**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**SUBJECT CODE: 19CS3041S**
**CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK**

## 2. Implementation of Playfair Cipher substitution technique

**Date of the Session:** 31/07/21 **Time of the Session: 03:20-05:00**

**Learning Outcomes:**

- To understand the concept of multiple-letter encryption.
- To understand the applications of the technique.

**Pre-Lab Task:**

1. Define diagram with an example.

The definition of a diagram is a graph, chart, drawing or plan that explains something by showing how the parts relate to each other. An example of diagram is **a chart showing how all the departments within an organization are related**

2. What is the reason to consider a 5 × 5 matrix in a playfair cipher technique?

### Generating the Key Square

- The 'key square' is a 5×5 grid consisting of alphabets that helps encrypt the plain text.
- All these 25 letters should be unique.
- Since the grid can accommodate only 25 characters, there is no 'J' in this table. Any 'J' in the plaintext is replaced by 'I'.
- Remove any characters or punctuation that are not present in the key square. Instead, spell out the numbers, punctuations, and any other non-alphabetic text.
- The key square will start with the key's unique alphabet in the order of appearance, followed by the alphabet's remaining characters in order

3. What to do if letters in plain text reoccur eg: *Hello*?

**1.** Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter.

**Plain Text:** “hello”

**After Split:** ‘he’ ‘lx’ ‘lo’

Here **‘x’** is the bogus letter.

**2.** If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

**Plain Text:** “helloe”

**AfterSplit:** ‘he’ ‘lx’ ‘lo’ ‘lz’

Here **‘z’** is the bogus letter.

4. What are the advantages of Playfair cipher?

**Advantages:**

1. It is significantly harder to break since the frequency analysis technique used to break simple substitution ciphers is difficult but still can be used on (25*25) = 625 digraphs rather than 25 monographs which is difficult.

2. Frequency analysis thus requires more cipher text to crack the encryption.

5. Trace what will be the encrypted message by using Playfair cipher if the message is ‘balloon’ and the key is “Monarchy”.

m → key underlined letters

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

5 x 5

Key:- MONARCHY

plain text:- Balloon.

- we need to fill the matrix 5 x 5 other than key. we need to go line wise A – Z and check if A is written then do not write and move to the next which is B if is not present in the key then write in the matrix

plain text:- b a ll o o n.

↓ divide into 2

b a | x | l o | o n.

cipher text:-

i b su pm na.

## In-Lab Task:

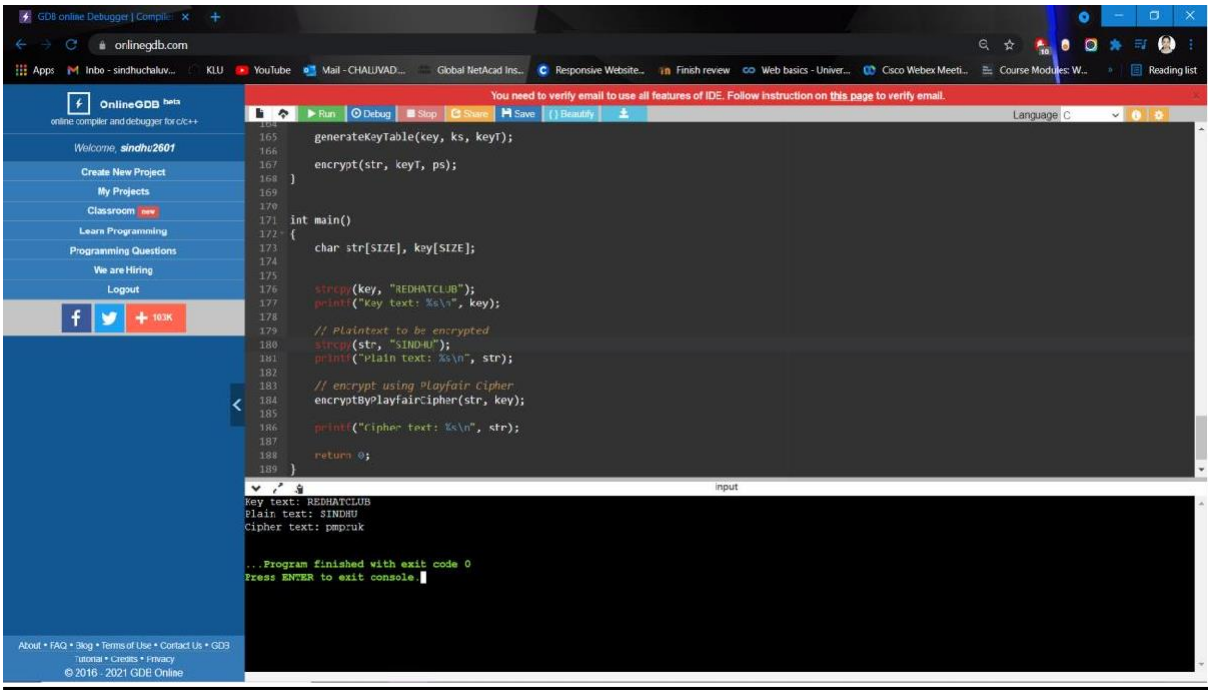
1) Write a code to implement Playfair Cipher Substitution Technique for the following input:
Sample Input:-

Plain Text: “Student to consider his/her name”

Secret Key: REDHATCLUB

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

*(**Note**: Ignore the whitespace and consider the text)*




## POST Task:

1) Write a Pseudocode for Playfair Cipher Substitution technique.

Sol)

```
#include  <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 30


void toLowerCase(char plain[], int ps)
{
  int i;
  for (i = 0; i < ps; i++) {
          if (plain[i] > 64 && plain[i] < 91)
                  plain[i] += 32;
  }
}


int removeSpaces(char* plain, int ps)
{
  int i, count = 0;
  for (i = 0; i < ps; i++)
          if (plain[i] != ' ')
                  plain[count++] = plain[i];
  plain[count] = '\0';
  return count;
}

// Function to generate the 5x5 key square
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
  int i, j, k, flag = 0, *dicty;
```

```
// a 26 character hashmap
// to store count of the alphabet
dicty = (int*)calloc(26, sizeof(int));
for (i = 0; i < ks; i++) {
        if (key[i] != 'j')
                dicty[key[i] - 97] = 2;
}

dicty['j' - 97] = 1;

i = 0;
j = 0;

for (k = 0; k < ks; k++) {
        if (dicty[key[k] - 97] == 2) {
                dicty[key[k] - 97] -= 1;
                keyT[i][j] = key[k];
                j++;
                if (j == 5) {
                        i++;
                        j = 0;
                }
        }
}

for (k = 0; k < 26; k++) {
        if (dicty[k] == 0) {
                keyT[i][j] = (char)(k + 97);
                j++;
                if (j == 5) {
                        i++;
                        j = 0;
                }
        }
```

```
  }
}

// Function to search for the characters of a digraph
// in the key square and return their position
void search(char keyT[5][5], char a, char b, int arr[])
{
  int i, j;

  if (a == 'j')
        a = 'i';
  else if (b == 'j')
        b = 'i';

  for (i = 0; i < 5; i++) {

        for (j = 0; j < 5; j++) {

              if (keyT[i][j] == a) {
                    arr[0] = i;
                    arr[1] = j;
              }
              else if (keyT[i][j] == b) {
                     arr[2] = i;
                    arr[3] = j;
              }
        }
  }
}

// Function to find the modulus with 5
int mod5(int a)
{
  return (a % 5);
```

```
}

// Function to make the plain text length to be even
int prepare(char str[], int ptrs)
{
  if (ptrs % 2 != 0) {
        str[ptrs++] = 'z';
        str[ptrs] = '\0';
  }
  return ptrs;
}

// Function for performing the encryption
void encrypt(char str[], char keyT[5][5], int ps)
{
  int i, a[4];

  for (i = 0; i < ps; i += 2) {

        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {
              str[i] = keyT[a[0]][mod5(a[1] + 1)];
              str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3]) {
              str[i] = keyT[mod5(a[0] + 1)][a[1]];
              str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
        else {
              str[i] = keyT[a[0]][a[3]];
              str[i + 1] = keyT[a[2]][a[1]];
        }
  }
```

```
}

// Function to encrypt using Playfair Cipher
void encryptByPlayfairCipher(char str[], char key[])
{
  char ps, ks, keyT[5][5];

  // Key
  ks = strlen(key);
  ks = removeSpaces(key, ks);
  toLowerCase(key, ks);

  // Plaintext
  ps = strlen(str);
  toLowerCase(str, ps);
  ps = removeSpaces(str, ps);

  ps = prepare(str, ps);

  generateKeyTable(key, ks, keyT);

  encrypt(str, keyT, ps);
}


int main()
{
  char str[SIZE], key[SIZE];


  strcpy(key, "REDHATCLUB");
  printf("Key text: %s\n", key);

  // Plaintext to be encrypted
```

```
  strcpy(str, "SINDHU");
  printf("Plain text: %s\n", str);

  // encrypt using Playfair Cipher
  encryptByPlayfairCipher(str, key);

  printf("Cipher text: %s\n", str);

  return 0;
}
```

*(For Evaluator's use only)*

| Comment of the Evaluator (if Any) | Evaluator's Observation |
|---|---|
| | Marks Secured:_____out of _____<br>Full Name of the Evaluator:<br>Signature of the Evaluator Date of Evaluation |