Amrita Vishwa Vidyapeetham, Amritapuri

Department of Computer Science and Engineering

22AIE314 Computer Security

Lab Sheet 1

Classical Encryption Methods

1. Implement Caesar cipher.
   a) Given plaintext "DEFEND THE EAST WALL" and a shift key 3, encrypt the message using the Caesar Cipher.
   b) Decrypt the given ciphertext "WKH HDJOH LV LQ SODFH" assuming shift key 3.

```
def encrypt_Caesar(text, key):
  return ''.join([chr(ord(i)+key) if i.isalpha() else i for i in text])

print(encrypt_Caesar("DEFEND THE EAST WALL", 3))
```

```
GHIHQG WKH HDVW ZDOO
```

```
def decrypt_Caesar(text, key):
  return ''.join([chr(ord(i)-key) if i.isalpha() else i for i in text])
print(decrypt_Caesar("GHIHQG WKH HDVW ZDOO", 3))
print(decrypt_Caesar("WKH HDJOH LV LQ SODFH", 3))
```

```
DEFEND THE EAST WALL
THE EAGLE IS IN PLACE
```

2. Implement Monoalphabetic Cipher
   a) Encrypt the plaintext "HELLO WORLD" using a random substitution key.
   b) Decrypt the given ciphertext "XUBBE MEHBT" using the provided key mapping: {H: X, E: U, L: B, O: E, W: M, R: H, D: T}.

```
from random import shuffle
key = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
shuffle(key)
print(key)
```

```
['Q', 'Z', 'C', 'W', 'N', 'G', 'S', 'D', 'L', 'F', 'E', 'T', 'U', 'I', 'X', 'B', 'M', 'J', 'R', 'Y', 'K', 'H', 'V', 'A', 'O', 'P']
```

```
def monoencrypt(text, key):
  return ''.join([key[ord(i.upper())-65] if i.isalpha() else i for i in text])

print(monoencrypt("HELLO WORLD", key))
```

```
DNTTX VXJTW
```

```
def monodecrypt(text, key):
  return ''.join([chr(key.index(i.upper())+65) if i.isalpha() else i for i in text])

print(monodecrypt("YLXXE OEWXJ", key))
```

```
TIOOK YKDOR
```

```
key = ['Q', 'Y', 'L', 'T', 'U', 'N', 'P', 'X', 'S', 'J', 'Z', 'B', 'V', 'F', 'E', 'K', 'G', 'H', 'A', 'W', 'C', 'I', 'M', 'O', 'R', 'D']

print(monodecrypt("XUBBE MEHBT", key))
```

```
HELLO WORLD
```

3. Implement Playfair Cipher.
   a) Encrypt the plaintext "MEET ME AT THE PARK" using the Playfair Cipher with key "SECURITY".
   b) Decrypt the given ciphertext "GATLMZ CLRSPB" assuming the same Playfair key.

```
def gen_key_mat(key):
  import numpy as np
  key.replace('J', 'I')
  mat = np.array(['#']*25)
  ind = 0
  for i in key:
    mat[ind] = i
    ind+=1
  mat[ind:] = sorted(list(set([chr(i) for i in range(65, 91)]) - set(key) - set('J')))
  mat.resize((5, 5))
  return mat


def playfair(text, keymat):
  if " " in text:
    return ' '.join([playfair(i, keymat) for i in text.split(' ')])
  if len(text) == 1:
    if text != 'X':
      text+='X'
    elif text !='Y':
      text+='Y'
    else:
      text+='Z'
    l1, l2 = np.where(keymat==text[0]), np.where(keymat==text[1])
    if l1[0] == l2[0]:
      return keymat[(l1[0]+1)%5, l1[1]][0]+keymat[(l2[0]+1)%5, l2[1]][0]
    elif l1[1] == l2[1]:
      return keymat[l1[0], (l1[1]+1)%5][0]+keymat[l2[0], (l2[1]+1)%5][0]
    elif l1[0]<l2[0]:
      return keymat[l1[0], l2[1]][0]+keymat[l2[0], l1[1]][0]
    else:
      return keymat[l2[0], l1[1]][0]+keymat[l1[0], l2[1]][0]
  else:
    # print(text)
    cipher = ''
    count, n = 0, len(text)
    l = []
    if n%2!=0:
      if text[-1]!='X':
        text+='X'
      elif text[-1]!='Y':
        text+='Y'
      else:
        text+='Z'
    while count <n:
      if text[count] != text[count+1]:
        l.append(text[count:count+2])
        count+=2
      elif text[count]!='X':
        l.append(text[count]+'X')
        count+=1
      elif text[count]!='Y':
        l.append(text[count]+'Y')
        count+=1
      else:
        l.append(text[count]+'Z')
        count+=1

    text=''.join(l)
    for i in range(1, len(text), 2):
      l1, l2 = np.where(keymat==text[i-1]), np.where(keymat==text[i])
      l1, l2 = [l1[0][0], l1[1][0]], [l2[0][0], l2[1][0]]
      # print('cipher', cipher, 'c1', text[i-1], 'c2', text[i], 'l1', l1, 'l2', l2)
      if l1[0] == l2[0]:
        # print('cond', 1)
        cipher+= keymat[l1[0], (l1[1]+1)%5][0]+keymat[l2[0], (l2[1]+1)%5][0]
      elif l1[1] == l2[1]:
        # print('cond', 2)
```

```
        cipher+= keymat[(l1[0]+1)%5, l1[1]][0]+keymat[(l2[0]+1)%5, l2[1]][0]
      elif l1[0]<l2[0]:
        # print('cond', 3)
        cipher+= keymat[l1[0], l2[1]][0]+keymat[l2[0], l1[1]][0]
      else:
        # print('cond', 4)
        cipher+= keymat[l2[0], l1[1]][0]+keymat[l1[0], l2[1]][0]
    # print('final', cipher)
    return cipher

text = "MEET ME AT THE PARK"
key = "SECURITY"
print(text, '\n', key)
keymat = gen_key_mat(key)
print(keymat)
cipher = playfair(text, keymat)
print(cipher)
```

```
MEET ME AT THE PARK
 SECURITY
[['S' 'E' 'C' 'U' 'R']
 ['I' 'T' 'Y' 'A' 'B']
 ['D' 'F' 'G' 'H' 'K']
 ['L' 'M' 'N' 'O' 'P']
 ['Q' 'V' 'W' 'X' 'Z']]
VTTF VT BY AFUV BOBP
```

```
def PlayFairDecrypt(text, keymat):
  if ' ' in text:
    return ' '.join([PlayFairDecrypt(i, keymat) for i in text.split(' ')])
  else:
    import numpy as np
    decipher = ''
    for i in range(1, len(text), 2):
      l1, l2 = np.where(keymat==text[i-1]), np.where(keymat==text[i])
      l1, l2 = [l1[0][0], l1[1][0]], [l2[0][0], l2[1][0]]
      if l1[0] == l2[0]:
        decipher+= keymat[l1[0], (l1[1]-1)%5][0]+keymat[l2[0], (l2[1]-1)%5][0]
      elif l1[1] == l2[1]:
        decipher+= keymat[(l1[0]-1)%5, l1[1]][0]+keymat[(l2[0]-1)%5, l2[1]][0]
      elif l1[0]<l2[0]:
        decipher+= keymat[l1[0], l2[1]][0]+keymat[l2[0], l1[1]][0]
      else:
        decipher+= keymat[l1[0], l2[1]][0]+keymat[l2[0], l1[1]][0]
    return decipher

cipher = 'GATLMZ CLRSPB'
print('Cipher: ', cipher)
print('Key: ', key)
print("Key Diagram:\n", keymat)
text = PlayFairDecrypt(cipher, keymat)
print('Decrypted Text: ', text)
```

```
Cipher:  GATLMZ CLRSPB
Key:  SECURITY
Key Diagram:
 [['S' 'E' 'C' 'U' 'R']
 ['I' 'T' 'Y' 'A' 'B']
 ['D' 'F' 'G' 'H' 'K']
 ['L' 'M' 'N' 'O' 'P']
 ['Q' 'V' 'W' 'X' 'Z']]
Decrypted Text:  HYIMPV SNURKR
```

4. Implement Hill Cipher (Use a 3x3 matrix for encryption)
   a) Encrypt the plaintext "ACT" using a 3x3 key matrix:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

   b) Decrypt the ciphertext "POH" using the inverse of the given key matrix.

```
Start coding or generate with AI.
```

5. Implement Polyalphabetic Cipher (Vigenère Cipher) -Use a repeating key to encrypt the message.
    a) Encrypt the plaintext "ATTACK AT DAWN" using the key "LEMON".
    b) Decrypt the ciphertext "LXFOPV EF RNHR" using the same key.

```
Start coding or generate with AI.
```

6. Implement One-Time Pad Cipher
    a) Encrypt the plaintext "HELLO" using the one-time pad key "XMCKL".
    b) Decrypt the ciphertext "EQNVZ" using the same key

```
Start coding or generate with AI.
```