Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Технологии и методы программирования»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«***Реализация исторических алгоритмов шифрования***»

Выполнил:

Чу Ван Доан – Студент группы N3347

(подпись)

Проверил:

Ищенко Алексей Петрович

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

Содержание

1. Задание

Реализация исторических алгоритмов шифрования

Написать программу, реализующую два исторических примера алгоритмов шифрования. Продумать интерфейс, руководство пользователя и описание работы алгоритмов (их историей и криптоустойчивостью), для демонстрации алгоритмов пользователю со встроенными примерами текстов (шифруем и дешифруем), а также с возможностью ввода произвольного текста с его шифровкой дешифровкой.

Вариант получается из номера N студента по списку группы по следующей формуле (N%10)+1.

Необходимо реализовать следующие этапы для каждого своего алгоритма:

1. Краткое описание алгоритма шифрования.
2. Определение ограничений на решаемую задачу.
3. Реализация данного алгоритма в виде программного кода.
4. Реализация и описание графического интерфейса.
5. Оформление отчета (включает описание выше и нижеизложенных пунктов со скриншотами работы программы и результатов)

Для защиты программы необходимо предложить несколько (3-4) различных текстов, в которых используется различное количество символов, например: может быть дан текст в котором встречаются много раз только три-четыре буквы, или текст где все буквы сообщения различные, а также вариативность текстов может быть русско-английская.

Варианты (подробные описания шифрования ищите в сети Интернет):

1. Вариант: 1 и 11
2. Вариант: 2 и 12
3. Вариант: 3 и 13
4. Вариант: 4 и 14
5. Вариант: 5 и 15
6. Вариант: 6 и 16
7. Вариант: 7 и 17
8. Вариант: 8 и 18
9. Вариант: 9 и 19
10. Вариант: 10 и 20

Соответствие номеров алгоритмам шифрования:

1. Шифр Цезаря
2. Шифровка последовательностей нулей и единиц
3. «Табличная шифровка»
4. «Матричная шифровка»
5. «Шифровка решеткой»
6. «Шифровка зафиксированной перестановкой»
7. Шифр Гронсфельда
8. Шифровка с помощью квадрата Полибия

9. Шифр Хилла (с длиной блока = 2)
10. Шифр Атбаш
11. Шифр Вижинера (для латинских букв)
12. Шифр Вижинера (для русских букв)
13. Шифр Плейфера
14. Шифр с использованием кодового слова
15. Шифр перестановки "скитала"
16. Простая табличная перестановка
17. Табличная шифровка с ключевым словом
18. Двойная табличная перестановка
19. Шифровка с помощью магического квадрата
20. Шифровка «тарабарская грамота» (весь алфавит)

Для каждого алгоритма шифрования необходимо написать программу (можно написать одну с меню-выбором из двух заданных алгоритмов), которая работает в двух режимах: шифрования и дешифрования. При любом из режимов программа считывает исходный текст из окна или файла, шифрует или дешифрует его и записывает или отображает в окно или в другой файл.

2. Ход работы

2.1. Выбор варианта.

Мой порядковый номер в списке группы N3347 — 23, поэтому (23 % 10) + 1 = 4.
Мои варианты — 4: «Матричная шифровка» и 14: «Шифр с использованием кодового слова»

2.2. Матричная шифровка

Матричная шифровка - это метод шифрования в криптографии, основанный на математике матриц, который преобразует исходный текст (plaintext) в зашифрованный текст (ciphertext). Этот метод использует умножение матриц и применяет их в качестве ключа шифрования. Это эффективная и популярная техника, применяемая как в классической, так и в современной криптографии.

2.2.1. Основные понятия

- Исходный текст (Plaintext): Текст, который необходимо зашифровать, обычно представленный в виде символов или чисел.
- Ключ-матрица (Key Matrix): Квадратная матрица (размера n × n), содержащая целые числа. Это основной элемент, определяющий безопасность шифрования.
- Зашифрованный текст (Ciphertext): Результат шифрования после применения умножения матриц.
- Операция по модулю (Modulo Operation): После шифрования результат обычно вычисляется по модулю (например, по модулю 26 для английского алфавита), чтобы значения шифра оставались в допустимом диапазоне символов.

2.2.2. Шаги матричного шифрования

- Подготовка данных:
    + Преобразование текста в числа на основе таблицы кодирования (например, 'A' = 0, 'B' = 1, ..., 'Z' = 25)
    + Разделение текста на блоки длиной, соответствующей размеру ключевой матрицы n.
- Выбор ключа-матрицы:

+ Выбирается квадратная матрица $n \times n$ n×n в качестве ключа. Матрица должна иметь обратную по модулю (то есть определитель матрицы не должен быть равен 0 по модулю).

- Шифрование:
  + Каждый блок исходного текста представляется в виде вектор-столбца.
  + Выполняется умножение ключа-матрицы на вектор исходного текста:

$$C = K * P \mod m$$

    - C: Вектор зашифрованного текста.
    - K: Ключ-матрица (квадратная матрица).
    - P: Вектор исходного текста.
    - m: Значение модуля.
- Дешифрование:
  + Для дешифрования используется обратная матрица ключа по модулю: K^{-1}
  + Формула дешифрования:

$$P = K^{-1} * C \mod m$$

- Вычисление по модулю: Все результаты умножения и обращения должны быть вычислены по модулю m, чтобы гарантировать корректность результатов.

2.2.3. Преимущества и недостатки:

- Преимущества: Легко реализуется с использованием матриц. Обеспечивает высокий уровень безопасности при правильном выборе ключа.
- Недостатки: Требует ключа, который должен быть обратимым по модулю, что усложняет его выбор. Сложность вычислений возрастает при больших данных.

2.2.4. Применение:

- В классической криптографии: Шифр Хилла (Hill Cipher) – пример матричного шифрования.
- В современной криптографии: Операции с матрицами используются в системах, таких как AES (основано на конечных полях Galois Field).

2.3. Шифрование с использованием кодового слова

Шифрование с кодовым словом – это метод криптографии, который использует ключевое слово для создания шифра. Это один из простейших и понятных способов

шифрования текста, где ключевое слово определяет, как текст преобразуется в зашифрованный.

2.3.1. Основные понятия

- Исходный текст (Plaintext): Это текст, который необходимо зашифровать.
- Ключевое слово (Keyword): Это слово, выбранное для создания шифра. Оно используется для формирования алфавита или правила шифрования.
- Шифрованный текст (Ciphertext): Это результат преобразования исходного текста с использованием кодового слова.
- Алфавит: Используется стандартный алфавит (например, латиница, кириллица), на основе которого формируется новая последовательность символов.

2.3.2. Методы шифрования с кодовым словом

2.3.2.1. Шифр с перестановкой алфавита:

- Кодовое слово используется для создания нового алфавита путем перестановки букв:
    + Алфавит: АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ.
    + Кодовое слово: "КОД".
    + Новый алфавит создается так: сначала записываются буквы кодового слова (без повторений), затем остальные буквы алфавита, не входящие в кодовое слово:

      КОДАБВГЕЖЗИКЛМНПРСТУФХЦЧШЩЪЫЬЭЮЯКОДАБВГЕЖЗИКЛМНПРСТУФХЦЧШЩЪЫЬЭЮЯКОДАБВГЕЖЗИКЛМНПРСТУФХЦЧШЩЪЫЬЭЮЯ
- Шифрование выполняется путем замены каждой буквы исходного текста на букву из нового алфавита.

2.3.2.2. Шифр Виженера (Vigenère Cipher)

- Кодовое слово повторяется столько раз, сколько нужно для покрытия всего текста.
- Каждая буква исходного текста сдвигается по алфавиту на значение, соответствующее букве кодового слова.
    + Исходный текст: "ПРИМЕР".
    + Кодовое слово: "КЛЮЧ".
    + Повторяем кодовое слово до длины текста: "КЛЮЧКЛ".
    + Сдвиг:

- "П" (16-я буква) + "К" (11-я буква) = "Ы" (27 % 33 = 27).
- Таким образом, "ПРИМЕР" → "ЫСЛЬИК".

2.3.2.3. Шифр Цезаря с кодовым словом

- Это улучшенная версия классического шифра Цезаря.
- Кодовое слово используется для создания нового алфавита, как в первом методе, а затем все буквы текста сдвигаются на фиксированное количество позиций.

2.3.3. Шаги шифрования с кодовым словом

- Выбор кодового слова. Кодовое слово должно быть уникальным и содержать не менее 3 символов.
- Создание шифровального алфавита.
  + Исключаются повторяющиеся буквы в ключе.
  + Формируется последовательность символов на основе ключа и оставшихся букв алфавита.
- Шифрование текста. Каждая буква исходного текста заменяется соответствующей буквой из нового алфавита.
- Дешифрование текста. Используется обратный процесс: каждая буква зашифрованного текста заменяется на соответствующую букву исходного алфавита.

2.3.4. Преимущества и недостатки

- Преимущества:
  + Простота реализации.
  + Легко адаптируется под любой алфавит.
  + Возможность использования уникального ключа для каждого сообщения.
- Недостатки:
  + Уязвимость к анализу частотности, если кодовое слово короткое.
  + Ограниченная стойкость при использовании стандартного алфавита.

2.3.5. Применение

- Классическая криптография: Шифры с перестановкой алфавита.
- Современные применения: Подходит для создания простых шифровальных систем в учебных и демонстрационных целях.

2.4. Выполнение программы.

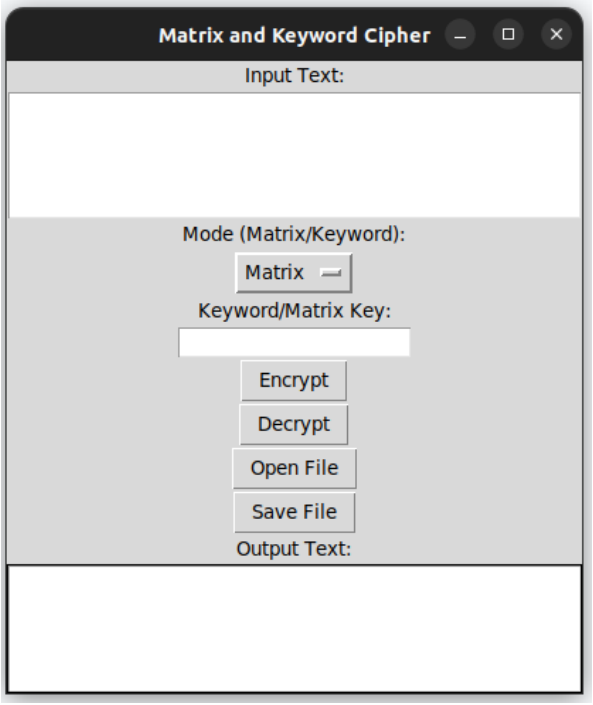

Рисунок 1 - Интерфейс программы

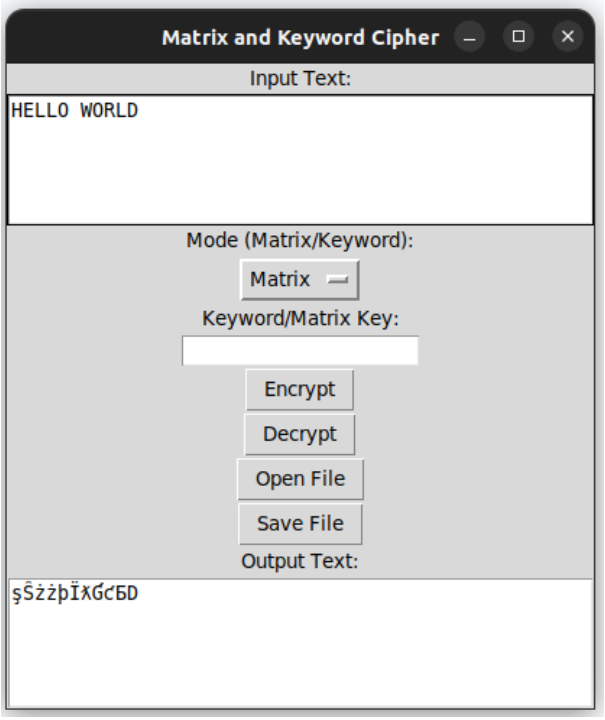

Рисунок 2 - Шифрование типа "Матричное шифрование"

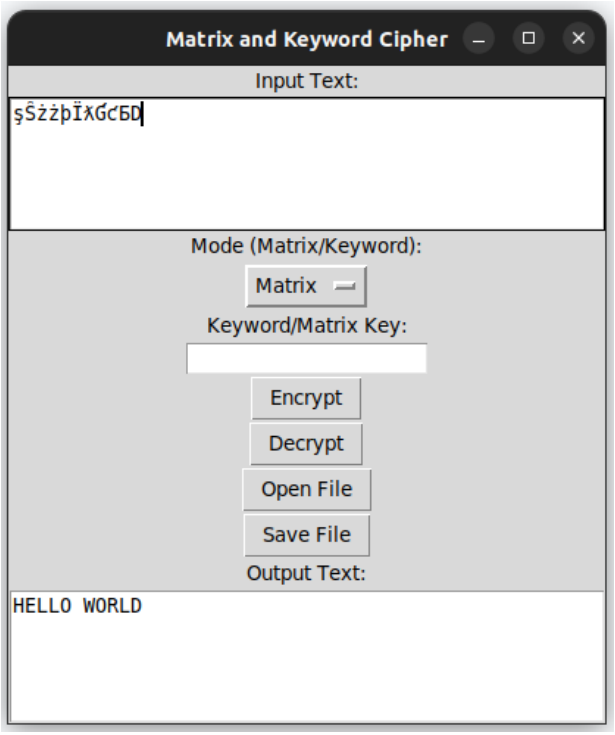

Рисунок 3 - Дешифрование типа "Матричное шифрование"

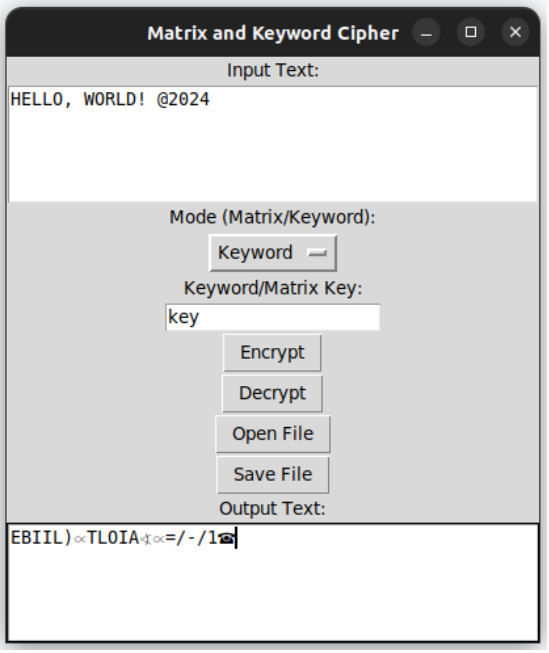

Рисунок 4 - Шифрование типа "С использованием кодового слова"

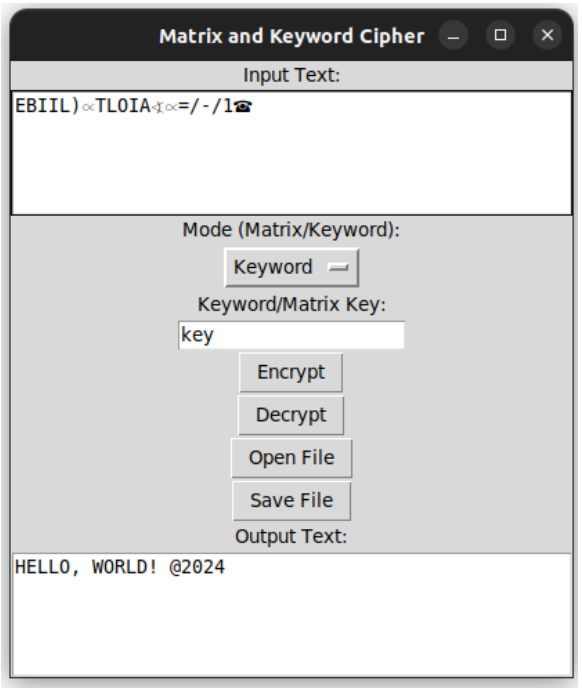

Рисунок 5 - Дешифрование типа "С использованием кодового слова".

3. Исходный код

```
import tkinter as tk
from tkinter import filedialog
import numpy as np

def matrix_encrypt(text, key_matrix):
    text_vector = [ord(char) for char in text]
    while len(text_vector) % len(key_matrix) != 0:
        text_vector.append(0)

    text_matrix = np.array(text_vector).reshape(-1, len(key_matrix)).T
    encrypted_matrix = np.dot(key_matrix, text_matrix) % 1114112
            encrypted_text = ''.join(chr(num) for num in
encrypted_matrix.T.flatten())
    return encrypted_text

def matrix_decrypt(text, key_matrix):
    text_vector = [ord(char) for char in text]
    text_matrix = np.array(text_vector).reshape(-1, len(key_matrix)).T

    det = int(round(np.linalg.det(key_matrix)))
    if det == 0:
        return "Error: Key matrix is not invertible!"
```

```
    det_mod_inverse = pow(det, -1, 1114112)
    adjugate_matrix = np.round(det * np.linalg.inv(key_matrix)).astype(int)
% 1114112
    inverse_matrix = (det_mod_inverse * adjugate_matrix) % 1114112

    decrypted_matrix = np.dot(inverse_matrix, text_matrix) % 1114112
            decrypted_text    =    ''.join(chr(num)    for    num    in
decrypted_matrix.T.flatten() if num > 0)
    return decrypted_text

# Mã hóa bằng từ khóa (Keyword Cipher) hỗ trợ Unicode
def keyword_encrypt(text, keyword):
    all_chars = ''.join(chr(i) for i in range(1114112))
    keyword_unique = ''.join(sorted(set(keyword), key=keyword.index))
     cipher_chars = keyword_unique + ''.join(c for c in all_chars if c not
in keyword_unique)

         cipher_map    =    {all_chars[i]:    cipher_chars[i]    for    i    in
range(len(all_chars))}
    try:
        encrypted_text = ''.join(cipher_map[char] for char in text)
    except KeyError:
        return "Error: Invalid character in input text!"
    return encrypted_text

def keyword_decrypt(text, keyword):
    all_chars = ''.join(chr(i) for i in range(1114112))
    keyword_unique = ''.join(sorted(set(keyword), key=keyword.index))
     cipher_chars = keyword_unique + ''.join(c for c in all_chars if c not
in keyword_unique)

        reverse_cipher_map   =   {cipher_chars[i]:   all_chars[i]   for   i   in
range(len(all_chars))}
    try:
        decrypted_text = ''.join(reverse_cipher_map[char] for char in text)
    except KeyError:
        return "Error: Invalid character in input text!"
    return decrypted_text

def encrypt():
    text = input_text.get("1.0", "end-1c")
    mode = mode_var.get()
    if mode == "Matrix":
        key = np.array([[2, 3], [1, 4]])
        result = matrix_encrypt(text, key)
```

```
    elif mode == "Keyword":
        keyword = shift_entry.get()
        if not keyword:
            result = "Error: Keyword cannot be empty!"
        else:
            result = keyword_encrypt(text, keyword)
    else:
        result = "Invalid mode!"
    output_text.delete("1.0", "end")
    output_text.insert("1.0", result)

def decrypt():
    text = input_text.get("1.0", "end-1c")
    mode = mode_var.get()
    if mode == "Matrix":
        key = np.array([[2, 3], [1, 4]])
        result = matrix_decrypt(text, key)
    elif mode == "Keyword":
        keyword = shift_entry.get()
        if not keyword:
            result = "Error: Keyword cannot be empty!"
        else:
            result = keyword_decrypt(text, keyword)
    else:
        result = "Invalid mode!"
    output_text.delete("1.0", "end")
    output_text.insert("1.0", result)

def open_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        with open(file_path, "r", encoding="utf-8") as file:
            input_text.delete("1.0", "end")
            input_text.insert("1.0", file.read())

def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt")
    if file_path:
        with open(file_path, "w", encoding="utf-8") as file:
            file.write(output_text.get("1.0", "end-1c"))

window = tk.Tk()
window.title("Matrix and Keyword Cipher")

tk.Label(window, text="Input Text:").pack()
```

```
input_text = tk.Text(window, height=5, width=50)
input_text.pack()

tk.Label(window, text="Mode (Matrix/Keyword):").pack()
mode_var = tk.StringVar(value="Matrix")
mode_menu = tk.OptionMenu(window, mode_var, "Matrix", "Keyword")
mode_menu.pack()

tk.Label(window, text="Keyword/Matrix Key:").pack()
shift_entry = tk.Entry(window)
shift_entry.pack()

tk.Button(window, text="Encrypt", command=encrypt).pack()
tk.Button(window, text="Decrypt", command=decrypt).pack()
tk.Button(window, text="Open File", command=open_file).pack()
tk.Button(window, text="Save File", command=save_file).pack()

tk.Label(window, text="Output Text:").pack()
output_text = tk.Text(window, height=5, width=50)
output_text.pack()

window.mainloop()
```

Заключение

В данной работе реализованы два типа шифрования: матричное шифрование и шифрование с использованием кодового слова.

- Матричное шифрование основано на линейной алгебре, обеспечивает высокую стойкость при правильном выборе ключа и поддерживает символы Unicode, но требует вычислений обратной матрицы.
- Шифрование с кодовым словом простое и эффективно для базовых задач, однако менее стойкое и уязвимо для частотного анализа.

Созданный графический интерфейс на tkinter позволяет пользователям шифровать и дешифровать текст, выбирать тип шифрования и работать с файлами. Реализация алгоритмов универсальна и поддерживает полный набор символов Unicode.