

SWE314 PROJECT

Fall 2019



Section: 56739

Name	ID
Majd Bandar Bin Musibeh	438200829
Ghada Abdurrahman AlGhurairi	438201865
Shahad Omar AlHarbi	438201706
Lujain Abdullah Almuhaitheef	438201099
Rahaf Mohammed Al-Omar	438202048

Introduction:

Information nowadays become more valuable than ever, and as everything that is valuable it can be targeted. Information – sensitive information to be more specific- need to be secured in order to transmit it without it being exposed to unauthorized parties.

In our system (Whisper), we are encrypting texts to a ciphertext using our own modern block cipher and decrypt it to the original plaintext.

Whisper consist of two rounds, each have a s-box, p-box and rail fence techniques, after these two rounds it will use the result text from round 2 and convert it again then it will swap the converted text.

How can Whisper Encrypt a plaintext?

The plaintext should be either letters a-z (capital, small or both) , digits 0-9 and spaces . If the user entered some characters like (!@#\$%^*(){} etc.) the system will not work (instead it will ask the user to enter a valid input consists of letters and digits). After that, the plaintext will go through the first round of encryption.

- 1- First, every letter and number in the plaintext should be converted to another letter or digit based on a special substitution boxes.

For letters:

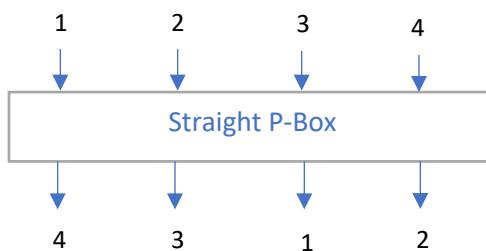
A → N	B → W	C → E	D → C
E → D	F → V	G → B	H → O
I → A	J → X	K → M	L → T
M → S	N → Z	O → J	P → K
Q → U	R → R	S → F	T → L
U → I	V → H	W → Q	X → P
Y → G	Z → Y		

For digits:

0 → 4	1 → 3	2 → 0	3 → 6	4 → 9
5 → 5	6 → 8	7 → 1	8 → 7	9 → 2

In case the user entered a space, the system will convert it to (~) and works with it as if its work with any letter.

- 2- Second, we will use a special straight p-box. We will divide the result of the substitution to four parts, if the input was just a one letter, we will add ~|} to make it four, if it two letters we will add ~| and if it three we will use ~. After the division, we will use the following p-box.



3- Finally in the round, we will do a simple rail fence and the width is 2.

Then, the second round is started by the same steps.

After the 2 rounds we have a more two steps.

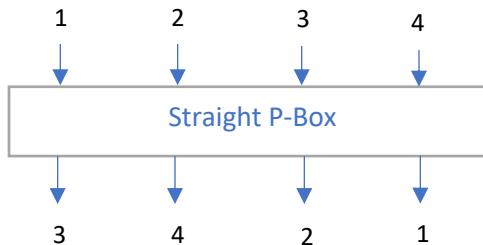
- 1- Convert the result of the second round to ASCII code (every character will take 3 digits after the converting).
- 2- The resulting ASCII codes, we will do a simple swap between each 2 digits.

Now we got the ciphertext!

Now, how can Whisper Decrypt a ciphertext?

First, we will check if the ciphertext is digits only, the number of digits is multiple of 12 (since we use 3 digits ASCII codes).

- 1- we will do a simple swap between each 2 digits.
- 2- Take every three digits and convert it to characters based on the ASCII code.
- 3- The round id started:
 - a- Simple rail fence with width 2.
 - b- The result is used in the next straight p-box after dividing the ciphertext to 4 parts as we mentioned.



- c- The last par in the round will be a substitution based on:

For letters:

A → I	B → G	C → D	D → E
E → C	F → S	G → Y	H → V
I → U	J → O	K → P	L → T
M → K	N → A	O → H	P → X
Q → W	R → R	S → M	T → L
U → Q	V → F	W → B	X → J
Y → Z	Z → N		

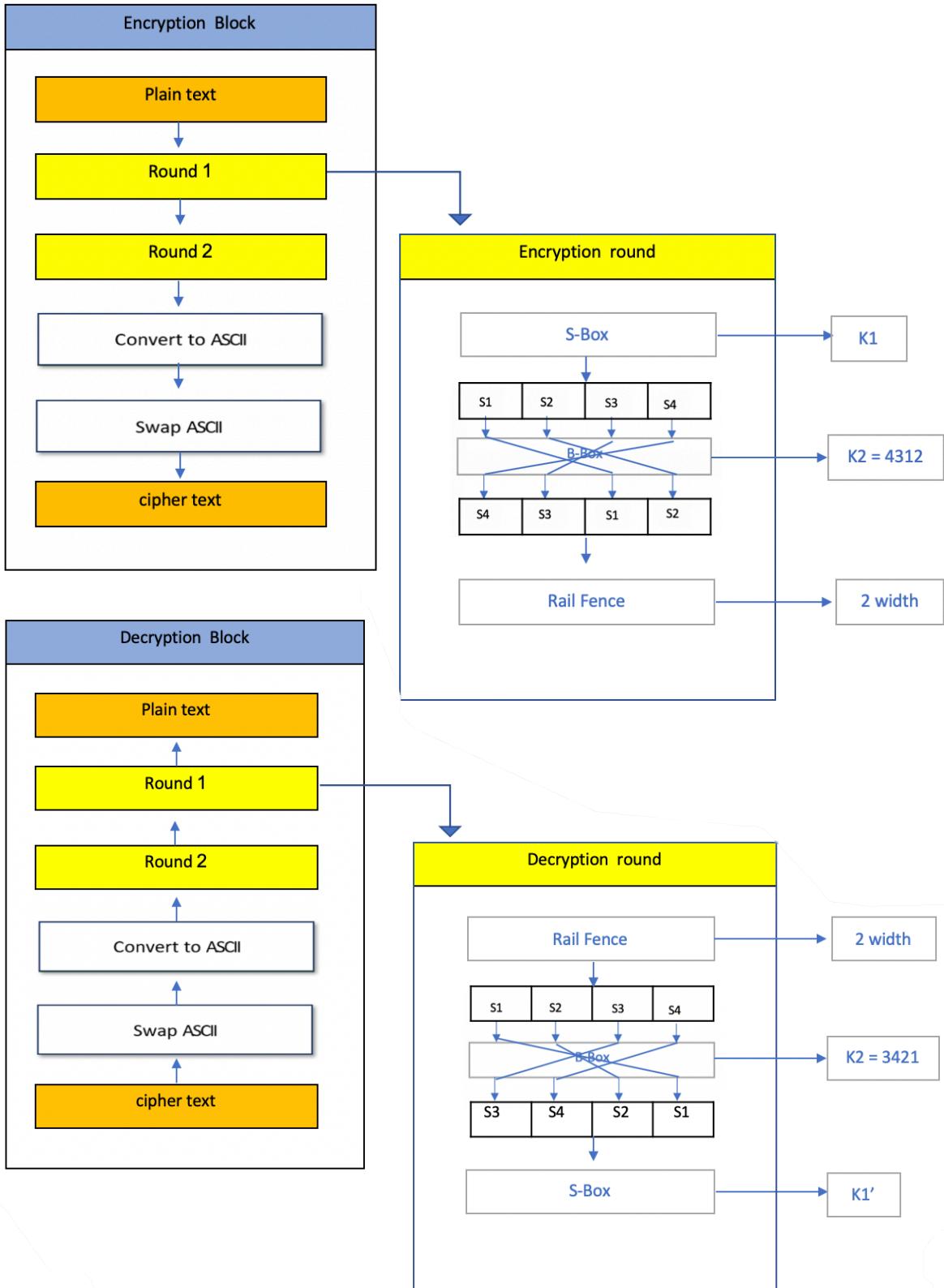
For digits:

0 → 2	1 → 7	2 → 9	3 → 1	4 → 0
5 → 5	6 → 3	7 → 8	8 → 6	9 → 4

If the character is (~) the system will convert it to a space.

The we will do another round by steps a-c, after finish this round we will get the plaintext!

Whisper Model:



screenshots of the source code:

Class encryption:

- The substitution method (convert):

```

28     public String convert(String a) {
29         int length = a.length();
30         String total = "";
31         char c;
32         for (int i = 0; i < length; i++) {
33             c = a.charAt(i);
34             switch (c) {
35                 case 'A':
36                     total += 'N';
37                     break;
38                 case 'B':
39                     total += 'W';
40                     break;
41                 case 'C':
42                     total += 'E';
43                     break;
44                 case 'D':
45                     total += 'C';
46                     break;
47                 case 'E':
48                     total += 'D';
49                     break;
50                 case 'F':
51                     total += 'V';
52                     break;
53                 case 'G':
54                     total += 'B';
55                     break;
56                 case 'H':
57                     total += 'O';
58                     break;
59                 case 'I':
60                     total += 'A';
61                     break;
62                 case 'J':
63                     total += 'J';
64                     break;
65             }
66         }
67         return total;
68     }

```

```

62         case 'J':
63             total += 'X';
64             break;
65         case 'K':
66             total += 'M';
67             break;
68         case 'L':
69             total += 'T';
70             break;
71         case 'M':
72             total += 'S';
73             break;
74         case 'N':
75             total += 'Z';
76             break;
77         case 'O':
78             total += 'J';
79             break;
80         case 'P':
81             total += 'K';
82             break;
83         case 'Q':
84             total += 'U';
85             break;
86         case 'R':
87             total += 'R';
88             break;
89         case 'S':
90             total += 'F';
91             break;
92         case 'T':
93             total += 'L';
94             break;
95         case 'U':
96             total += 'I';
97             break;
98         case 'V':
99             total += 'H';
100            break;
101        case 'W':
102            total += 'O';
103            break;
104        case 'X':
105            total += 'P';
106            break;
107        case 'Y':
108            total += 'G';
109            break;
110        case 'Z':
111            total += 'Y';
112            break;
113        case '0':
114            total += '4';
115            break;
116        case '1':
117            total += '3';
118            break;
119        case '2':
120            total += '0';
121            break;
122        case '3':
123            total += '6';
124            break;
125        case '4':
126            total += '9';
127            break;
128        case '5':
129            total += '5';
130            break;
131        case '6':
132            total += '8';
133            break;
134        case '7':
135            total += '1';
136            break;
137        case '8':
138            total += '7';
139            break;
140        case '9':
141            total += '2';
142            break;
143        case ' ':
144            total += '~';
145            break;
146        case 'a':
147            total += 'n';
148            break;
149        case 'b':
150            total += 'w';
151            break;
152        case 'c':
153            total += 'e';
154            break;
155        case 'd':
156            total += 'c';
157            break;
158        case 'e':
159            total += 'd';
160            break;
161        case 'f':
162            total += 'v';
163            break;
164        case 'g':
165            total += 'b';
166            break;
167        case 'h':
168            total += 'o';
169            break;
170        case 'i':
171            total += 'r';
172            break;
173        case 'j':
174            total += 'l';
175            break;
176        case 'k':
177            total += 't';
178            break;
179        case 'l':
180            total += 's';
181            break;
182        case 'm':
183            total += 'z';
184            break;
185        case 'n':
186            total += 'x';
187            break;
188        case 'o':
189            total += 'y';
190            break;
191        case 'p':
192            total += 'u';
193            break;
194        case 'q':
195            total += 'f';
196            break;
197        case 'r':
198            total += 'g';
199            break;
200        case 's':
201            total += 'h';
202            break;
203        case 't':
204            total += 'i';
205            break;
206        case 'v':
207            total += 'p';
208            break;
209        case 'w':
210            total += 'q';
211            break;
212        case 'x':
213            total += 'n';
214            break;
215        case 'y':
216            total += 'm';
217            break;
218        case 'z':
219            total += 'l';
220            break;
221        case '0':
222            total += '0';
223            break;
224        case '1':
225            total += '1';
226            break;
227        case '2':
228            total += '2';
229            break;
230        case '3':
231            total += '3';
232            break;
233        case '4':
234            total += '4';
235            break;
236        case '5':
237            total += '5';
238            break;
239        case '6':
240            total += '6';
241            break;
242        case '7':
243            total += '7';
244            break;
245        case '8':
246            total += '8';
247            break;
248        case '9':
249            total += '9';
250            break;
251    }
252    return total;
253 }

```

```

98         case 'V':
99             total += 'H';
100            break;
101        case 'W':
102            total += 'O';
103            break;
104        case 'X':
105            total += 'P';
106            break;
107        case 'Y':
108            total += 'G';
109            break;
110        case 'Z':
111            total += 'Y';
112            break;
113        case '0':
114            total += '4';
115            break;
116        case '1':
117            total += '3';
118            break;
119        case '2':
120            total += '0';
121            break;
122        case '3':
123            total += '6';
124            break;
125        case '4':
126            total += '9';
127            break;
128        case '5':
129            total += '5';
130            break;
131        case '6':
132            total += '8';
133            break;
134        case '7':
135            total += '1';
136            break;
137        case '8':
138            total += '7';
139            break;
140        case '9':
141            total += '2';
142            break;
143        case ' ':
144            total += '~';
145            break;
146        case 'a':
147            total += 'n';
148            break;
149        case 'b':
150            total += 'w';
151            break;
152        case 'c':
153            total += 'e';
154            break;
155        case 'd':
156            total += 'c';
157            break;
158        case 'e':
159            total += 'd';
160            break;
161        case 'f':
162            total += 'v';
163            break;
164        case 'g':
165            total += 'b';
166            break;
167        case 'h':
168            total += 'o';
169            break;
170        case 'i':
171            total += 'r';
172            break;
173        case 'j':
174            total += 'l';
175            break;
176        case 'k':
177            total += 't';
178            break;
179        case 'l':
180            total += 's';
181            break;
182        case 'm':
183            total += 'z';
184            break;
185        case 'n':
186            total += 'x';
187            break;
188        case 'o':
189            total += 'y';
190            break;
191        case 'p':
192            total += 'u';
193            break;
194        case 'q':
195            total += 'f';
196            break;
197        case 'r':
198            total += 'g';
199            break;
200        case 's':
201            total += 'h';
202            break;
203        case 't':
204            total += 'i';
205            break;
206        case 'v':
207            total += 'p';
208            break;
209        case 'w':
210            total += 'q';
211            break;
212        case 'x':
213            total += 'n';
214            break;
215        case 'y':
216            total += 'm';
217            break;
218        case 'z':
219            total += 'l';
220            break;
221        case '0':
222            total += '0';
223            break;
224        case '1':
225            total += '1';
226            break;
227        case '2':
228            total += '2';
229            break;
230        case '3':
231            total += '3';
232            break;
233        case '4':
234            total += '4';
235            break;
236        case '5':
237            total += '5';
238            break;
239        case '6':
240            total += '6';
241            break;
242        case '7':
243            total += '7';
244            break;
245        case '8':
246            total += '8';
247            break;
248        case '9':
249            total += '9';
250            break;
251    }
252    return total;
253 }

```

```

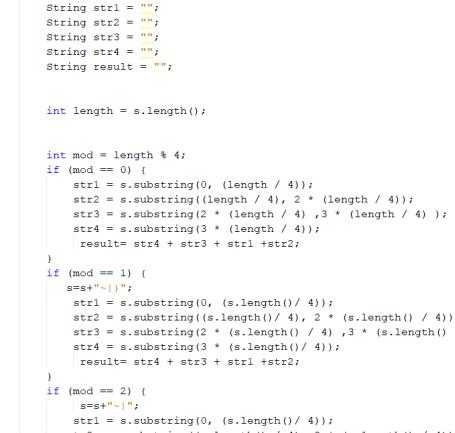
134         case '7':
135             total += '1';
136             break;
137         case '8':
138             total += '7';
139             break;
140         case '9':
141             total += '2';
142             break;
143         case ' ':
144             total += '~';
145             break;
146         case 'a':
147             total += 'n';
148             break;
149         case 'b':
150             total += 'w';
151             break;
152         case 'c':
153             total += 'e';
154             break;
155         case 'd':
156             total += 'c';
157             break;
158         case 'e':
159             total += 'd';
160             break;
161         case 'f':
162             total += 'v';
163             break;
164         case 'g':
165             total += 'b';
166             break;
167         case 'h':
168             total += 'o';
169             break;
170         case 'i':
171             total += 'r';
172             break;
173         case 'j':
174             total += 'l';
175             break;
176         case 'k':
177             total += 't';
178             break;
179         case 'l':
180             total += 's';
181             break;
182         case 'm':
183             total += 'z';
184             break;
185         case 'n':
186             total += 'x';
187             break;
188         case 'o':
189             total += 'y';
190             break;
191         case 'p':
192             total += 'u';
193             break;
194         case 'q':
195             total += 'f';
196             break;
197         case 'r':
198             total += 'g';
199             break;
200         case 's':
201             total += 'h';
202             break;
203         case 't':
204             total += 'i';
205             break;
206         case 'v':
207             total += 'p';
208             break;
209         case 'w':
210             total += 'q';
211             break;
212         case 'x':
213             total += 'n';
214             break;
215         case 'y':
216             total += 'm';
217             break;
218         case 'z':
219             total += 'l';
220             break;
221         case '0':
222             total += '0';
223             break;
224         case '1':
225             total += '1';
226             break;
227         case '2':
228             total += '2';
229             break;
230         case '3':
231             total += '3';
232             break;
233         case '4':
234             total += '4';
235             break;
236         case '5':
237             total += '5';
238             break;
239         case '6':
240             total += '6';
241             break;
242         case '7':
243             total += '7';
244             break;
245         case '8':
246             total += '8';
247             break;
248         case '9':
249             total += '9';
250             break;
251    }
252    return total;
253 }

```

```
173     case 'j':
174         total += 'x';
175         break;
176     case 'k':
177         total += 'm';
178         break;
179     case 'l':
180         total += 't';
181         break;
182     case 'm':
183         total += 's';
184         break;
185     case 'n':
186         total += 'z';
187         break;
188     case 'o':
189         total += 'j';
190         break;
191     case 'p':
192         total += 'k';
193         break;
194     case 'q':
195         total += 'u';
196         break;
197     case 'r':
198         total += 'r';
199         break;
200     case 's':
201         total += 'f';
202         break;
203     case 't':
204         total += 'l';
205         break;
206     case 'u':
207         total += 'i';
```

```
209         case 'v':
210             total += 'h';
211             break;
212         case 'w':
213             total += 'q';
214             break;
215         case 'x':
216             total += 'p';
217             break;
218         case 'y':
219             total += 'g';
220             break;
221         case 'z':
222             total += 'y';
223             break;
224         case '~':
225             total += '!';
226             break;
227         case '|':
228             total += '|';
229             break;
230         case ')':
231             total += ')';
232             break;
233     }
234 }
235
236 }
237
238 }
```

- The permutation method (swap):



```
public String swap(String s) {
    String str1 = "";
    String str2 = "";
    String str3 = "";
    String str4 = "";
    String result = "";

    int length = s.length();

    int mod = length % 4;
    if (mod == 0) {
        str1 = s.substring(0, (length / 4));
        str2 = s.substring((length / 4), 2 * (length / 4));
        str3 = s.substring(2 * (length / 4), 3 * (length / 4));
        str4 = s.substring(3 * (length / 4));
        result = str4 + str3 + str1 + str2;
    }
    if (mod == 1) {
        s += "|";
        str1 = s.substring(0, (s.length() / 4));
        str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
        str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
        str4 = s.substring(3 * (s.length() / 4));
        result = str4 + str3 + str1 + str2;
    }
    if (mod == 2) {
        s += "|";
        str1 = s.substring(0, (s.length() / 4));
        str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
        str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
        str4 = s.substring(3 * (s.length() / 4));
        result = str4 + str3 + str1 + str2;
    }
}
```

```
310         if (mod == 3) {
311             s+=s~"";
312             str1 = s.substring(0, (s.length() / 4));
313             str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
314             str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
315             str4 = s.substring(3 * (s.length() / 4));
316             result= str4 + str3 + str1 +str2;
317         }
318
319         return result;
320     }
321 }
```

- Rail Fence method (Railfence):

```
Security.java x decryption.java x decFramee.java x encFrame.java x encryption.java x |  
Source History | |                  
335 public String RailFence(String s){  
336     String odd ="";  
337     String even ="";  
338  
339     for(int i = 1 ; i<= s.length() ; i++){  
340         if(i % 2 == 0)  
341             even+= s.charAt(i-1);  
342         else odd+= s.charAt(i-1);  
343     }  
344     return odd + even ;  
345 }  
346 }
```

- Converting to ASCII method (ASCII):

```
Security.java x decryption.java x decFramee.java x encFrame.java x encryption.java x lu.png x |  
Source History | |                  
239 public String ASCII(String b) {  
240     int length = b.length();  
241     String tota = "";  
242     int c;  
243  
244     char ch;  
245     for (int i = 0; i < length; i++) {  
246         if (Character.isDigit(b.charAt(i))) {  
247             ch = (char) b.charAt(i);  
248             c=ch;  
249             tota += "%"+c;  
250             continue;  
251         }  
252  
253         if (Character.isUpperCase(b.charAt(i))) {  
254             c = b.charAt(i);  
255             tota += "%"+c;  
256             continue;  
257         }  
258  
259         if(b.charAt(i)=='a'||b.charAt(i)=='b'|| b.charAt(i)=='c'){  
260             c = b.charAt(i);  
261             tota += "%"+c;  
262             continue;  
263         }  
264  
265         c = b.charAt(i);  
266         tota += c;  
267     }  
268     return tota;  
269 }  
270 }
```

- Swap ASCII method (convert Ascii):

```
public String convertAscii(String s){  
    int length = s.length();  
    char[] array = s.toCharArray();  
    String total="";  
    int j=1;  
    int k = 0;  
    for( ; j+2<=length && k+2<=length;j+=2,k+=2){  
        total=total+" "+array[j]+" "+array[k];  
    }  
    total=total+" "+array[j]+" "+array[k];  
  
    return total;  
}
```

- Method (validate):

```

323     public boolean validate(String s){ //return true if the input is valid false other wise
324
325         if(s.length() < 1)
326             return false;
327         for(int i=0 ; i<s.length() ; i++)
328             if (!Character.isLetterOrDigit(s.charAt(i)) && (s.charAt(i) != ' '))
329                 System.out.println("you are allowed to enter digits and characters only");
330             return false;
331
332         return true;
333
334     }

```

Class decryption:

- The substitution method (convert):

```

20  public class decryption {
21      String msg;
22
23
24      public decryption(String msg){
25          this.msg=msg;
26      }
27
28      public String convert(String a) {
29          int length = a.length();
30          String total = "";
31          char c;
32          for (int i = 0; i < length; i++) {
33              c = a.charAt(i);
34              switch (c) {
35                  case 'N':
36                      total += 'A';
37                      break;
38                  case 'W':
39                      total += 'B';
40                      break;
41                  case 'E':
42                      total += 'C';
43                      break;
44                  case 'C':
45                      total += 'D';
46                      break;
47                  case 'D':
48                      total += 'E';
49                      break;
50                  case 'V':
51                      total += 'F';
52                      break;
53                  case 'B':
54                      total += 'G';

```

```

56          case 'O':
57              total += 'H';
58              break;
59          case 'A':
60              total += 'I';
61              break;
62          case 'X':
63              total += 'J';
64              break;
65          case 'M':
66              total += 'K';
67              break;
68          case 'T':
69              total += 'L';
70              break;
71          case 'S':
72              total += 'M';
73              break;
74          case 'Z':
75              total += 'N';
76              break;
77          case 'J':
78              total += 'O';
79              break;
80          case 'K':
81              total += 'P';
82              break;
83          case 'U':
84              total += 'Q';
85              break;
86          case 'R':
87              total += 'R';
88              break;
89          case 'Y':
90              total += 'S';

```

```

Security.java  decryption.java  decFramee.java  encFrame.java  encryption.java
Source History |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
92
93     case 'J':
94         total += 'T';
95         break;
96     case 'I':
97         total += 'U';
98         break;
99     case 'H':
100        total += 'V';
101        break;
102    case 'Q':
103        total += 'W';
104        break;
105    case 'P':
106        total += 'X';
107        break;
108    case 'G':
109        total += 'Y';
110        break;
111    case 'Y':
112        total += 'Z';
113        break;
114    case 'A':
115        total += '0';
116        break;
117    case '3':
118        total += '1';
119        break;
120    case 'O':
121        total += '2';
122        break;
123    case 'E':
124        total += '3';
125        break;
126    case '9':
127        total += '4';

```

```

Security.java  decryption.java  decFramee.java  encFrame.java  encryption.java
Source History |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
128
129     case '5':
130         total += '5';
131         break;
132     case '8':
133         total += '6';
134         break;
135     case '1':
136         total += '7';
137         break;
138     case '7':
139         total += '8';
140         break;
141     case '2':
142         total += '9';
143         break;
144     case '4':
145         total += ' ';
146         break;
147     case 'N':
148         total += 'a';
149         break;
150     case 'W':
151         total += 'b';
152         break;
153     case 'E':
154         total += 'c';
155         break;
156     case 'C':
157         total += 'd';
158         break;
159     case 'D':
160         total += 'e';
161         break;
162     case 'V':
163         total += 'f';

```

```

Security.java  decryption.java  decFramee.java  encFrame.java  encryption.java
Source History |  |  |  |  |  |  |  |  |  |  |  |  |
164
165     case 'B':
166         total += 'g';
167         break;
168     case 'O':
169         total += 'h';
170         break;
171     case 'A':
172         total += 'i';
173         break;
174     case 'X':
175         total += 'j';
176         break;
177     case 'M':
178         total += 'k';
179         break;
180     case 'T':
181         total += 'l';
182         break;
183     case 'S':
184         total += 'm';
185         break;
186     case 'Z':
187         total += 'n';
188         break;
189     case 'J':
190         total += 'o';
191         break;
192     case 'K':
193         total += 'p';
194         break;
195     case 'U':
196         total += 'q';
197         break;
198     case 'R':
199         total += 'r';

```

```

Security.java  decryption.java  decFramee.java  encFrame.java  encryption.java
Source History |  |  |  |  |  |  |  |  |  |  |  |  |
200
201     case 'F':
202         total += 's';
203         break;
204     case 'L':
205         total += 't';
206         break;
207     case 'I':
208         total += 'u';
209         break;
210     case 'H':
211         total += 'v';
212         break;
213     case 'Q':
214         total += 'w';
215         break;
216     case 'P':
217         total += 'x';
218         break;
219     case 'G':
220         total += 'y';
221         break;
222     case 'Y':
223         total += 'z';
224         break;
225     case ' ':
226         total += ' ';
227         break;
228     case ' ':
229         total += ' ';
230         break;
231     case ' ':
232         total += ' ';
233         break;
234     }

```

- The permutation method (swap):

```

256
257     public String swap(String s) {
258         String str1 = "";
259         String str2 = "";
260         String str3 = "";
261         String str4 = "";
262         String result = "";
263
264         int length = s.length();
265
266         int mod = length % 4;
267         if (mod == 0) {
268             str1 = s.substring(0, (length / 4));
269             str2 = s.substring((length / 4), 2 * (length / 4));
270             str3 = s.substring(2 * (length / 4), 3 * (length / 4));
271             str4 = s.substring(3 * (length / 4));
272             result= str3 + str4 + str2 +str1;
273         }
274         if (mod == 1) {
275             str1 = s.substring(0, (s.length() / 4));
276             str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
277             str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
278             str4 = s.substring(3 * (s.length() / 4));
279             result= str3 + str4 + str2 +str1;
280         }
281         if (mod == 2) {
282             str1 = s.substring(0, (s.length() / 4));
283             str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
284             str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
285             str4 = s.substring(3 * (s.length() / 4));
286             result= str3 + str4 + str2 +str1;
287         }
288         if (mod == 3) {
289             str1 = s.substring(0, (s.length() / 4));
290             str2 = s.substring((s.length() / 4), 2 * (s.length() / 4));
291             str3 = s.substring(2 * (s.length() / 4), 3 * (s.length() / 4));
292             str4 = s.substring(3 * (s.length() / 4));
293             result= str3 + str4 + str2 +str1;
294         }
295     }
296
297     return result;
298 }
299
300
301
302
303
304 }

```

- Rail Fence method (Railfence):

```

320
321     public String RialFence(String s){
322         String odd = s.substring(0, (s.length()/ 2));
323         String even =s.substring(s.length()/ 2);
324         String result ="";
325         for(int i= 0 ; i < (s.length()/2) ;i++){
326             result = result + odd.charAt(i) +" "+ even.charAt(i)+" ";
327         }
328         return result ;
329     }
330

```

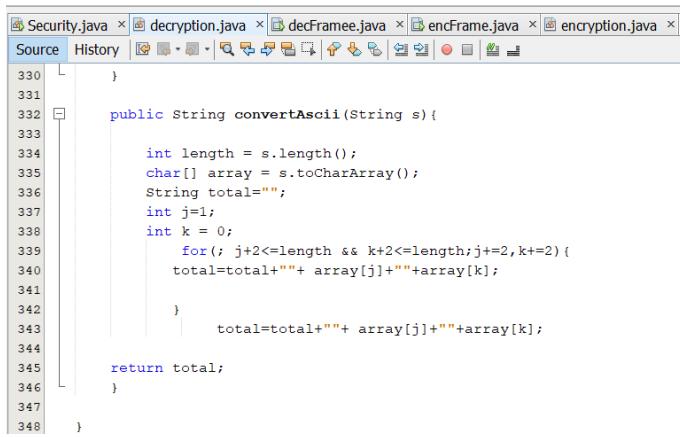
- Converting to ASCII method (ASCII):

```

239
240     public String ASCII(String b) {
241         int F=0;
242         int L =3;
243         char c;
244         String result = " " ;
245         while(F<b.length() && L<b.length()){
246             c=(char)Integer.parseInt(b.substring(F, L));
247             result += c;
248             F+=3;
249             L+=3;
250         }
251         c=(char)Integer.parseInt(b.substring(F));
252         result += c;
253
254         return result;
255     }
256

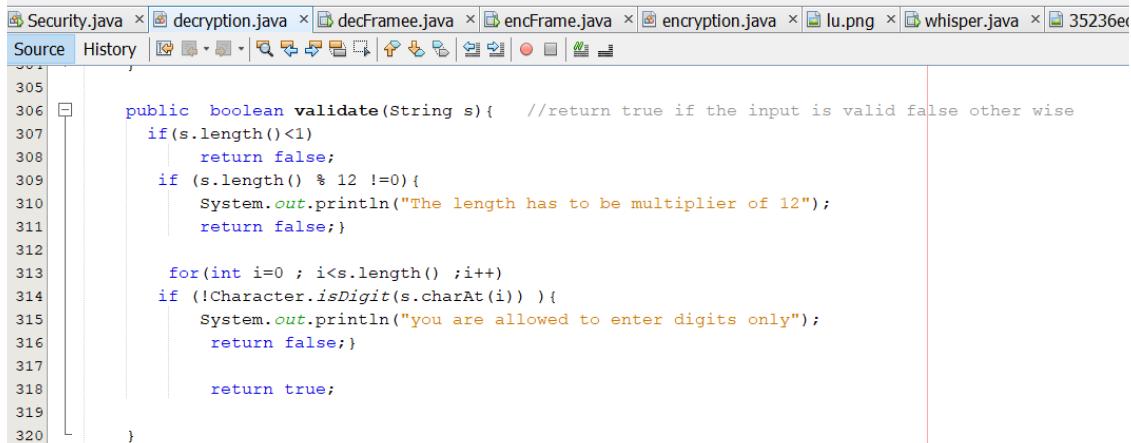
```

- Swap ASCII method (convert Ascii):



```
330     }
331
332     public String convertAscii(String s){
333
334         int length = s.length();
335         char[] array = s.toCharArray();
336         String total="";
337         int j=1;
338         int k = 0;
339         for(; j+2<=length && k+2<=length;j+=2,k+=2){
340             total=total+"<"+array[j]+"+"+array[k];
341
342         }
343         total=total+"<"+array[j]+"+"+array[k];
344
345     return total;
346 }
347
348 }
```

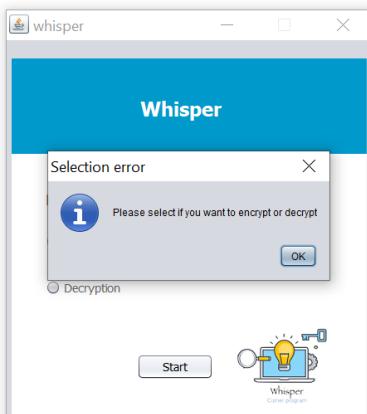
- Method (validate):



```
305
306     public boolean validate(String s){ //return true if the input is valid false otherwise
307         if(s.length()<1)
308             return false;
309         if (s.length() % 12 !=0){
310             System.out.println("The length has to be a multiple of 12");
311             return false;
312
313         for(int i=0 ; i<s.length() ;i++)
314             if (!Character.isDigit(s.charAt(i)) ){
315                 System.out.println("you are allowed to enter digits only");
316                 return false;
317
318             }
319
320     }
321 }
```

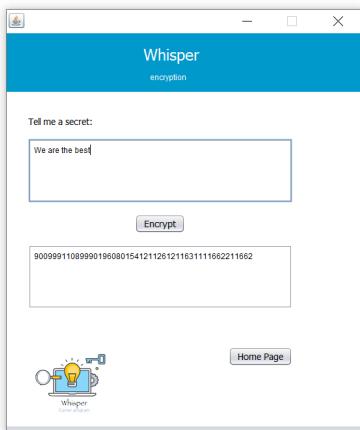
sample of the tests:

1- The user selects Start without selecting an option.

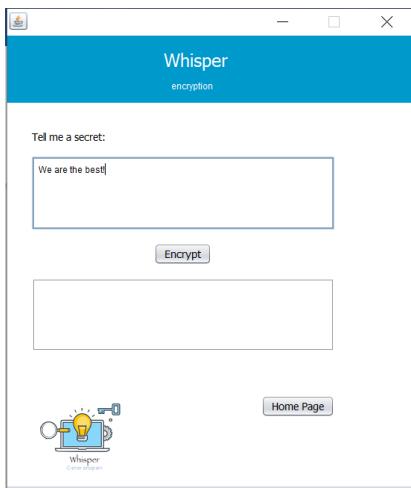


2-Using encryption:

2A – When the user enters a valid input(letters and spaces) and results with a ciphertext.



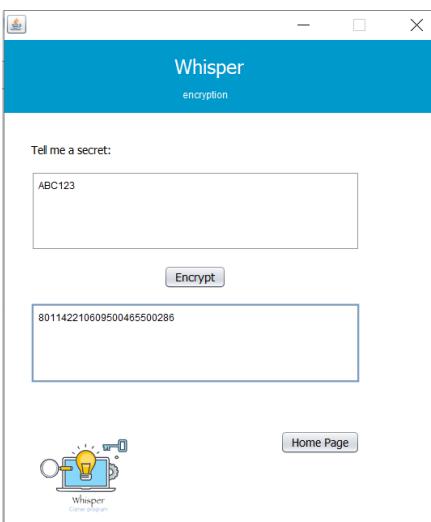
2B- User enters an invalid input (letters with symbols).



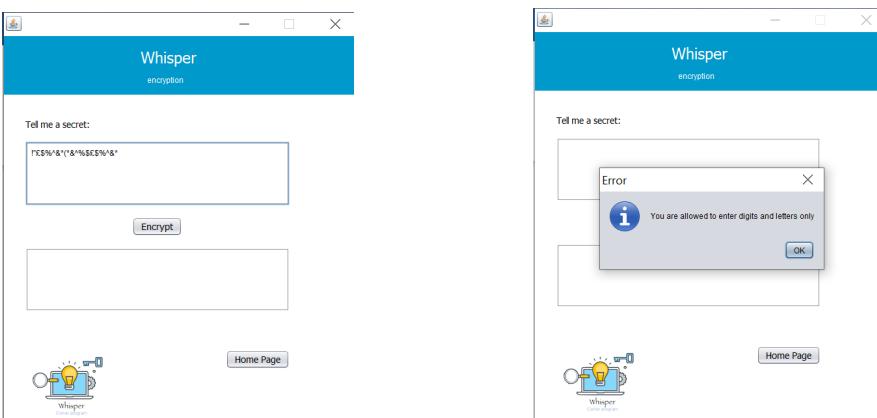
2C- When the user selects on Encrypt without typing a plaintext.



2D- When the user enters a valid input (letters and numbers) and results with a ciphertext.

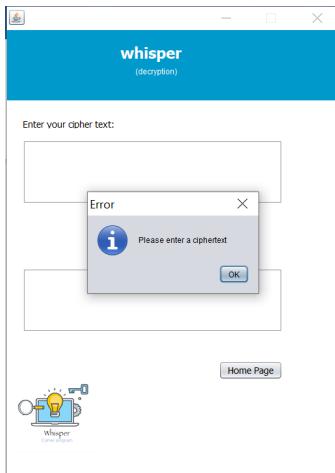


2E- The user enters an invalid input (symbols).

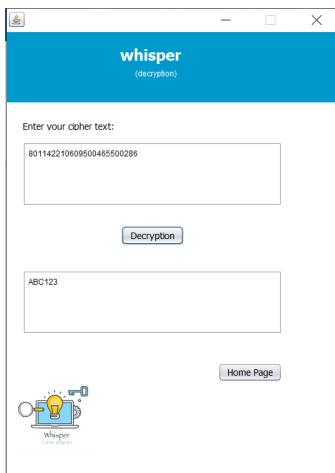


3-Using decryption:

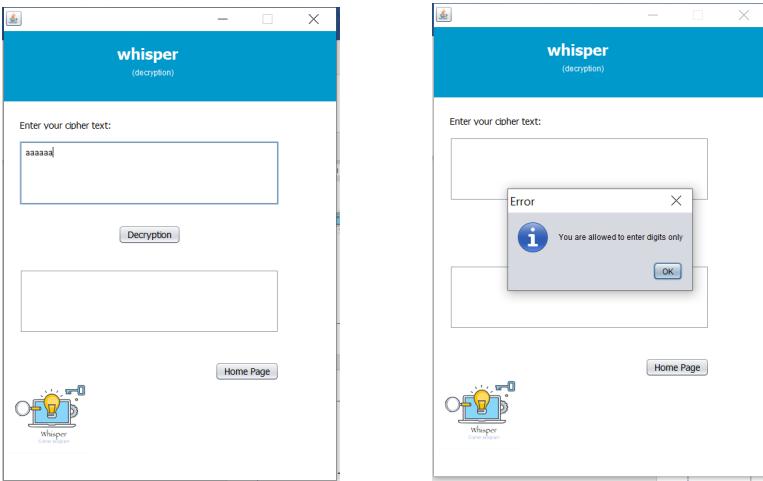
3A – The user selects decrypt without entering a ciphertext.



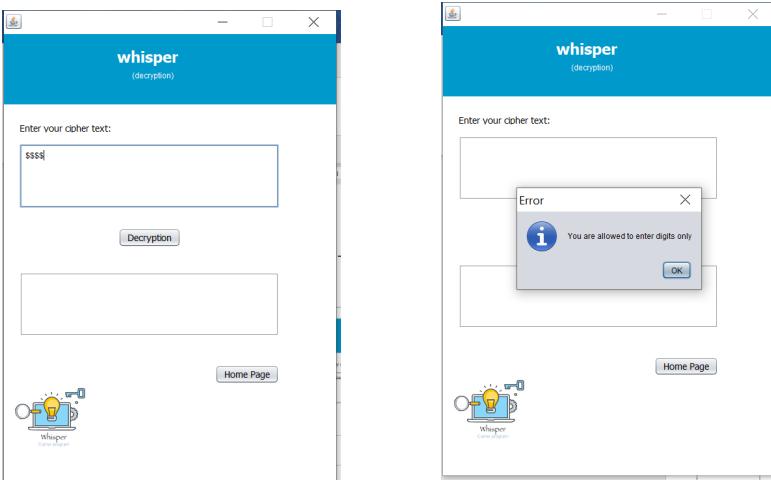
3B – The user enters a valid ciphertext and results with a plaintext.



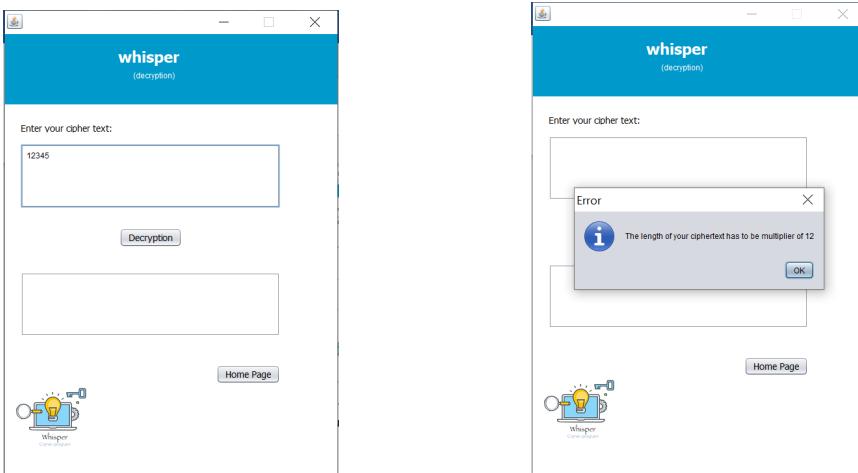
3C – The user enters an invalid input as ciphertext (user enters letters).



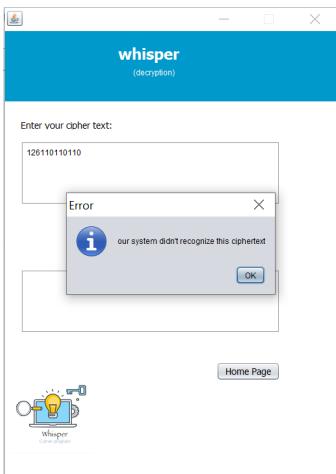
3D- The user enters an invalid input as ciphertext (user enters symbols).



3E – The user enters an invalid input as ciphertext (user enters ciphertext where its length isn't a multiple of 12).



3F- The user enters an invalid input as ciphertext (user enters a ciphertext that the system does not recognize).



At the end, as a team, we have a great experience to work together on this project. And we saw that Whisper has more than three components and it can encrypt and decrypt both letters (dealing with capital and small letters) and numbers which make it a strong model, the two rounds influence the strength of the model. Because of these reasons, Whisper system achieved the goal of security and protected the valuable information.

References :

- [1] Richardson, T. and Thies, C. (2013). *Secure software design*. Burlington, Mass.: Jones & Bartlett Learning.
- [2] Wu, C. (2010). *An introduction to object-oriented programming with Java*. 5th ed. New York: McGraw-Hill Higher Education.
- [3] C. Baziotsis, "How to go back to a JFrames in Java," Stack Overflow, 01-Mar-1962. [Online]. Available: <https://stackoverflow.com/questions/8771558/how-to-go-back-to-a-jframes-in-java/8771756>. [Accessed: 18-Nov-2019].
- [4] P. K. P. Kharbuja, "How to add radio buttons in button group?," Stack Overflow, 01-Feb-1963. [Online]. Available: <https://stackoverflow.com/questions/13832209/how-to-add-radio-buttons-in-button-group>. [Accessed: 18-Nov-2019].
- [5] Java #N3 - Making a GUI in NetBeans. [Online]. Available: <https://www.youtube.com/watch?v=LFr06ZKIpSM>. [Accessed: 18-Nov-2019].
- [6] "GUI Programming - Java Programming Tutorial", *Ntu.edu.sg*, 2019. [Online]. Available: https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html. [Accessed: 19- Nov- 2019].
- [7] J. resizing?, Z. Junejo, I. Gaur, B. Vijay, C. Latte and D. Gavkar, "JFrame: How to disable window resizing?", Stack Overflow, 2019. [Online]. Available: <https://stackoverflow.com/questions/18031704/jframe-how-to-disable-window-resizing>. [Accessed: 19- Nov- 2019].
- [8] J. NetBeans, A. Thompson and Y. K., "JFrame Resizing in Desktop Application - NetBeans", Stack Overflow, 2019. [Online]. Available: <https://stackoverflow.com/questions/8265548/jframe-resizing-in-desktop-application-netbeans>. [Accessed: 19- Nov- 2019].