**Security and Privacy COMP3031J**

**Data Encryption Standard James Crown**

**Student Number:**

**BJUT: H20180909002  
UCD: 15534833**

**Overview:**

* Implemented in Java
* UI done using Processing library extension
* Accepts File input or Text input
* Encrypts Text to Binary
* Decrypts Binary to Text

**Classes**

* Main

The Main class consists of two key parts.  
**1)Algorithm**

* This algorithm function is called when the UI triggers it through encrypt or decrypt button being pressed. It creates a key and message object using the current textbox values. It then calls message.calculate using the current type which will be 0 if encrypt is pressed and 1 if decrypt is pressed. It then has two states depending on the type integer. In general, both states get a result from the message object either in binary or text format, then they create a file called encryptionResults or decryptionResults and stores the results in there.

**2)UI**

* The UI is entirely created through the extended library. We first set the screen size to full screen in the setting method and load the background image in setup(). We resize the image to the screen and create a control applet. We continue to add features to the control applet such as text fields and bangs (buttons) . These features have scaled positioning , sizes etc.  
  The draw function then paints all info to the screen starting with the background. Simple text is used for Key: and message:. I then calculated the parameters of all the buttons as the inbuilt library functions were not working. These parameters are later used in the click event function to check if the mouse is over a button or not when clicked. When the mouse is clicked we have multiple possible outputs, sometimes the file reader is brought up and other times we store the text field values, set outputBool to true and call the main algorithm. The main algorithm instantly calculates all its values and the newly true outputBool triggers draw() to draw out the results of the algorithm. I paid special attention to ensuring that large binary outputs stay within the center through adding newlines to a dummy string and using a scrollable text box.  
  The fileReader was something I got help from a website to initialise but was overall very simple to implement. It opens the file explorer and I have it limited to txt dat or jpg. We can choose a key file or a message File. The Key must be 8 characters long for anything to calculate.
* Message

Takes a String input and key Object as arguments. Its prepare method will change any text string to binary if needed. It also extends the string length so that is in multiples of 64 characters. Then it splits up this string into blocks of 64 bits and adds these blocks to a block array list. In the calculate method which is always called, this block array list is iterated and each block gets the calculate method within it called. Then it calls the getTextAllBlocks() function which iterates through the blockchain once more and gets their outputs. Its changeToBinary function converts the input string into bytes and then into binary, we make sure all values are of the correct length also. And then add them to a final binary string.

* Block

Block takes input string and key object in constructor. Its calculate function is a collection of other methods. It runs the blocks binary string through its initial permutation, then runs CipherFunction with the new String, Key and type integer as arguments. The results of the function is then permuted through the inverse permutation. It also has a method to convert binary to text when decrypting.

* CipherFunction

Cipher Function takes a String input , key Object and type integer. We first split the input string into left and right. Next we loop 16 times and for each iteration we set the Ln+1 = Rn and then we have two options that depend on the type argument provided in the constructor. The two options only differ on which keys they use as one is used for encryption and the other for decryption. The formula is otherwise the same , Rn+1 = XOR( Ln, f(Rn, subkey(n)))

After this Loop we add the final L16 and R16 together but R16 at the front and we are left with the final output.

XOR – loop through input strings, if two characters are equal then add 0 to output string, otherwise add 1.

Function f – Arguments(oldR,kn). Firstly we permute oldR through E to create the newR. Then we XOR the newR and subkey to create newRK1. This string is 48 bits long so we iterate through it in groups of 6 bits. First and last bits add together as a rowNumber and the middle 4 add together as the columnNumber. Then for the first 6 bits we go to S1 and find the corresponding values for that row and column and add it to an arraylist. We do this for each set of 6 bits corresponding to S1,S2…S8.   
We then convert the selection of decimal values we have into a binary string and permute that string through prime P and return the output.

* Key

This class takes a String input into its Constructor, this is the key.  
The key is converted to Binary. The binary key is then permuted through PC1 and split into C0 and D0  
I then compute the list of CnDn’s through left shifting Cn-1 and Dn-1 once or twice depending on their corresponding left shift amounts and then reconnected. Each CnDn is run through PC2 to create the subkeys.

**Files**

* EncryptionResults.txt
* DecryptionResults.txt
* Chosen File for Key or Message input also.
* cryptoBack.jpeg (background image)

**Libraries**

* jdk 9.0.4
* core UI
* controlP5 Used for Text Boxes, Buttons etc