Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 1

Простые симметричные шифры
Тема

Преподаватель _____ М. М. Кучеров
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цели

1. Ознакомиться с основами симметричного шифрования;
2. Ознакомиться с простыми симметричными криптографическими шифрами на основе методов подстановок, перестановок и гаммирования;
3. Освоить основные этапы проектирования и реализации симметричных шифров;

2 Квадрат Полибия

Полибий - греческий историк, полководец и государственный деятель, живший в III веке до н.э. Он предложил оригинальный код простой замены, который стал известен как «квадрат Полибия» (англ. Polybius square) или шахматная доска Полибия. Данный вид кодирования изначально применялся для греческого алфавита, но затем был распространен на другие языки. Буквы алфавита вписываются в квадрат или подходящий прямоугольник. Если букв для квадрата больше, то их можно объединять в одной ячейке.

Такую таблицу можно использовать как в шифре Цезаря. Для шифрования на квадрате находим букву текста и вставляем в шифровку нижнюю от неё в том же столбце. Если буква в нижней строке, то берём верхнюю из того же столбца.

3 Ход работы

3.1 Блок-схема

На рисунке 1 представлена блок-схема с реализацией шифрования произвольного сообщения с использованием квадрата Полибия.

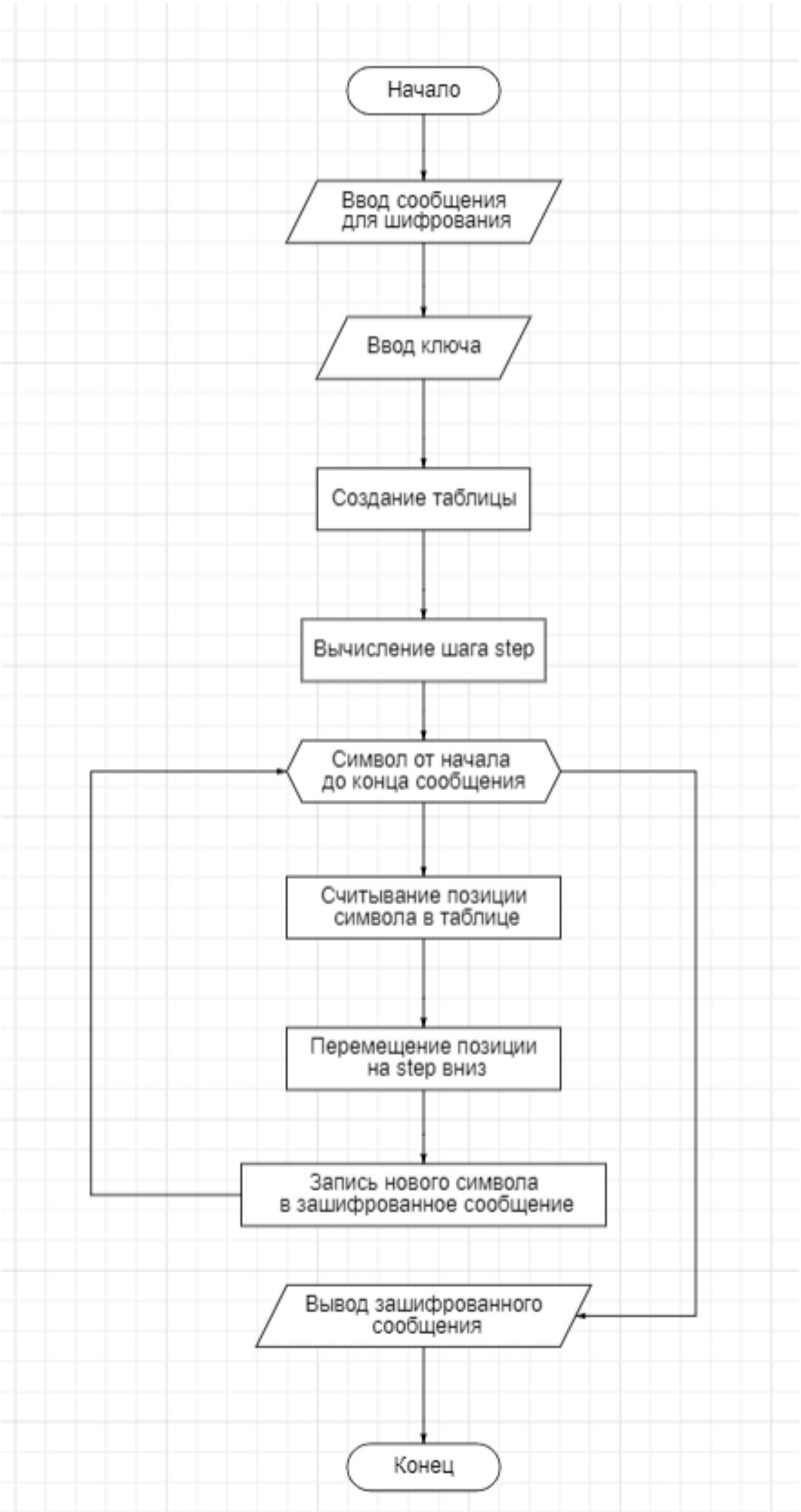

Рисунок 1 – Блок-схема шифрования по квадрату Полибия

3.2 Реализация программы

На листинге 1 представлен программный код с реализацией алгоритма на языке Python.

Листинг 1 – Квадрат Полибия

```
import numpy as np
import random


ALPHABET = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',
'н', 'о', 'п', 'р', 'с', 'т',
            'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я', '.',
',', ' ']


def create_cipher_table(key=None):
    random.seed(key)
    alphabet = ALPHABET.copy()
    random.shuffle(alphabet)

    table = np.array(alphabet).reshape(6, 6)
    return table


def encrypt(message: str, key=None, decrypt=False, print_random_info=False) ->
str:
    message = message.lower()
    cipher_table = create_cipher_table(key)
    rows_num, cols_num = cipher_table.shape

    random.seed(key)
    step = random.randint(0, rows_num - 1)

    if print_random_info:
        print(cipher_table)
        print('Шаг:', step)
    if decrypt:
        step *= -1

    encrypted_message = ''
    for char in message:
```

Окончание листинга 1

```
            if np.where(cipher_table == char):
                encrypted_message += char
                continue
            y_coord = np.where(cipher_table == char)[0][0]
            x_coord = np.where(cipher_table == char)[1][0]
            encrypted_message += cipher_table[(y_coord + step) % rows_num][x_coord]

    return encrypted_message


def main():
    message = input("Введите сообщение:\n")
    key = input("Введите ключ:\n")
    encrypted_message = encrypt(message, key=key, print_random_info=True)
    decrypted_message = encrypt(encrypted_message, key=key, decrypt=True)

    print('Исходное сообщение:', message)
    print('Зашифрованное сообщение:', encrypted_message)
    print('Расшифрованное сообщение:', decrypted_message)

if __name__ == '__main__':
    main()
```

3.3 Модификация алгоритма

Для того чтобы повысить криптоустойчивость алгоритма, было принято решение использовать дополнительное значение для шифрования – ключ. В зависимости от значения ключа в случайном порядке определяется позиция каждого символа алфавита в таблице и случайный шаг, представляющий из себя количество строк, которые необходимо пропустить, чтобы добраться до буквы символа для шифрования.

Данные шаги позволили сделать метод прямого перебора решения практически невозможным, если неизвестен ключ.

3.4 Примеры работы алгоритма шифрования

На рисунках 2-4 представлены примеры работы алгоритма шифрования

```
Введите сообщение:
От топота копыт пыль по полю летит
Введите ключ:
ключ
[['х' 'г' 'ж' 'е' 'ч' 'в']
 ['ы' 'т' 'у' 'ъ' 'и' 'н']
 ['б' 'щ' ' ' '.' 'й' ',']
 ['д' 'ш' 'ь' 'о' 'с' 'ц']
 ['э' 'ф' 'я' 'а' 'ё' 'з']
 ['п' 'ю' 'л' 'р' 'к' 'м']]
Шаг: 1
Исходное сообщение: От топота копыт пыль по полю летит
Зашифрованное сообщение: ащьщахащрьчахбщьхбжяьхаьхажгьжъщйщ
Расшифрованное сообщение: от топота копыт пыль по полю летит
```

Рисунок 2 – Первый пример

```
Введите сообщение:
Береги сапоги снову, а честь смолоду.
Введите ключ:
Бульба
[['а' 'й' 'ш' 'г' 'и' 'о']
 ['ч' 'п' '.' 'б' 'н' 'э']
 ['я' 'ж' 'к' 'ь' 'у' 'е']
 [' ' 'ъ' 'л' 'з' 'м' 'ы']
 ['р' 'х' 'д' 'ё' 'ц' 'ю']
 ['щ' 'т' ',' 'в' 'с' 'ф']]
Шаг: 2
Исходное сообщение: Береги сапоги снову, а честь смолоду.
Зашифрованное сообщение: зюаюьущняъеьущнмебц.щящ юнпёщнсе,ешцл
Расшифрованное сообщение: береги сапоги снову, а честь смолоду.
```

Рисунок 3 – Второй пример

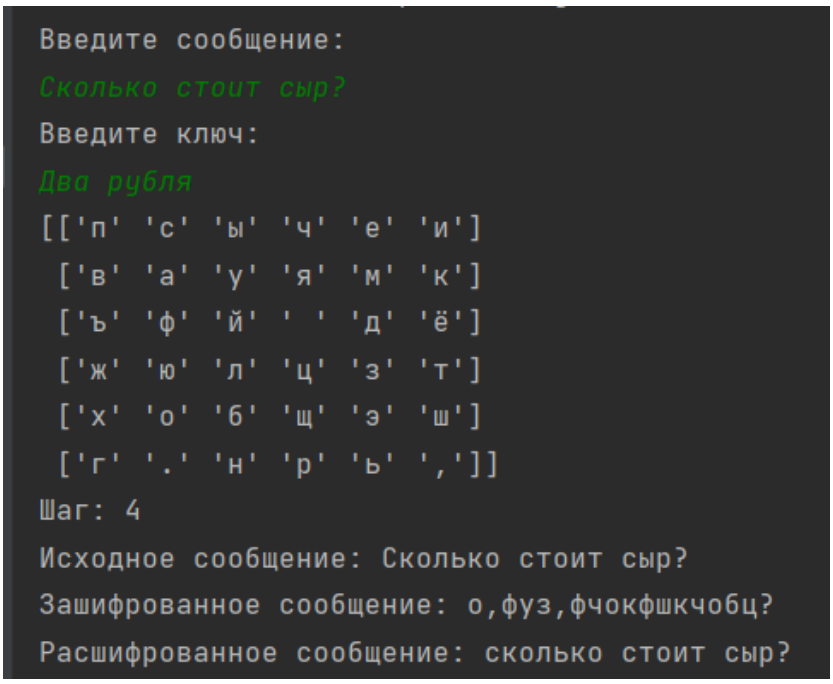

Рисунок 3 – Третий пример

4 Вывод

В данной работе был изучен алгоритм шифрования «квадрат Полибия». Этот алгоритм был реализован и модифицирован на языке Python с реализацией консольного интерфейса пользователя.