(i) is taken equal to Q(i) itself, for i = 1,2,…etc. Thus,

$$x(i) = Q(i) \qquad (6)$$

Therefore, the domain of x and Q are same.

Thus, $x(i) \in \{1, 2, ..., N\}$.

Substituting from Eq.(6) for Q in Eqs.(4) and (5), we get,

$$x(i) = T\left(x(i-1), y(i)\right) \qquad (7)$$

for i = 2,3,…,L, and

$$x(1) = 1 \text{ for } i = 1. \qquad (8)$$

Thus, Eqs.(7) and (8) determine the output sequence from the input sequence y of length L for the transition matrix T of size NxN.

D. *Size and elements of T*

Since the columns of T are specified by the values of y(i) which has a domain of {1:N}, the number of columns required is N. Among the column numbers 1 to N, the present column number exactly corresponds to the value of the present input y(i). For a given Q(i–1) and y(i), the element at row Q(i–1) and column y(i) in T gives the next state and output of the FST. This acts as the row index for the next iteration. Hence the number of rows of T should be N which represents the range of x. Also the element at location (Q(i–1),y(i)) represents the output. Therefore, the domain of the matrix element should also have to be {1:N}. This holds good for all the elements of T, because every element of T is covered by taking all possible combinations of (Q(i–1),y(i)). Thus the size of T is NxN and its elements belong to the domain {1:N}.

III. INVERTIBILITY AND INVERSION OF THE FINITE STATE TRANSDUCER

A. *Translational Inverse*

The FST should be designed for invertibility, in the sense, it should be possible to determine the input sequence of the FST from its output sequence. From Eq.(7), we see that x(i), x(i–1) and y(i) are related according to the mapping specified by T. The value of y(i) uniquely determines x(i) for a given x(i–1). Therefore, it should be possible to determine, that y(i), which corresponds to a specific x(i) and a given x(i–1). Therefore, y(i) can be expressed as,

$$y(i) = S\big(x(i-1), x(i)\big) \qquad (9)$$

Here, S is called as the *translational inverse* matrix of T. The matrix S is realised such that the element at x(i–1) th row and x(i) th column gives y(i). Thus, x(i–1) is the row index and x(i) is the column index of S that gives y(i).

B. *Size of S*

Since x(i–1), having the domain {1:N}, specifies the row number of S, the number of rows of S has to be N to take care of the N possible distinct values of x(i–1). Similarly, x(i) whose domain is {1:N} specifies the column numbers of S. Therefore, the number of columns of S should be N. Therefore the size of S is NxN.

C. *Derivation of S from T*

Consider Eq.(7). For matrix T, call the x(i–1) th row as r and y(i) th column as c. Then from Eq.(7),

$$T\big(r,c\big) = x(i) \qquad (10)$$

Here, $r = x(i-1)$ (11)

with $r \in \{1, 2, ..., N\}$

and $c = y(i)$ (12)

with $c \in \{1, 2, ..., N\}$

Using Eqs. (10), (11) and (12) in Eq.(9) yields,

$$S\big(r, T(r,c)\big) = c \qquad (13)$$

Let $d = T\big(r,c\big)$. Then Eq.(13) can be expressed as,

$$S\big(r,d\big) = c \qquad (14)$$

where $d = T\big(r,c\big)$ (15)

D. *Algorithm to realize S from T*

Determine the elements of S at row r and column d as follows.

```
Outer loop:   for r = 1,2,…, N
Inner Loop:       for c = 1,2,…, N
                      d = T(r,c);
                      S(r,d) = c;
                  end
              end
```

At the end of the inner loop, all elements of row r of S are assigned. The outer loop covers all the rows.

E. *Uniqueness of the row elements of T*

We see that the column indices of S are the elements of T in a given row. Since we cannot have duplicate column indices for a matrix, the elements of T cannot have duplicate values along a row. This condition should be satisfied for the proper realization of S. For example, consider the case of duplicate entries in a row of T as,

$$T(r, c_1) = T(r, c_2) = d \qquad (16)$$

Here, on row r, at two columns c_1 and c_2, the elemental values are same. Therefore, from Eqs.(13) and (16),

$$S\big(r,d\big) = c_1$$

and $$S\big(r,d\big) = c_2$$

The above equations cannot be satisfied simultaneously. They form an inconsistent system of equations. That is S(r,d) value cannot be determined uniquely. Therefore, *for the correct realization of S, the rows of T should not contain duplicate entries.*

F. *Uniqueness of the column elements of T*

Another constraint on T is, for any column of T, all the elements of that column should be unique. That is, there should not be any duplicate elements along any column of T. This requirement should be satisfied for the purpose of error detection as will be described later.

Thus there should not be any repeating elements along the rows of T and also along the columns of T. Since there are N elements in a row or column in the range {1:N}, the above constraint means that the rows and columns of T are permutations in the range {1:N}.

Under these conditions, S can be obtained from T as follows. In Eq.(13), let

$$T\big(r,c\big) = d \qquad (17)$$

Then we can rewrite Eq.(13) as,

$$S\big(r,d\big) = c \qquad (18)$$

where, $d = T\big(r,c\big)$

For a given row r, d is the row element of T at column location c. Since the elements of row r form a permutation in the range {1:N}, d covers all the values in this range. Therefore, from Eq.(18), all the column elements of S are determined for this r. In this way all the rows of S are determined when r takes values 1,2,…,N. From Eqs.(17) and (18), we can see that each row of S is the permutation inverse of the corresponding row of T and vice-versa.

G. *Invertibility of S*

From Eqs.(17) and (18) we see that c, the output of S, used as the input to T, gives d which is the corresponding input of S. Therefore T is the translational inverse of S. Thus the data encoded by S can be uniquely recovered by T and vice-versa. *T and S are the translational inverses of each other.*

H. *T and S Example*

Here, N=6 and each row of T is generated using the random permutation generation function *randperm(N)* [3]. While generating the next row of T, It is ascertained that it has no duplicate elements along any column. S is obtained from T using the algorithm described earlier. T and S thus obtained are shown below.

$$T = \begin{bmatrix} 4 & 5 & 2 & 6 & 3 & 1 \\ 1 & 2 & 3 & 5 & 6 & 4 \\ 2 & 6 & 5 & 1 & 4 & 3 \\ 3 & 4 & 6 & 2 & 1 & 5 \\ 6 & 3 & 1 & 4 & 5 & 2 \\ 5 & 1 & 4 & 3 & 2 & 6 \end{bmatrix} \qquad S = \begin{bmatrix} 6 & 3 & 5 & 1 & 2 & 4 \\ 1 & 2 & 3 & 6 & 4 & 5 \\ 4 & 1 & 6 & 5 & 3 & 2 \\ 5 & 4 & 1 & 2 & 6 & 3 \\ 3 & 6 & 2 & 4 & 5 & 1 \\ 2 & 5 & 4 & 3 & 1 & 6 \end{bmatrix}$$

IV. ENCRYPTION AND DECRYPTION USING S AND T

A. *Encryption using S*

The given data sequence to be encrypted is designated by $x(i)$ whose domain is $x(i) \in \{1:N\}$ for i = 1,2,…, L–1 where (L–1) is the length of the sequence. An additional element whose value is always 1 is appended at the end of the sequence for the purpose of error detection. Thus the total length of the augmented array x is L. Here after, x refers to the augmented array. The last element could have been any other value in the range {1:N}. But it is chosen as 1 for convenience. Thus for any input sequence,

$$x(L) = 1 \qquad (19)$$

Matrix S which has been determined from a specific T, is kept ready and is used for encryption to get the encrypted sequence y form x as given by Eq.(9) which is reproduced here.

$$y(i) = S\big(x(i-1), x(i)\big) \qquad (20)$$

for i = 2,3,…, L.

For convenience, y(1) is taken equal to x(1). That is,

$$y(1) = x(1) \qquad (21)$$

Eqs.(20) and (21) form the encryption equations. Here, $x(i)$ is the plain sequence and $y(i)$ is the cipher sequence. The cipher sequence is transmitted to the destination and decrypted using the T matrix which is already made available at the destination.

B. *Decryption using T*

The decryption equations are given by Eqs.(7) and (8) which are reproduced here as,

$$x(i) = T\big(x(i-1), y(i)\big) \qquad (22)$$

for i = 2,3,…, L, and

$$x(1) = y(1) \quad \text{for i=1.} \qquad (23)$$

During decryption, the original input sequence x is recovered because T is the translational inverse of S.

The basic block diagram is shown in Fig.1.

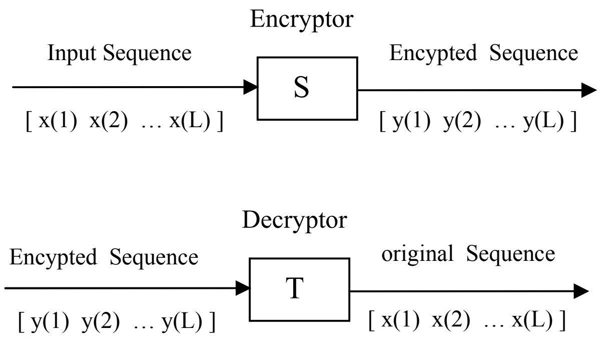

Fig.1. Basic Encryption and Decryption using S and T

V. ERROR DETECTION

The data sequence received by the destination is y(i) for i = 1,2,…, L. The sequence y(i) is decoded using Eq.(22). Suppose a single error has occurred at position i = k for certain k where k can be anywhere in the range 1 to L. That is, y(k) is the single error term in the received sequence. Therefore, all the terms before y(k) are error free. Hence all the terms decoded up to k are also error free. Therefore, the decoded sequence x(i) is error free for i = 1,2,…,(k-1). Let the original, error free value of y(k) be u(k). Because of the error, the value of y(k) is now different from u(k). Call this as v(k). That is,

$$\hat{y}(k) = v(k) \qquad (24)$$

The cap symbol on y(k) indicates that it is the error term. v(k) is different from the error free value u(k).

Consider the decoding result for i = k given by Eq.(22). When i = k, Eq.(22) gives,

$$\hat{x}(k) = T\big(x(k-1), \hat{y}(k)\big) \qquad (25)$$

The cap symbol on x(k) indicates that it is the response term for the error term $\hat{y}(k)$. From Eqs.(24) and (25),

$$\hat{x}(k) = T\big(x(k-1), v(k)\big) \qquad (26)$$

If there were no error, x(k) by Eq.(22) would have been,

$$x(k) = T\big(x(k-1), u(k)\big) \qquad (27)$$

where u(k) is the error free value of y(k).

From Eqs.(26) and (27), $x(k)$ and $\hat{x}(k)$ are two elements of T along the row $x(k-1)$ at two distinct column locations u(k) and v(k). Since the row elements of T are unique, $\hat{x}(k)$ would be different from x(k).

Now consider the (k+1) th decrypted value x(k+1) under the error condition with $\hat{x}(k)$ and y(k+1) as inputs. From Eq.(22) with i = k+1,

$$\hat{x}(k+1) = T\left(\hat{x}(k), y(k+1)\right) \qquad (28)$$

The error free output would have been,

$$x(k+1) = T\left(x(k), y(k+1)\right) \qquad (29)$$

In Eqs.(28) and (29), $\hat{x}(k+1)$ and x(k+1) are the elements of T in the common column y(k+1) at different row locations. Since the column elements of T are unique, (T is so designed.) $\hat{x}(k+1)$ and x(k+1) will be different. Now the error at symbol location k+1 causes the error at location k+2 and so on. Thus the error at location k propagates to the succeeding locations.

In this way, with error at location k, the encrypted sub sequence $[\,\hat{x}(k), \hat{x}(k+1), \hat{x}(k+2), ..., \hat{x}(L)\,]$ will be different from their respective error free values. This is an important property and can be stated as,

$$\hat{x}(j) \neq x(j) \qquad (30)$$

for j = k, k+1,…, L.

Hence the last element $\hat{x}(L)$ also would be different from x(L) which is always 1. Hence, with a single error at location k, the last decrypted element $\hat{x}(L)$ would be different from 1. This fact identifies the presence of an error. Thus, during decryption, if the last element is not equal to 1, it indicates the presence of an error.

VI. ERROR DETECTION EXAMPLE

Here, S and T matrices are as given in the Example of section III A, with N = 6. The plain text sequence to the encrypter is taken as,

$$x = \begin{bmatrix} 4 & 3 & 5 & 4 & 5 & 2 & 6 & 5 & 1 \end{bmatrix} \qquad (31)$$

L, the length of the sequence is, L = 9 and x(9) = 1.

The encrypted cipher sequence obtained using Eqs.(20) and (21) is,

$$y = \begin{bmatrix} 4 & 1 & 3 & 4 & 6 & 6 & 5 & 1 & 3 \end{bmatrix} \qquad (32)$$

The decryption using Eqs.(22) and (23) exactly reproduces the original plain text x when there is no error.

Now, let an error occur at location k =2. Let y(2) change to 2 instead of 1. Then the cipher sequence with error would be,

$$\hat{y} = \begin{bmatrix} 4 & \hat{2} & 3 & 4 & 6 & 6 & 5 & 1 & 3 \end{bmatrix} \qquad (33)$$

The error term is shown with the cap in Eq.(33).

Under this condition the decrypted sequence using Eqs.(22) and (23) is found to be,

$$\hat{x} = \begin{bmatrix} 4 & 4 & 6 & 3 & 3 & 3 & 4 & 3 & 5 \end{bmatrix} \qquad (34)$$

Here, the last term of $\hat{x}$ is 5 instead of expected 1. Hence the conclusion is that an error has occurred. Also we can see that the terms of $\hat{x}$ in locations 2 to 9 (k to L) are all different from those of x. This confirms Eq.(30).

VII. GENERATION OF LARGE SIZED T AND S

All the rows and columns of T and S are permutations in the range {1:N}. When permutations are generated say, row by row, it should satisfy the constraint that the column elements should not repeat for any column. When N is large, the above constraint may pose a challenge while generating T. One simple solution is to use the random permutation function to generate the first row of T. Then apply circular left shift (rotation), by one element, to the first row to get the second row and so on. That is,

$$T(i+1, j) = T(i, j+1) \qquad (35)$$

for i = 1,2,…, N-1 and j = 1,2,…, N.

When j reaches N, take j+1=1 which represents circular fold back of the column index.

The circular shift of a permutation is also a permutation. Hence all the rows of T are permutations. Now consider the columns of T. The circular left rotation operation to get successive rows of T, from the first row, is shown in Fig.2. Here the first row having only 4 elements are taken for the purpose of demonstration.

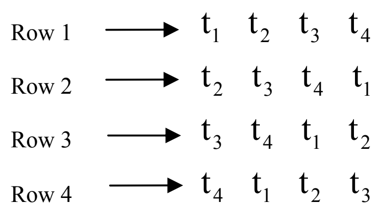
Row 1 → t_1 t_2 t_3 t_4
Row 2 → t_2 t_3 t_4 t_1
Row 3 → t_3 t_4 t_1 t_2
Row 4 → t_4 t_1 t_2 t_3

Fig.2. Circular Left shift of successive rows

From fig.2, it can be seen that all the columns of T are also permutations. Instead of circular left shift, circular right shift also can be used. Then T will be a *circulant matrix* [4]. The following algorithm generates T and S.

```
1.  Generate the first row of T, a random permutation in
    the range {1:N} as,
    $P_1$ = randperm(N).
2.  Generate the remaining rows as,
      for j = 1 to N-1
          $P_{j+1}$ = circular_left_shift($P_j$)
      end
3.  Obtain S from T as described earlier.
```

VII. CONCLUSION

A new technique of encryption and decryption with error detection capability is presented. This technique can be applied to ASCII and Unicode character strings.

REFERENCES

[1] Finite State Transducer, Wikipedia, the free encyclopedia. En.wikipedia.org/wiki/Finite_state_transducer.

[2] A . P . Parkes, A Concise Introduction to Languages and Machines. Springer-Verlag London Ltd. 2008. PP 189-190.

[3] Randompermutation,, MATLAB. ww.mathworks.com/help/techdoc/ref/randperm.html.

[4] Circulant Matrix, Wikipedia, the free encyclopedia. En.wikipedia.org/wiki/Circulant_matrix.