Electronics and Communication Engineering Department
Semester V
(Academic Year 2021-22)

2CSOE78: Scientific Programming

"Scrambler and Descrambler for Digital Signals"

Submitted by : Dixit Dudhat (20BEC033)
Sanchit Sharma (20BEC108)
Siddharth Agarwal (20BEC119)

Submitted to : Prof. Usha Patel

Abstract:

This document is based on the special assignment performed by us for the course of Scientific Programming and is based on the topic of Scrambler and Descrambler for Digital Signals using Python.

Introduction:

In Digital Communication when we are transmitting large no. of data bits serially it might contain some section where there is presence of continuous 0's or 1's. When this data is passed through the channel then due to the presence of noise the section of the bit stream containing continuous 0's and 1's gets changed and at the receiving end we face difficulty in determining the original data. So, to solve this issue we use the concept of Scrambling and Descrambling.

Theory:

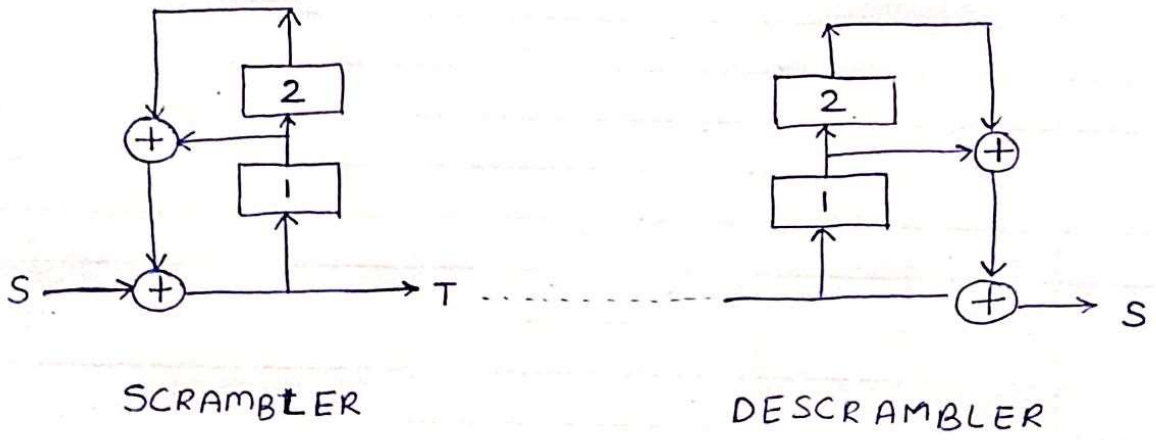
A Scrambler (also referred to as a randomizer) is a device that manipulates a data stream before transmitting. It replaces sequences (referred to as whitening sequences) with other sequences without removing undesirable sequences. The manipulations done by the Scrambler are reversed by a Descrambler at the receiving side. The hardware of the Scrambler contains no. of Shift Registers. The hardware inserts delays to the input based on the given polynomial.

Let us understand this concept with help of an example,

Data input (S) = 1 1 1 0 0 0 0 0

Given Polynomial :- $T = S \oplus (D \oplus D^2) T$ {Where T is output to the scrambler}

Hardware circuit for the given polynomial:



Suppose, $D \oplus D^2 = F$

$\therefore$ The polynomial becomes, $T = S \oplus FT$

Now,

$T = S \oplus F(S \oplus FT)$ {Putting $T = S \oplus FT$}

$= S \oplus FS \oplus F^2T$

$= S \oplus FS \oplus F^2(S \oplus FT)$

$= S \oplus FS \oplus F^2S \oplus F^3T$

$= S \oplus FS \oplus F^2S \oplus F^3S \oplus F^4S \oplus \ldots\ldots \oplus F^8S$

As $F = D \oplus D^2$. Hence finding the powers of D.

$F = 1, (2)$

$F^2 = (2), (4)$

$F^3 = 3, (4), (5), (6)$

$F^4 = 4, (8)$

$F^5 = (5), 9, 10, (6)$

$F^6 = 6, 7, 7, (8)$

$F^7 = 7, (8), 9, 10$

$F^8 = (8), 9, 9, 10$

{Eliminating the repetetive powers and powers more than 8}

$\therefore T = S \oplus DS + D^3S \oplus D^4S \oplus D^6S \oplus D^7S$

S	=	1	1	1	0	0	0	0	0
DS	=		1	1	1	0	0	0	0
D^3S	=				1	1	1	0	0
D^4S	=					1	1	1	0
D^6S	=							1	1
D^7S	=								1
T	=	1	0	0	0	0	0	0	0

We have successfully scrambled the input data. Now, the data bits are ready to get pass through the noisy channel.

At the receiver end we need to descramble the data bits in it's original form. So, for that we have done the following procedure.

Descrambler (Receivier Side) :-

$$S = T \oplus (D + D^2)T$$

$$\therefore S = T \oplus DT \oplus D^2T$$

T =	1	0	0	0	0	0	0	0
DT =		1	0	0	0	0	0	0
D^2T =			1	0	0	0	0	0
S =	1	1	1	0	0	0	0	0

(Descrambler Output).

We have successfully received the transmitted data bits with the help of Scrambler and Descrambler.

Concepts used in project:

- Basic python loops, conditional statements, recursion and user defined functions.
- Numpy
- File handling
- Image Processing
- Matplotlib
- Dynamic Programming

Process we have followed in making of this project:

1. **C++ code for scrambler.**
   We had to first implement the code in C++ because we were having trouble implementing the algorithm directly in Python.

2. **Converted it into recursive python code**
   We had the code converted to Python after the C++ algorithms had been successfully implemented and tested. We had it first implemented in a recursive manner for our convenience.

3. **Appended descrambler algorithm into it**
   Now that the scrambler in Python has been successfully implemented and tested, we have also included the descrambler method in the code.

4. **Added file handling. Now the program can take input directly from the file.**
   Up until this point, the programme only accepted input from the terminal, so after incorporating data scrambling and descrambling algorithms, we added file handling. In our project, file handling has added a feature that allows the user to input data directly into a file and receive output from it.

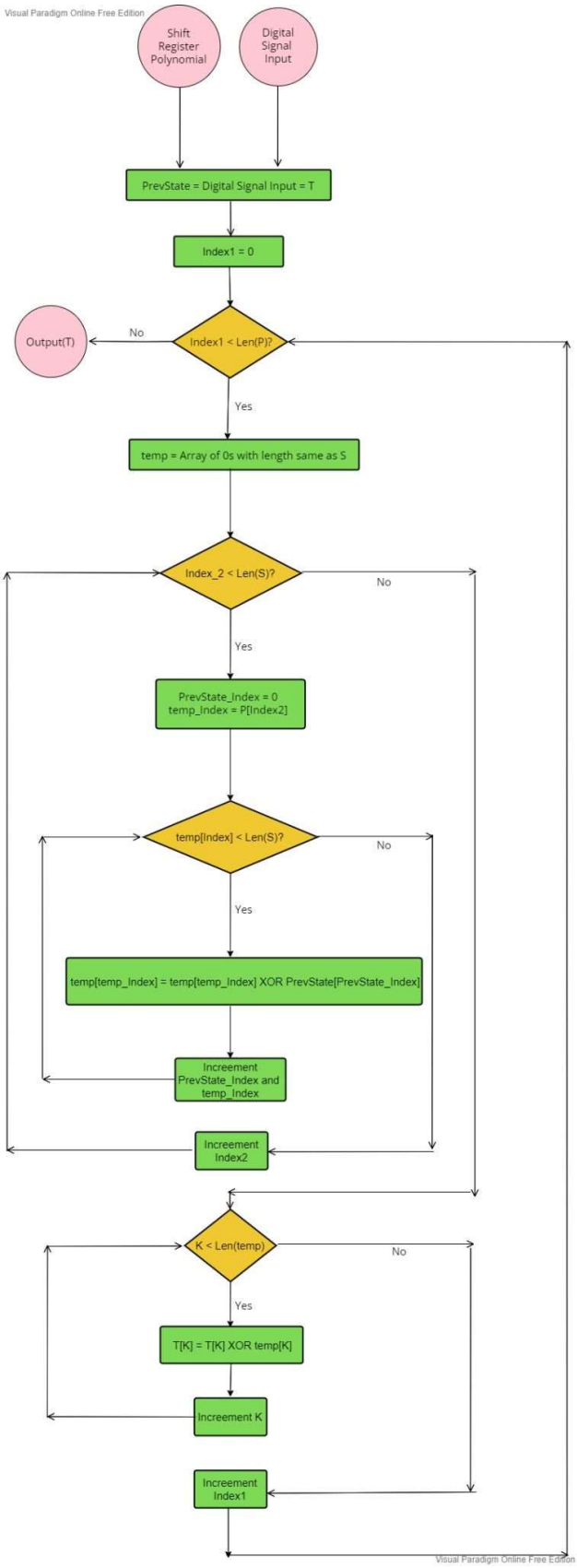
5. **For handling large numbers of operations converted to iterative code.**
   We had changed the recursive code to iterative code in order to make it more time complexity efficient. This also benefited our program in terms of the call stack that the program was earlier using in the recursive algorithm.

6. **Added image handling**
   Till now users can scramble and descramble only digital signals using our project . So now we have developed an algorithm for image handling.

7. **Image handling integration with main code failed due to large size of image**
   Because our algorithm required millions of operations to scramble and descramble the image, the kernel was unable to handle the programme and it crashed during testing, limiting the image handling algorithm from being integrated with the main code.

8. **Added image compression algorithm using dynamic programming**
   We had incorporated a dynamic programming approach into our code to reduce the amount of operations required.

Flowchart:


Visual Paradigm Online Free Edition
Shift Register Polynomial
Digital Signal Input
PrevState = Digital Signal Input = T
Index1 = 0
Output(T)
No
Index1 < Len(P)?
Yes
temp = Array of 0s with length same as S
Index_2 < Len(S)?
No
Yes
PrevState_Index = 0
temp_Index = P[Index2]
temp[Index] < Len(S)?
No
Yes
temp[temp_Index] = temp[temp_Index] XOR PrevState[PrevState_Index]
Increement PrevState_Index and temp_Index
Increement Index2
K < Len(temp)
No
Yes
T[K] = T[K] XOR temp[K]
Increement K
Increement Index1
Visual Paradigm Online Free Edition

Output:

```
output of the scrambler is :  [1 0 0 0 0 0 0 0]
output of the descrambler is :  [1 1 1 0 0 0 0 0]
True
output of the scrambler is :  [1 0 0 0 1 1 0 0 1 0 1]
output of the descrambler is :  [1 0 1 1 1 1 1 0 0 0 0]
True
output of the scrambler is :  [1 0 1 0 0 1 0 0 0 0 0]
output of the descrambler is :  [1 0 1 1 1 1 1 1 1 1 1]
True
```

Results of our project:

- Data Scrambler and Descrambler
- Data can be fed through terminal or through file
- Image scrambler

Issues faced while developing this project:

1. Image is of huge size so to scramble and Descramble the terminal have to do millions of operations.
   To tackle this we had compressed the image before scrambling and decompressed it afterwards.
2. We can't scramble it using random because if we scramble it using random we will not be able to descramble it.
   To tackle this we came with our algorithm that we had implemented using dynamic programming for speed efficiency.
3. When we do the XORing operation in the scrambler algorithm on the image it will modify the original content of the image.

Conclusion:

In this project we have successfully implemented the algorithm for Scrambler and Descrambler for Digital Signals using Python. We have learnt various concepts of python like image processing, file handling, array operations using numpy. Also we have acquired a knowledge of dynamic programming. Using this project a user can Scramble and Descramble binary data or image through manually or using file input and output.