

**Name: Muhammad Rafiq**

**Seat No: B17101061**

**Course: Network Security & Cryptography (NS'21 Lab)**

**Assignment : Assignment # 6**

**Section: A**

**Task 1: Write a program for Play Fair Cipher that can encrypt and decrypt text it..**

**PlayFair Cipher**

In [1]:

```
1 import numpy as np
2 class PlayFairCipher:
3
4     def __init__(self):
5         print("initialized")
6
7     def make5by5table(self, key):
8         index = 0
9         alphabet = 65
10        l = [[0]*5 for i in range(5)]
11        for i in range(len(l)):
12            for j in range(len(l[0])):
13                if index < len(key):
14                    l[i][j] = key[index]
15                    index += 1
16                else:
17                    flag = True
18                    while(flag):
19                        if chr(alphabet) not in key+"J":
20                            l[i][j] = chr(alphabet)
21                            alphabet += 1
22                            flag = False
23                    else:
24                        alphabet += 1
25
26        return l
27
28    def keyPreprocessing(self, key):
29        key = key.upper()
30        unique_dicts = {}
31        for i in key:
32            if i not in unique_dicts:
33                unique_dicts[i] = 1
34
35        key = "".join([str(e) for e in unique_dicts.keys()])
36        return key
37
38
39
40
41
42
```

```

43 def plaintextPreprocessing(self, plain_text):
44     l = []
45     temp = ""
46     i = 0
47     plain_text = plain_text.upper().replace(" ", "")
48     while i < len(plain_text):
49         if temp == "":
50             temp += plain_text[i]
51         else:
52             if plain_text[i] == temp:
53                 temp += "X"
54                 l.append(temp)
55                 temp = ""
56                 temp += plain_text[i]
57             else:
58                 temp += plain_text[i]
59                 l.append(temp)
60                 temp = ""
61         i += 1
62     return l
63 def encrypt_playCipher(self, plain_text, key):
64     final_list = []
65     key = self.keyPreprocessing(key)
66     plain_text = self.plaintextPreprocessing(plain_text)
67     table = self.make5by5table(key)
68     table = np.array(table)
69     for plain in plain_text:
70         i_start=j_start=i_end=j_end = 0
71         for i in range(len(table)):
72             for j in range(len(table[0])):
73                 if table[i][j] == plain[0]:
74                     i_start, j_start = i , j
75                 elif table[i][j] == plain[1]:
76                     i_end, j_end = i , j
77             if i_start != i_end and j_start != j_end:
78                 final_list.append(table[i_start % 5, j_end % 5] + table[i_end % 5, j_start % 5])
79             elif i_start == i_end:
80                 final_list.append(table[i_start % 5, (j_start + 1) % 5] + table[i_end % 5, (j_end + 1) % 5])
81             elif j_start == j_end:
82                 final_list.append(table[(i_start + 1) % 5, j_start % 5] + table[(i_end + 1) % 5, j_end % 5])
83
84     final_ans = ''.join([str(e) for e in final_list])
85     return final_ans

```

```

86
87     def decrypt_playCipher(self, cipher_text, key):
88         final_list = []
89         key = self.keyPreprocessing(key)
90         cipher_text = self.plaintextPreprocessing(cipher_text)
91         table = self.make5by5table(key)
92         table = np.array(table)
93         for plain in cipher_text:
94             i_start=j_start=i_end=j_end = 0
95             for i in range(len(table)):
96                 for j in range(len(table[0])):
97                     if table[i][j] == plain[0]:
98                         i_start, j_start = i , j
99                     elif table[i][j] == plain[1]:
100                         i_end, j_end = i , j
101             if i_start != i_end and j_start != j_end:
102                 final_list.append(table[i_start % 5,j_end % 5] + table[i_end % 5,j_start % 5])
103             elif i_start == i_end:
104                 final_list.append(table[i_start % 5, (j_start - 1) % 5] + table[i_end % 5, (j_end - 1) % 5])
105             elif j_start == j_end:
106                 final_list.append(table[(i_start - 1) % 5, j_start % 5] + table[(i_end - 1) % 5, j_end % 5])
107
108         final_ans = ''.join([str(e) for e in final_list])
109         return final_ans
110

```

## Task 2: Encrypt the plain text "Palyfair Cipher is completely different way to encrypt the message", where key is "Success".

```

In [2]: 1 play = PlayFairCipher()
2 plain_text= "Playfair Cipher is completely different way to encrypt the message"
3 cipher_text = play.encrypt_playCipher(plain_text, "success")
4
5 print("Text After encryption of a plain text given in a question would become: \n")
6 print(cipher_text)

```

Text After encryption of a plain text given in a question would become:

QKEZHCMOSLTDGYOBSQKRCRAMXBKLCGCYGTZZUZRRSLAYEQOZNGRAUUSMG

In [3]:

```
1 plain_text = play.decrypt_playCipher(cipher_text, "success")
2
3 print("Text after decryption of cipher text which we encrypted before would become: \n")
4 print(plain_text)
```

Text after decryption of cipher text which we encrypted before would become:

PLAYFAIRCIPHERISCOMPLETELYDIFFERENTWAYTOENCRYPTTHEMESSAGE

**-----END-----**

Muhammad Rafta  
BIA101061

Assignment: Play Fair Cipher.

Encrypt "Play Fair cipher is completely different way to encrypt the message"  
Key is "Success"

Generating table of  $5 \times 5$

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

Generating Pairs.

PL AY FA IR CI PH ER IS CO  
MP LE TE LY DE FX FE RE  
NT WA YT OF NC RY PT TH  
BM ES SA GE

Encrypted message would become  
with key = Success.

QKEZ HCMO SLTD GY OBS QKRM C RA  
MX BKLC GYGT ZUZ RRS LAYE  
QOZNGRAHUSMG.