**Development log**

**Frank M**

**Encryption Stage:**

In this stage I will develop the encryption algorithm, which I have based on the Advance Encryption Standard as well as another algorithm on top of that to add entropy to the encryption. The goal for this is to develop an encryption algorithm that can consistently output an at least 50% efficiency encrypted output.

**Data dictionary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Size | Use |
| Message | String | 16 characters | To take the input from the user. |
| SBox | Integer Array | 16 x 10 integers | To store where each index is sent during the first round of encryption. |
| CRbox | Integer Array | 4 x 4 integers | To store where each index is sent during the second round of encryption. |
| NewArray | Integer Array | 16 x 16 integers | To store the output of the different rounds of encryption. |
| RandomNum | Integer | 1 integer | A random number to improve entropy. |
| TempArray1 | Integer Array | 4 integers | A temporary integer used to swap the rows and columns of the array. |
| TempArray2 | Integer Array | 4 integers | A temporary integer used to swap the rows and columns of the array. |
| TempArray3 | Integer Array | 4 integers | A temporary integer used to swap the rows and columns of the array. |
| TempArray4 | Integer Array | 4 integers | A temporary integer used to swap the rows and columns of the array. |
| TempArray5 | Integer Array | 4 integers | A temporary integer used to swap the rows and columns of the array. |
| cShiftedArray | Integer Array | 16 x 16 integers | The output of shifting the columns of the array. |
| rShiftedArray | Integer Array | 16 x 16 integers | The output of shifting the rows of the array. |
| StringOutput | String | 16 characters | The string that is outputted from the program. |
| Bytes | Bytes Array | 16 bytes | Used as an intermediatory stage that convers the message into an array of integers |
| newLength | Integer | 1 integer | Determines how many arrays the message will be split up into. (Used more in the next stage) |

**Diagram:**

A diagram of a computer flowchart

AI-generated content may be incorrect.

**Code and class diagrams:**

This class takes the input given (an array of length 16) and using the sBox (a list of predetermined ways to jumble up a length 16 array) jumbles up the array to add entropy. I have done it this way because both the sender and receiver will have the sBox therefore the receiver can just look at the key given and apply the specific shift in its sBox.

|  |
| --- |
| **Name:**  sBox\_jumble |
| **Methods:**  sBox\_jumble() |
| **Attributes:**  Array - sBox[16][10]  Array - newArray[16]  Int - randomNum |

import java.util.Random;  
public class sBox\_jumble {  
 //My beautiful sBox  
 static int[][] *sBox* = {  
 {10,9,8,3,1,15,12,11,5,2,7,13,0,14,4,6},  
 {3,8,1,2,13,6,7,5,4,14,0,12,11,9,10,15},  
 {5,4,15,13,2,9,7,6,10,11,14,12,3,8,1,0},  
 {5,10,2,13,9,7,3,14,12,8,1,6,11,0,4,15},  
 {5,11,14,1,7,3,6,12,9,15,13,2,10,8,0,4},  
 {12,9,5,3,15,6,2,14,13,10,11,7,0,4,1,8},  
 {14,2,12,3,9,8,0,4,1,15,11,5,10,6,7,13},  
 {14,15,7,5,8,12,0,13,2,10,1,11,9,4,3,6},  
 {10,2,12,11,3,14,4,1,5,9,0,8,7,6,13,15},  
 {6,5,8,0,15,10,13,1,7,11,12,9,4,2,14,3}  
 };  
 public static int[] sBox\_\_jumble(int[] Array){  
 int[] newArray = new int[16];  
 Random rand = new Random();  
 int randomNum = rand.nextInt(10);  
 for(int i=0;i<16;i++){  
 newArray[*sBox*[randomNum][i]] = Array[i];  
 }  
  
 return newArray;  
 }  
}

This class takes the input a length 16 array and outputs the same array with its columns or rows shifted. It does this by using the CRBox which is an array of 5 ways to shift the rows/ columns. I have done it this way because both the receiver and sender with have the same CRBox which means that the receiver will be able to look at the key and determine which one of the shifts were used.

|  |
| --- |
| **Name:**  Column\_Row\_Shift |
| **Methods:**  ColumnShift()  RowShift() |
| **Attributes:**  Array - CRBox  Array - rShiftedArray  Array - cShiftedArray  Array - tempArray1  Array - tempArray2  Array - tempArray3  Array - tempArray4  Array - tempArray5 |

import java.util.Random;  
public class Column\_Row\_Shift {  
  
 static int[][] *CRBox* = {  
 {4,12,8,0},  
 {0,8,4,12},  
 {4,0,12,8},  
 {8,12,4,0}  
 };  
 public static int[] ColumnShift(int[] Array) {  
 int[] tempArray1 = new int[4];  
 int[] tempArray2 = new int[4];  
 int[] tempArray3 = new int[4];  
 int[] tempArray4 = new int[4];  
 int[] cShiftedArray = new int[16];  
 for(int i = 0; i < 4; i++){  
 tempArray1[i] = Array[i];  
 tempArray2[i] = Array[i + 4];  
 tempArray3[i] = Array[i + 8];  
 tempArray4[i] = Array[i + 12];  
 }  
 Random rand = new Random();  
 int randomNum = rand.nextInt(4);  
 for(int i = 0; i < 4; i++){  
 cShiftedArray[i + *CRBox*[0][randomNum]] = tempArray1[i];  
 cShiftedArray[i + *CRBox*[1][randomNum]] = tempArray2[i];  
 cShiftedArray[i + *CRBox*[2][randomNum]] = tempArray3[i];  
 cShiftedArray[i + *CRBox*[3][randomNum]] = tempArray4[i];  
 }  
 return cShiftedArray;  
 }  
 public static int[] RowShift(int[] Array) {  
 int[] key = new int[24];  
 for(int i = (Array.length - 24); i > Array.length - 24; i++){  
 key[i] = Array[i];  
 }  
 int[] tempArray1 = new int[4];  
 int[] tempArray2 = new int[4];  
 int[] tempArray3 = new int[4];  
 int[] tempArray4 = new int[4];  
 int[] rShiftedArray = new int[16];  
 for(int i = 0; i < 4; i++){  
 tempArray1[i] = Array[i];  
 tempArray2[i] = Array[i + 4];  
 tempArray3[i] = Array[i + 8];  
 tempArray4[i] = Array[i + 12];  
 }  
 Random rand = new Random();  
 int randomNum = rand.nextInt(4);  
 for(int i = 0; i < 4; i++){  
 rShiftedArray[i + *CRBox*[randomNum][0]] = tempArray1[i];  
 rShiftedArray[i + *CRBox*[randomNum][1]] = tempArray2[i];  
 rShiftedArray[i + *CRBox*[randomNum][2]] = tempArray3[i];  
 rShiftedArray[i + *CRBox*[randomNum][3]] = tempArray4[i];  
 }  
 return rShiftedArray;  
 }  
}

This class is used to convert the message into an array of integers, split up the message into 16 length arrays and encrypt it. It does this by accessing the methods in other classes like Column\_Row\_Shift. I have set it up in this way so that when I go to develop the decryption algorithm I can just use different classes and methods in the main class instead of having all the encryption methods and classed in the main class.

|  |
| --- |
| **Name:**  Encryption |
| **Methods:**  encryption()  encryption() |
| **Attributes:**  Array – Bytes[16]  Int – newLength  Array – newArray  Int - randomNum |

//Libraries  
import java.nio.charset.StandardCharsets;  
import java.util.Random;  
  
//Encryption class  
public class encryption {  
 String message;  
 public encryption(String message) {  
 this.message = message;  
 }  
 public int[] encryption(String message){  
 byte[] bytes = message.getBytes(StandardCharsets.*UTF\_8*);  
 int newLength = (16 -(bytes.length % 16)) % 16;  
 int[] Array = new int[(newLength + bytes.length)];  
 for(int i = 0 ; i < bytes.length ; i++){  
 Array[i] = bytes[i];  
 }  
 for(int i = 0; i < 12; i++){  
 Array = sBox\_jumble.*sBox\_\_jumble*(Array);  
 Array = Column\_Row\_Shift.*RowShift*(Array);  
 Array = Column\_Row\_Shift.*RowShift*(Array);  
 }  
 Random rand = new Random();  
 for(int i = 0; i < 16; i++){  
 int randomNum = rand.nextInt(10);  
 if(Array[i] == 0){  
 Array[i] = 101 + randomNum;  
 }  
 Array[i] = 100 + (Array[i] \* Array[i]);  
 if(randomNum < 2){  
 Array[i] = Array[i] + (Array[i] \* randomNum);  
 }  
 if(randomNum > 2 && randomNum < 5){  
 Array[i] = Array[i] - (Array[2] \* randomNum);  
 }  
 if(randomNum > 5 && randomNum < 8 && i > 1){  
 Array[i] = Array[i-1] / randomNum;  
 Array[i] = Array[i] / (randomNum % (Array[i-1] + 1));  
 }  
 }  
 return Array;  
 }  
}

**References:**

(Gupta, 2025)

(Kartik, 2025)

(randomsapien, 2025)

(Kartik, Returning Multiple values in Java, 2025)

(Code\_r, 2025)