SWE 314: Software Security Engineering
2st Semester 1441-1442H

College of Computer & Info Sciences
Department of Software Engineering

Hidden Message.

Case study: message protection
Mobile application to protect/secure messages.

Assignment #2

#	Student Name	ID
1	Hussa AlQuraishi	437200828
2	Fouz AlRabai	437200482

Table of content

Introduction ***3***

Source Code ***3***

Encryption Source Code: 3

Decryption Source Code: 5

Execution results ***7***

Happy scenario: 7

Sad scenario: 8

References ***8***

Introduction

In this assignment, it was assigned to us to implement a cryptosystem. We implement cipher application by using Keyword columnar algorithm that operates in Java programing language and Android Studio IDE.

The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows and then reading the ciphertext off in columns one by one [1]. First of all, we need to create a matrix of size n*m, (n= rows number, m=columns numbers) where m equals to numbers of keyword's characters and n equals to plaintext divided by columns number. In the first row write the key down, after that in the second row write the order of character in alphabetical order of the characters in the keyword(write 1 to character A and 2 to character M that means A precedes M in alphabetical order), then write your plaintext. Finally, read off in columns, in the order specified by the keyword. For example, the plaintext: AttackAtDawn, keyword: spyman.

S	P	Y	M	A	N
5	4	6	2	1	3
A	t	t	a	c	k
A	t	D	a	w	n

The ciphertext will be cwaaknttAAtD

Hidden Message application has a great feature which is, some ciphertexts were encrypted by Hidden Message can only be decrypted by Hidden Message. This feature happens because Hidden Message adds randomly some spaces to the ciphertext.

Source Code

Encryption Source Code:

```
int stars=0;
colNum=key.length();
mod =userText.length()%colNum;
rowNum=(userText.length()/colNum)+2;
if(mod != 0 ){
    rowNum++;
    stars=key.length()-mod;
    if(stars != 0){
        for(int i =0;i<stars;i++){
            userText=userText+" ";
        }
    }
}


System.out.println(rowNum);
char[][]plain=new char[rowNum+2][colNum];
//to put key at first row
for(int i =0;i<key.length();i++) {
    plain[0][i]=key.charAt(i);
}
//to sort key
char[] ke = new char[key.length()];
char[] keSort=new char[key.length()];
for(int i =0;i<key.length();i++) {
    ke[i]=plain[0][i];
    keSort[i]=plain[0][i];
}

Arrays.sort(keSort);
//insert index
for(int i =0;i<colNum;i++) {
    for (int j=0;j<colNum;j++ ) {
        if( keSort[j]== ke[i] ){
            String n = j+1+"";
            plain[1][i]=n.charAt(0);
        }
    }

}
int index =0;
for(int i =2;i<rowNum;i++) {
    for (int j=0;j<key.length();j++ ) {
        if(index < userText.length()) {

            plain[i][j]=userText.charAt(index);

            index++;

        }


    }
}
//to encrypt
String ciphir="";
index =0;
int h=49;
for(int i =2;i<rowNum;i++) {
    System.out.println("row "+i);
    for (int j=0;j<key.length();j++ ) {
        if(index < userText.length()) {
            for(int k=0;k<key.length();k++) {
                int x =plain[1][k];
                if( x == h) {
                    System.out.println("inside if ");
                    for(int w=2;w<rowNum;w++)
                        ciphir=ciphir+plain[w][k];

                                        index++;
                    h++;
                }
            }
        }
    }
}
```

Decryption Source Code:

```java
//first is row then col
        colNum=key.length();
        mod =userText.length()%colNum;

        rowNum=(userText.length()/colNum)+2;
if(mod != 0)
    rowNum++;
        char[][]plain=new char[rowNum+2][colNum];
        //to put key at first row
        for(int i =0;i<key.length();i++) {
            plain[0][i]=key.charAt(i);
        }
        //to sort key
        char[] ke = new char[key.length()];
        char[] keSort=new char[key.length()];
        for(int i =0;i<key.length();i++) {
            ke[i]=plain[0][i];
            keSort[i]=plain[0][i];
        }

        Arrays.sort(keSort);
        //insert index
        for(int i =0;i<colNum;i++) {
            for (int j=0;j<colNum;j++ ) {
                if( keSort[j]== ke[i] ){
                    String n = j+1+"";
                    plain[1][i]=n.charAt(0);

                }
            }

        }
        int index =0;
        for(int i =2;i<rowNum;i++) {
            for (int j=0;j<key.length();j++ ) {
                if(index < userText.length()) {

                    plain[i][j]=userText.charAt(index);


                    index++;

                }

            }
        }
        //to dycrypt
        String plainT="";


        char[][]encrypted =new char[rowNum][colNum];
        for(int i =0;i<key.length();i++) {
            encrypted[0][i]=key.charAt(i);
        }

        for(int i =0;i<key.length();i++) {
            for (int j=0;j<key.length();j++ ) {
                if( keSort[j]== ke[i] ){
                    String n = j+1+"";
                    encrypted[1][i]=n.charAt(0);
                }
            }

        }

        // now the real decrypt
```

```
index =0;
int f=49;
int loopsize;
int mod1=userText.length()%colNum;

    loopsize =rowNum;

        int q =2;
       for (int j=0;j<key.length();j++ ) {

    if(index < userText.length()) {
        for(int k=0;k<key.length();k++) {
            int x =encrypted[1][k];

            if( x == f) {
                 for(q =2;q<loopsize;q++) {

                    if(index<userText.length()  ){
                        encrypted[q][k]=userText.charAt(index);
                    index++;


                    }

                    //this to make sure

                }

                                       f++;
            }
        }
    }
}

for (int i = 2; i < encrypted.length; i++) {
    for (int j = 0; (encrypted[i] != null && j < encrypted[i].length); j++)
        plainT =plainT+encrypted[i][j] +"";


}
```

Execution results

Happy scenario:

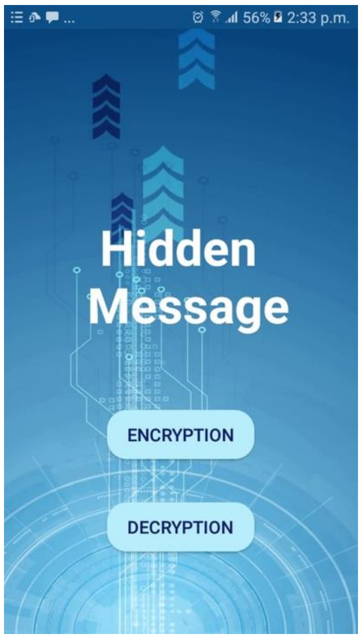


Figure 1 Home page

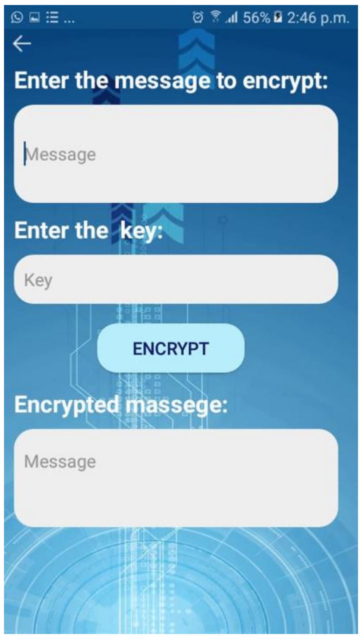
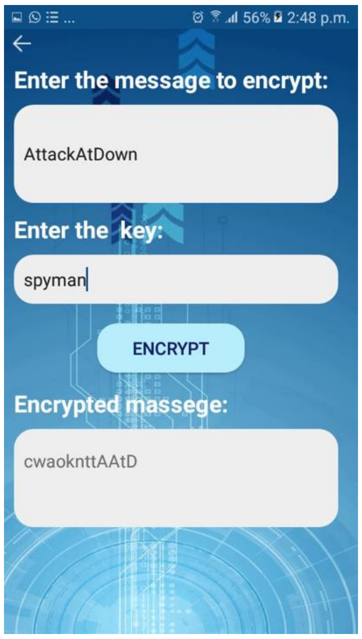


Figure 2 Encrypt page



Enter the message to encrypt:

AttackAtDown

Enter the key:

spyman

Encrypted massege:

cwaoknttAAtD

Figure 3 filled fields and press the ENCRYPT button

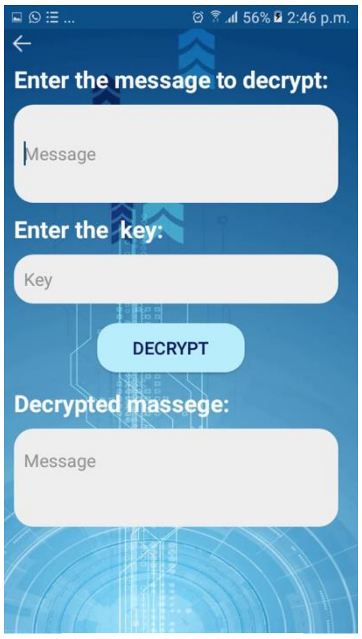
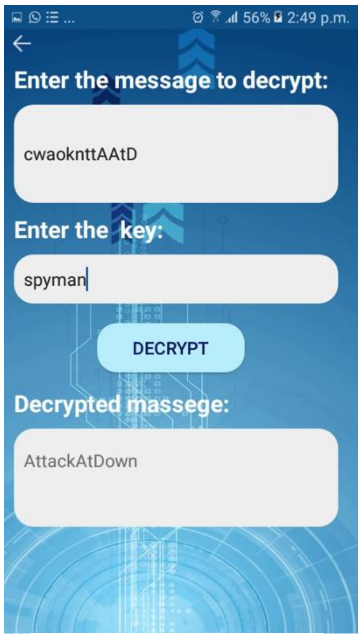


Figure 4 Decrypt page



Enter the message to decrypt:

cwaoknttAAtD

Enter the key:

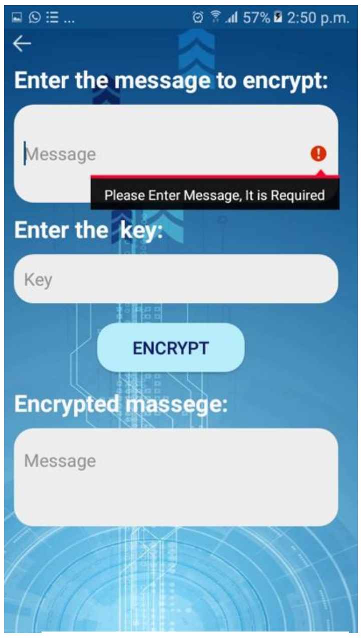
spyman

Decrypted massege:

AttackAtDown

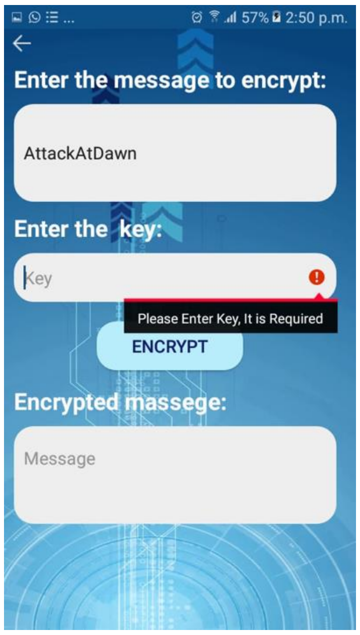
Figure 5 filled fields and press the DECRYPT button

Sad scenario:



Please Enter Message, It is Required

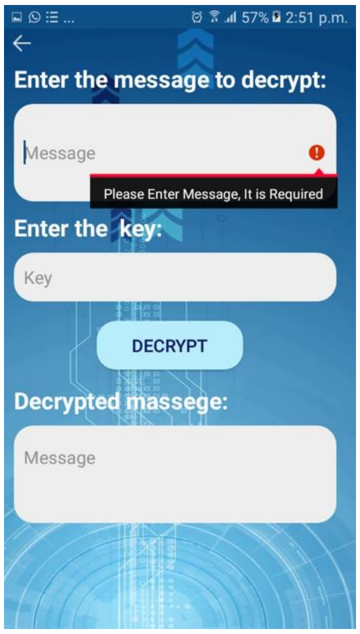
Figure 6 press the ENCRYPT button with empty fields



AttackAtDawn

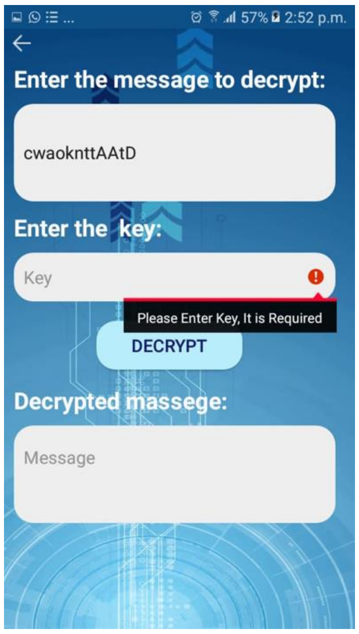
Please Enter Key, It is Required

Figure 7 press the ENCRYPT button with missing Key field



Please Enter Message, It is Required

Figure 8 press the DECRYPT button with empty fields



cwaoknttAAtD

Please Enter Key, It is Required

Figure 9 press the DECRYPT button with missing Key field

References

[1] GeeksforGeeks, "Columnar Transposition Cipher," GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/columnar-transposition-cipher/. [Accessed 1 March 2020].