Cryptography and Network Security Lab

Assignment 3

Student Details

Name : Krunal Rank
Adm No. : U18CO081

1

```
# sample text : once upon a time there was a little girl named goldilocks she went
for a walk in the forest pretty soon she came upon a house she knocked and when no
one answered she walked right in at the table in the kitchen there were three bowls
of porridge goldilocks was hungry she tasted the porridge from the first bowl

class ERRORS:
    """
    Error Messages
    """
    INVALID_KEY = (
        """Given Key is invalid for Playfair Cipher.
Rules while deciding a key:
1. Key must contain only Lowercase Alphabetical Letters.
2. Key must contain unique Alphabetical Letters.
3. Key must not be empty.
4. Key must contain at most 25 characters.
"""
    )
    INVALID_CHOICE = "Please enter a valid Integer choice."
    INVALID_TEXT = "Please enter a valid Text. Text must only contain Lowercase
Alphabets."


def playfair_cipher_encrypt(text: str, key: str) -> str:
    """
    Encrypts text using Playfair cipher encryption technique
    """
    # Validating Input

    if len(text) == 0:
        raise Exception(ERRORS.INVALID_TEXT)
    if len(key) == 0 or len(key) > 25:
        raise Exception(ERRORS.INVALID_KEY)
```

```
    text = text.lower()
    for letter in text:
        if not (letter.isalpha() or letter.isspace()):
            raise Exception(ERRORS.INVALID_TEXT)

    key = key.lower()
    key_chars = []
    for letter in key:
        if letter in key_chars or not letter.isalpha():
            raise Exception(ERRORS.INVALID_KEY)
        key_chars.append(letter)

    # Constructing 2D  5 x 5 Key Matrix
    # Storing it in list and using Row Major notation
    LETTER_INDICES = [-1 for i in range(26)]
    INDEX_LETTERS = ["" for i in range(25)]

    pos = 0
    for letter in key:
        LETTER_INDICES[ord(letter)-ord('a')] = pos
        INDEX_LETTERS[pos] = letter
        pos += 1

    for i in range(26):
        # skip "j"
        if i == 9:
            continue
        # assign matrix position to remaining letters
        if LETTER_INDICES[i] == -1:
            LETTER_INDICES[i] = pos
            INDEX_LETTERS[pos] = chr(ord('a') + i)
            pos += 1

    # Replace "j" in text with "i"
    text = text.replace("j", "i")

    # Make pairs of letters
    text_list = list(text.replace(" ", ""))

    pairs = []
    text_list_pos = 0
    while text_list_pos < len(text_list):
        # Check for odd length remaining characters
        if text_list_pos == len(text_list) - 1:
            if text_list[text_list_pos] == "z":
```

```
                pairs.append(("z", "y"))
            else:
                pairs.append((text_list[text_list_pos], "z"))
            text_list_pos += 1
            break

        # Check for repetitive characters
        if text_list[text_list_pos] == text_list[text_list_pos + 1]:
            if text_list[text_list_pos] == "x":
                pairs.append(("x", "y"))
            else:
                pairs.append((text_list[text_list_pos], "x"))
            text_list_pos += 1
            continue

        pairs.append((text_list[text_list_pos], text_list[text_list_pos+1]))
        text_list_pos += 2

    # Constructing Ciphertext
    cipher_text_list = []
    for (char1, char2) in pairs:
        char1_pos = ord(char1) - ord('a')
        char2_pos = ord(char2) - ord('a')
        char1_row, char1_col = LETTER_INDICES[char1_pos]//5, LETTER_INDICES[char1_pos]
% 5
        char2_row, char2_col = LETTER_INDICES[char2_pos]//5, LETTER_INDICES[char2_pos]
% 5

        if char1_row == char2_row:
            cipher_char1_pos = char1_row*5 + (char1_col + 1) % 5
            cipher_char2_pos = char2_row*5 + (char2_col + 1) % 5
            cipher_text_list.append(INDEX_LETTERS[cipher_char1_pos])
            cipher_text_list.append(INDEX_LETTERS[cipher_char2_pos])
        elif char1_col == char2_col:
            cipher_char1_pos = ((char1_row + 1) % 5)*5 + (char1_col) % 5
            cipher_char2_pos = ((char2_row + 1) % 5)*5 + (char2_col) % 5
            cipher_text_list.append(INDEX_LETTERS[cipher_char1_pos])
            cipher_text_list.append(INDEX_LETTERS[cipher_char2_pos])
        else:
            cipher_char1_pos = char1_row*5 + char2_col
            cipher_char2_pos = char2_row*5 + char1_col
            cipher_text_list.append(INDEX_LETTERS[cipher_char1_pos])
            cipher_text_list.append(INDEX_LETTERS[cipher_char2_pos])

    return "".join(cipher_text_list)
```

```
def playfair_cipher_decrypt(text: str, key: str) -> str:
    """
    Decrypts text using Playfair cipher encryption technique
    """
    # Validating Input
    if len(text) == 0 or len(text) % 2 != 0:
        raise Exception(ERRORS.INVALID_TEXT)
    if len(key) == 0 or len(key) > 25:
        raise Exception(ERRORS.INVALID_KEY)

    text = text.lower()
    for letter in text:
        if not (letter.isalpha() or letter.isspace()):
            raise Exception(ERRORS.INVALID_TEXT)

    # Check for "j" in text
    if "j" in text:
        raise Exception(ERRORS.INVALID_TEXT)

    key = key.lower()
    key_chars = []
    for letter in key:
        if letter in key_chars or not letter.isalpha():
            raise Exception(ERRORS.INVALID_KEY)
        key_chars.append(letter)

    # Constructing 2D  5 x 5 Key Matrix
    # Storing it in list and using Row Major notation
    LETTER_INDICES = [-1 for i in range(26)]
    INDEX_LETTERS = ["" for i in range(25)]

    pos = 0
    for letter in key:
        LETTER_INDICES[ord(letter)-ord('a')] = pos
        INDEX_LETTERS[pos] = letter
        pos += 1

    for i in range(26):
        # skip "j"
        if i == 9:
            continue
        # assign matrix position to remaining letters
        if LETTER_INDICES[i] == -1:
            LETTER_INDICES[i] = pos
            INDEX_LETTERS[pos] = chr(ord('a') + i)
```

```
            pos += 1

    # Make pairs of letters
    text_list = list(text.replace(" ", ""))

    pairs = []
    text_list_pos = 0
    while text_list_pos < len(text_list):
        pairs.append((text_list[text_list_pos], text_list[text_list_pos+1]))
        text_list_pos += 2

    # Constructing Ciphertext
    decipher_text_list = []
    for (char1, char2) in pairs:
        char1_pos = ord(char1) - ord('a')
        char2_pos = ord(char2) - ord('a')
        char1_row, char1_col = LETTER_INDICES[char1_pos]//5, LETTER_INDICES[char1_pos]
% 5
        char2_row, char2_col = LETTER_INDICES[char2_pos]//5, LETTER_INDICES[char2_pos]
% 5

        if char1_row == char2_row:
            decipher_char1_pos = char1_row*5 + (char1_col + 4) % 5
            decipher_char2_pos = char2_row*5 + (char2_col + 4) % 5
            decipher_text_list.append(INDEX_LETTERS[decipher_char1_pos])
            decipher_text_list.append(INDEX_LETTERS[decipher_char2_pos])
        elif char1_col == char2_col:
            decipher_char1_pos = ((char1_row + 4) % 5)*5 + (char1_col) % 5
            decipher_char2_pos = ((char2_row + 4) % 5)*5 + (char2_col) % 5
            decipher_text_list.append(INDEX_LETTERS[decipher_char1_pos])
            decipher_text_list.append(INDEX_LETTERS[decipher_char2_pos])
        else:
            decipher_char1_pos = char1_row*5 + char2_col
            decipher_char2_pos = char2_row*5 + char1_col
            decipher_text_list.append(INDEX_LETTERS[decipher_char1_pos])
            decipher_text_list.append(INDEX_LETTERS[decipher_char2_pos])

    return "".join(decipher_text_list)


def playfair_cipher_encrypt_dialog():
    """
    Runs Playfair Cipher Encryption Dialog
    """
    text = input("Enter text to be encrypted: ")
    key = input("Enter key: ")
```

```
    encrypted_text = playfair_cipher_encrypt(text, key)

    print(f"Encrypted Text: {encrypted_text}")


def playfair_cipher_decrypt_dialog():
    """
    Runs Playfair Cipher Decryption Dialog
    """
    text = input("Enter text to be decrypted: ")
    key = input("Enter key: ")

    decrypted_text = playfair_cipher_decrypt(text, key)

    print(f"Decrypted Text: {decrypted_text}")


def main_dialog():
    """
    Runs main dialog sequence
    """
    try:
        choice = int(input(
            "Playfair Cipher Program\n1. Encrypt\n2. Decrypt\nPlease enter your
choice: "))
    except Exception as e:
        raise Exception(ERRORS.INVALID_CHOICE)

    if choice == 1:
        playfair_cipher_encrypt_dialog()
    elif choice == 2:
        playfair_cipher_decrypt_dialog()
    else:
        raise Exception(ERRORS.INVALID_CHOICE)


if __name__ == "__main__":
    try:
        main_dialog()
    except Exception as e:
        print(e)
```

```
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/CNS/Assignment_3$ python3 1.py
Playfair Cipher Program
1. Encrypt
2. Decrypt
Please enter your choice: 1
Enter text to be encrypted: once upon a time there was a little girl named goldilocks she went for a walk in the forest pretty
soon she came upon a house she knocked and when no one answered she walked right in at the table in the kitchen there were thre
e bowls of porridge goldilocks was hungry she tasted the porridge from the first bowl
Enter key: svnit
Encrypted Text: pvdaxmpvessqlelczlubafktnzerbltqhtfuaeowketkpbfinfbzctslpmbuefqdislcgmzlvsqmleizvmpvnfadfuazqpscgpsaatlchipblde
bicxgctvppvabivzbzlailcubflaeqthkstscnznllebcrltinldltshpctnllzbzlzlelpczacwvftmgqppzqtbkblrgkdgrdhvufafxvhqznflefaeleilcqppzqt
bkalmprslcksmtvewvrt
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/CNS/Assignment_3$ python3 1.py
Playfair Cipher Program
1. Encrypt
2. Decrypt
Please enter your choice: 2
Enter text to be decrypted: pvdaxmpvessqlelczlubafktnzerbltqhtfuaeowketkpbfinfbzctslpmbuefqdislcgmzlvsqmleizvmpvnfadfuazqpscgps
aatlchipbldebicxgctvppvabivzbzlailcubflaeqthkstscnznllebcrltinldltshpctnllzbzlzlelpczacwvftmgqppzqtbkblrgkdgrdhvufafxvhqznflefa
eleilcqppzqtbkalmprslcksmtvewvrt
Enter key: svnit
Decrypted Text: onceuponatimetherewasalitxtlegirlnamedgoldilocksshewentforawalkintheforestprettysoonshecameuponahousesheknocked
andwhennooneansweredshewalkedrightinatxthetableinthekitchentherewerethrexebowlsofporxridgegoldilockswashungryshetastedtheporxri
dgefromthefirstbowlz
```