MainPage/Info Security/Encript Lab 1.1

Университет ИТМО
Факультет ФПИ и КТ

Отчёт по лабораторной работе 1

«Основы шифрования данных»

по дисциплину

«Информационная безопасность»

Студент: Чжоу Хунсян
Группа: P33131
Преподаватель:

Санкт-Петербург 2024

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Варианты заданий

> Вариант: 23 % 10 = 3

Реализовать шифрование и дешифрацию файла с использованием метода биграмм. Ключевое слово вводится.

Исходный код

Функция для генерации матрицы ключи

У нас на русском языке существуют 33 буквы, поэтому мы добавил 3 буквы для генерации матрицы с размером 6*6=36

```
from lib2to3.main import diff_texts

alphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯABC"  # Add English letter A,B,C to build 6*6


# function for generating key matrix
def generate_key_matrix(key):
    key = ''.join(sorted(set(key.upper()), key=lambda x: key.index(x)))  # remove repeat
    key_matrix = []
    key_matrix_row = []
    used = set()

    for letter in key:
        if letter not in used:
            key_matrix_row.append(letter)
            used.add(letter)
        if key_matrix_row.__len__() == 6:
            key_matrix.append(key_matrix_row)
            key_matrix_row = []

    for letter in alphabet:
        if letter not in used:
            key_matrix_row.append(letter)
            used.add(letter)
        if key_matrix_row.__len__() == 6:
            key_matrix.append(key_matrix_row)
            key_matrix_row = []

    return key_matrix
```

Функция для получения позиции букв в матрице

```
# function to find position of letter in key matrix
def find_position(key_matrix, letter):
    for row in range(6):
        for col in range(6):
            if key_matrix[row][col] == letter.upper():
                return row, col
```

Некоторые вспомогательные функции

```
# function to judge whether a character is an english letter
def is_letter_in_alphabet(char):
    return char in alphabet or char in alphabet.lower()


# function to transform text pair matrix to string
def text_pairs_to_string(text_pairs):
    text = ""
    for text_pair in text_pairs:
        for char in text_pair:
            text += char
    return text


# function to print matrix
def print_matrix(matrix):
    for row in matrix:
        print(row)


# function to get english letters from pairs
def get_letter_from_pair(text_pair):
    pair_letter = []
    for letter in text_pair:
        if is_letter_in_alphabet(letter):
            pair_letter.append(letter)
    return pair_letter
```

Функция предварительной обработки

Здесь мы разделили исходный текст в форме text pairs, у которого каждый содержит 2 буквы в порядке из исходного текста, и в ходе обработке мы пробускали другие знаки и пробелы и сохраняли первоначальную капитализацию.

```
# function to preprocess_text
# preprocess
def preprocess_text(text):
    # text = text.replace("J", "I").replace("j", "i")  # replace J with I and remove space
    processed_text_pairs = []
    pair = []
    pair_letter = []
    i = 0

    while i < len(text):
        pair.append(text[i])
        if is_letter_in_alphabet(text[i]):
            pair_letter.append(text[i])
        if pair_letter.__len__() == 2:
            if pair_letter[0] == pair_letter[1]:  # if there are two same letters in one p
                pair.reverse()
                pair.remove(pair_letter[1])
                pair.reverse()
                pair_letter.remove(text[i])
                pair.append('A')
                processed_text_pairs.append(pair)
                pair_letter = []
                pair = []
                i -= 1
            else:
                processed_text_pairs.append(pair)
                pair_letter = []
                pair = []
        if i == len(
                text) - 1 and pair_letter.__len__() == 1:  # if in the end the length of t
            # in the end
            pair.append('A')
            processed_text_pairs.append(pair)

        i += 1

    # print(f"DEBUG: processed_text_pairs = \n {processed_text_pairs}")

    return processed_text_pairs
```

Функция биграммного шифрования PlayFair

- Если две буквы находятся в одной строке, возьмите правую букву (если это самая правая буква, возьмите самую левую букву строки).
- Если две буквы находятся в одном столбце, возьмите нижнюю букву (если это нижняя буква, возьмите верхнюю букву столбца).
- Если две буквы не находятся в одной строке и столбце, возьмите диагональную букву прямоугольника, где расположены две буквы.

```python
# encrypt process
def encrypt(text, key_matrix):
    encrypted_text_pairs = []
    preprocessed_text_pairs = preprocess_text(text)  # preprocess plaint text

    for preprocessed_text_pair in preprocessed_text_pairs:
        preprocessed_pair_letter = get_letter_from_pair(preprocessed_text_pair)
        row1, col1 = find_position(key_matrix, preprocessed_pair_letter[0])
        row2, col2 = find_position(key_matrix, preprocessed_pair_letter[1])

        encrypted_pair_letter = ['', '']
        if row1 == row2:  # if same row
            encrypted_pair_letter[0] = key_matrix[row1][(col1 + 1) % 6]
            encrypted_pair_letter[1] = key_matrix[row2][(col2 + 1) % 6]
        elif col1 == col2:  # if same col
            encrypted_pair_letter[0] = key_matrix[(row1 + 1) % 6][col1]
            encrypted_pair_letter[1] = key_matrix[(row2 + 1) % 6][col2]
        else:  # if square
            encrypted_pair_letter[0] = key_matrix[row1][col2]
            encrypted_pair_letter[1] = key_matrix[row2][col1]

        # make the capitalization the same as plaint text
        if preprocessed_pair_letter[0].islower():
            encrypted_pair_letter[0] = encrypted_pair_letter[0].lower()
        if preprocessed_pair_letter[1].islower():
            encrypted_pair_letter[1] = encrypted_pair_letter[1].lower()

        # set encrypted text pair matrix
        encrypted_text_pair = []
        letter_count = 0
        for letter in preprocessed_text_pair:
            if is_letter_in_alphabet(letter):
                encrypted_text_pair.append(encrypted_pair_letter[letter_count])
                letter_count += 1
            else:
                encrypted_text_pair.append(letter)
        encrypted_text_pairs.append(encrypted_text_pair)

    return encrypted_text_pairs
```

Функция расшифровки больших букв Playfair

Обратный процесс функции шифрования

```
# decrypt process
def decrypt(encrypted_text_pairs, key_matrix):
    decrypted_text_pairs = []

    for encrypted_text_pair in encrypted_text_pairs:
        encrypted_pair_letter = get_letter_from_pair(encrypted_text_pair)
        row1, col1 = find_position(key_matrix, encrypted_pair_letter[0])
        row2, col2 = find_position(key_matrix, encrypted_pair_letter[1])

        decrypted_pair_letter = ['', '']
        if row1 == row2:  # if same row
            decrypted_pair_letter[0] = key_matrix[row1][(col1 - 1) % 6]
            decrypted_pair_letter[1] = key_matrix[row2][(col2 - 1) % 6]
        elif col1 == col2:  # if same col
            decrypted_pair_letter[0] = key_matrix[(row1 - 1) % 6][col1]
            decrypted_pair_letter[1] = key_matrix[(row2 - 1) % 6][col2]
        else:  # if square
            decrypted_pair_letter[0] = key_matrix[row1][col2]
            decrypted_pair_letter[1] = key_matrix[row2][col1]

        if encrypted_pair_letter[0].islower():
            decrypted_pair_letter[0] = decrypted_pair_letter[0].lower()
        if encrypted_pair_letter[1].islower():
            decrypted_pair_letter[1] = decrypted_pair_letter[1].lower()

        decrypted_text_pair = []
        letter_count = 0
        for letter in encrypted_text_pair:
            if is_letter_in_alphabet(letter):
                decrypted_text_pair.append(decrypted_pair_letter[letter_count])
                letter_count += 1
            else:
                decrypted_text_pair.append(letter)
        decrypted_text_pairs.append(decrypted_text_pair)

    decrypted_text_pairs = remove_x(decrypted_text_pairs)

    return decrypted_text_pairs
```

Удалить лишние буквы, добавленные во время предварительной обработки

```
# function for removing additional 'A' in result during preprocess and encryption
def remove_x(decrypted_text_pairs):
    res_text_pairs = []
    for i in range(len(decrypted_text_pairs) - 1):
        text_pair = decrypted_text_pairs[i]
        text_pair_next = decrypted_text_pairs[i + 1]
        pair_letter = get_letter_from_pair(text_pair)
        pair_letter_next = get_letter_from_pair(text_pair_next)

        if pair_letter[0] == pair_letter_next[0] and pair_letter[1] == 'A':
            text_pair.remove('A')

        res_text_pairs.append(text_pair)

    if decrypted_text_pairs[-1][-1] == 'A':
        decrypted_text_pairs[-1].remove('A')
        res_text_pairs.append(decrypted_text_pairs[-1])

    return res_text_pairs
```

Функция для чтения и записи файлов

```
# read from file
def read_file(filename):
    with open(filename, 'r', encoding='UTF-8') as file:
        return file.read()


# write into file
def write_file(filename, content):
    with open(filename, 'w', encoding='UTF-8') as file:
        file.write(content)
```

Функция для сравнения исходного текста и расшифрованныого текста

```
def compare_strings(str1, str2):
    # make sure length of str1 and str2 are the same
    if len(str1) != len(str2):
        raise ValueError("String lengths do not match")

    # compare and count different letters in str1 and str2
    diff_count = 0
    for char1, char2 in zip(str1, str2):
        if char1 != char2:
            diff_count += 1

    return diff_count
```

Основная функция

```
# main function
if __name__ == "__main__":
    key_input = input("Please input keyword: ")
    # key_input = "SECRETKEY"
    matrix = generate_key_matrix(key_input)

    print("Generated key matrix: ")
    print_matrix(matrix)

    # 从read plain text from file
    plaintext = read_file('plaintext_en.txt')
    print(f"Plain text: \n{plaintext}")

    # encrypt text
    text_pairs_encrypted = encrypt(plaintext, matrix)
    write_file('encrypted_en.txt', text_pairs_to_string(text_pairs_encrypted))

    print(f"Encrypted text: \n{text_pairs_to_string(text_pairs_encrypted)}")

    # decrypt text
    text_pairs_decrypted = decrypt(text_pairs_encrypted, matrix)
    write_file('decrypted_en.txt', text_pairs_to_string(text_pairs_decrypted))

    print(f"Decrypted text: \n{text_pairs_to_string(text_pairs_decrypted)}")

    diff_texts = compare_strings(plaintext, text_pairs_to_string(text_pairs_decrypted))
    print(f"Different between text: \n{diff_texts}")
```

Результаты работы программы

Пример Usage

```
/usr/bin/python3 /Users/2398768715qq.com/GitHub/ITMO-Labs/InfoSecurity/Lab1Playfair/InfoSe
Please input keyword: КЛЮЧ
Generated key matrix:
['К', 'Л', 'Ю', 'Ч', 'А', 'Б']
['В', 'Г', 'Д', 'Е', 'Ё', 'Ж']
['З', 'И', 'Й', 'М', 'Н', 'О']
['П', 'Р', 'С', 'Т', 'У', 'Ф']
['Х', 'Ц', 'Ш', 'Щ', 'Ъ', 'Ы']
['Ь', 'Э', 'Я', 'А', 'В', 'С']

Plain text:
В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

Encrypted text:
Г пчнлбъ зюъдм улжффщ ощ зфтгеёгргр рюёесаътч ймщйжибижмлдтапч ккнф:
Туёабюб ощ рсзжгчм ръчщмкэоън юуёгрй птъдтпесацмц пдщёмйм фз гйнкмор нмугттжсшл. Вфм звюча
Уё йфозгё тгнпкэучфмё куёгрнк ощ улипбкмфбюм ржмтгрмемпюса нзеёкэ. Арч нйжгчк злючабюб люч
Т рзнмыяк улипбкмфёуозм нйжгчй нх фцртуфрргз л ьлтргтйнёмучкэознт рсмжчпрцзжёуйл. Жсю рзиё
Нключамргчbзщо аркузн жсби щмтурцзжёумг фййвёуозжи рсмфмфзрч. Нх ффигёгр тдцрч сдтфмд п та

Decrypted text:
В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

Different between text:
0

Process finished with exit code 0
```

Вывод

В ходу лабораторной работы, я выучил несколько подходы для щифрования и дешифрования текст в файле.