МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 НА ТЕМУ:**

**Исследование криптографических шифров на основе перестановки символов**

Выполнила студентка 3 курса 4 группы

Сятковской Екатерины

Минск 2023

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров.

**Теоретические сведения**

Криптоанализ – это раздел криптологии, занимающийся методами взлома шифров или методами организации криптографических атак на шифры.

Сущность перестановочного шифрования состоит в том, что исходный текст (*М*) и зашифрованный текст (*С*) основаны на использовании одного и того же алфавита, а тайной или ключевой информацией является алгоритм перестановки.

Шифры перестановки относятся к классу симметричных. Элементами текста могут быть отдельные символы (самый распространенный случай), пары, тройки букв и т.д.

В классической криптографии шифры перестановки делятся на два подкласса:

* шифры простой, или одинарной, перестановки – при зашифровании символы открытого текста *Мi* перемещаются с исходных позиций в новые (в шифртексте *Сi*) один раз;
* шифры сложной, или множественной, перестановки – при зашифровании символы открытого текста *Мi* перемещаются с исходных позиций в новые (в шифртексте *Сi*) несколько раз.

Среди шифров рассматриваемого подкласса иногда выделяют ***шифры простой перестановки (или перестановки без ключа)***. Символы открытого текста *Мi* перемешиваются по каким-либо правилам. Формально каждое из таких правил может рассматриваться в качестве ключа.

***Шифры простой блочной перестановки*** строятся по тем же правилам, что и шифры простой перестановки. Блок должен состоять из 2 или более символов. Если общее число таких символов в сообщении не кратно длине сообщения, то последний блок можно дополнить произвольными знаками.

Основой современных ***шифров маршрутной перестановки*** является геометрическая фигура, обычно прямоугольник или прямоугольная матрица. В ячейки этой фигуры по определенному маршруту (слева направо, сверху вниз или каким-либо иным образом) записывается открытый текст. Для получения шифрограммы нужно записать символы этого сообщения в иной последовательности, т. е. по иному маршруту.

***Шифр вертикальной перестановки*** является разновидностью шифра маршрутной перестановки. К особенностям вертикального шифра можно отнести следующие:

* количество столбцов в таблице фиксируется и определяется длиной ключа;
* маршрут вписывания: слева направо, сверху вниз;
* шифрограмма выписывается по столбцам в соответствии с их нумерацией (ключом).

Ключ может задаваться в виде текста (слова или словосочетания). Лексикографическое местоположение символов в ключевом выражении определяет порядок считывания столбцов.

Особенностью ***шифров множественной перестановки*** является минимум двукратная перестановка символов шифруемого сообщения. В простейшем случае это может задаваться перемешиванием не только столбцов, но и строк. Таким образом, этот случай соответствует использованию двух основных ключей: длина одного из них равна числу столбцов, другого – числу строк. К ключевой информацию мы можем относить также способы вписывания сообщения и считывания отдельных символов из текущего столбца матрицы.

**Задание 1:** Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции:

* выполнять зашифрование/расшифрование текстовых документов (объемом не менее 500 знаков), созданных на основе алфавита языка в соответствии с нижеследующей таблицей вариантов задания; при этом следует использовать шифры подстановки из третьего столбца данной таблицы;

|  |  |  |
| --- | --- | --- |
| Вариант | Алфавит | Шифр |
| 2 | Русский | 1. Маршрутная перестановка (маршрут –по спирали; параметры таблицы – по указанию преподавателя)  2. Множественная перестановка, ключевые  слова – собственные имя и фамилия |

Смысл маршрутной перестановки заключается в том, что сообщение записывается в виде таблицы определенного размера и шифрование происходит по определенному маршруту. При шифровании зигзагом символы считываются по следующему маршруту:

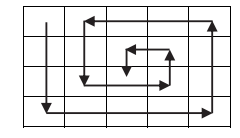


Рисунок 1 – Маршрутная перестановка по спирали

Например, для сообщения «Привет волшебный лес», таблица будет выглядеть следующим образом:



Рисунок 2 – Таблица 4 × 5 для исходного сообщения

Считывается текст по приведенному выше узору:

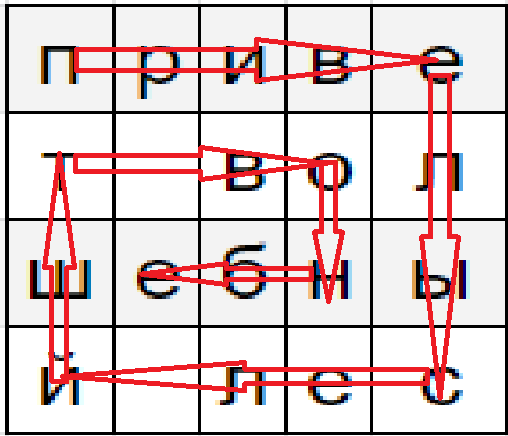


Рисунок 3 – Муршрутная перестановка зигзагом для сообщения

Получаем следующий текст «привелысел йшт вонбе».

Для того чтобы расшифровать данный текст необходимо провести обратные операции – записать текст по маршруту зигзаг и прочитать и его построчно.

Ниже приведен код основных методов программы, реализующей алгоритм, на языке Python:

import numpy as np  
from datetime import datetime  
  
def form\_matrix(message : str**,** route\_step : int) -> np.ndarray:  
 message\_len = len(message)  
 if message\_len % route\_step != **0**:  
 message += ' ' \* (route\_step - message\_len % route\_step)  
 matrix = np.array(list(message)**,** dtype=np.str\_)  
 matrix = matrix.reshape(route\_step**,** -**1**)  
 return matrix  
  
def encrypt(message : str**,** route\_step : int) -> str:  
 message = message.replace('ё'**,** 'е').replace('Ё'**,** 'Е') # замена ё на е  
 matrix = form\_matrix(message**,** route\_step)  
 print('Таблица:')  
 for row in matrix:  
 print(row)  
 res = ''  
 i**,** j = matrix.shape  
 for r in range(**0,** (min(i**,** j) + **1**) // **2**):  
 for c in range(r**,** j - r):  
 res += matrix[r**,** c]  
 for c in range(r + **1,** i - r):  
 res += matrix[c**,** j - r - **1**]  
 for c in range(j - r - **2,** r - **1,** -**1**):  
 res += matrix[i - r - **1,** c]  
 for c in range(i - r - **2,** r**,** -**1**):  
 res += matrix[c**,** r]  
 return res  
  
def decrypt(message: str**,** route\_step : int) -> str:  
 message = message.replace('ё'**,** 'е').replace('Ё'**,** 'Е') # замена ё на е  
 message\_len = len(message)  
 if message\_len % route\_step != **0**:  
 message += ' ' \* (route\_step - message\_len % route\_step)  
 matrix = np.empty((route\_step**,** message\_len // route\_step)**,** dtype=np.str\_)  
 i**,** j = matrix.shape  
 r\_count = **0** r\_start = **0** r\_end = i  
 c\_count = **0** c\_start = **0** c\_end = j  
 for letter in message:  
 matrix[r\_count**,** c\_count] = letter  
 if c\_count < c\_end - **1** and r\_count == r\_start:  
 c\_count += **1** elif r\_count < r\_end - **1** and c\_count == c\_end - **1**:  
 r\_count += **1** elif c\_count > c\_start and r\_count == r\_end - **1**:  
 c\_count -= **1** elif r\_count > r\_start and c\_count == c\_start:  
 r\_count -= **1** if r\_count == r\_start + **1**:  
 r\_start += **1** r\_end -= **1** c\_start += **1** c\_end -= **1** print('\nТаблица:')  
 for row in matrix:  
 print(row)  
 res = ''.join(matrix.ravel())  
 return res  
  
#словарь {символ:количество потворений этого символа} отсортированный по ключу  
def get\_letters\_amount(seq):  
 letters\_dictionary = {}  
 for i in seq:  
 if i.isalpha():  
 if i not in letters\_dictionary:  
 letters\_dictionary[i] = **0** letters\_dictionary[i] += **1** return dict(sorted(letters\_dictionary.items()))  
  
def spiral\_route\_cipher(message: str**,** route\_step : int):  
 start\_time = datetime.now()  
 encrypted = encrypt(message**,** route\_step)  
 encrypt\_time = datetime.now() - start\_time  
 print('\nЗашифрованное сообщение:'**,** encrypted)  
 start\_time = datetime.now()  
 decrypted = decrypt(encrypted**,** route\_step)  
 decrypt\_time = datetime.now() - start\_time  
 print('\nРасшифрованное сообщение:'**,** decrypted)  
 print('\nВремя зашифрования:'**,** encrypt\_time)  
 print('Время расшифрования:'**,** decrypt\_time)  
 return get\_letters\_amount(encrypted)

Листинг 1 – Реализация маршрутной перестановки по спирали

С помощью приведенного кода зашифруем и расшифруем сообщение, приведенное в примере:

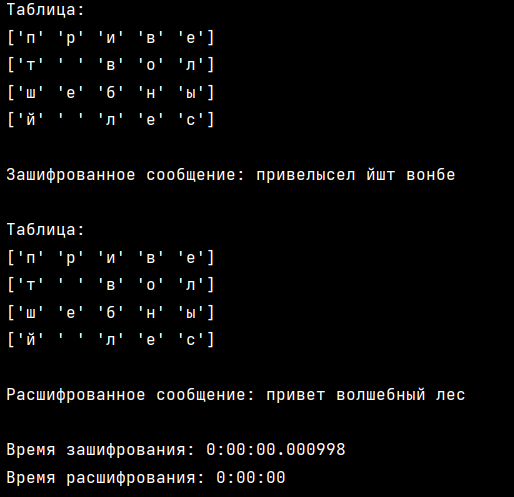


Рисунок 4 – Результат выполнения программы для маршрутной перестановки

Время шифрования в данном алгоритме вышло больше времени дешифрования.

Для **шифра множественной перестановки** были взяты два ключа. Первый – имя – указывает на количество строк, второй – фамилия – на количество столбцов в таблице.

То есть, вверху таблицы записывается первый ключ, слева вертикально записывается второй ключ. Каждому символу в каждом ключе присваивается порядковый номер по следованию букв алфавита. В

Рассмотрим данный шифр на том же примере:



Рисунок 5 – Таблица для множественной перестановки без сортировки

Далее строки сортируются по порядку:



Рисунок 6 – Таблица для множественной перестановки с отсортированными строками

Затем по порядку сортируются столбцы:



Рисунок 7 – Отсортированная таблица для множественной перестановки

Сообщение считывается по столбцам. Полученное зашифрованное сообщение: «н ст б лв ш ее е ып в и л йр о».

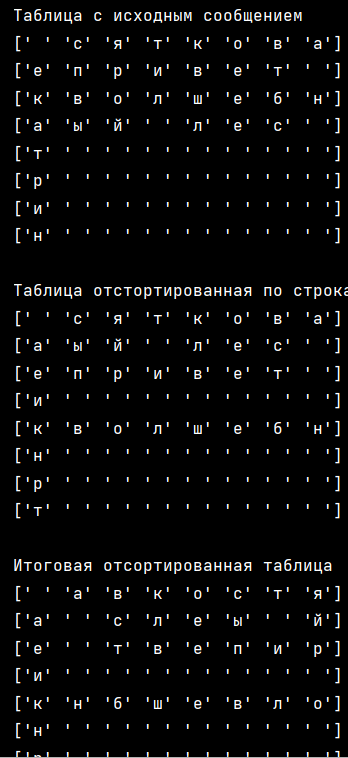
Дешифрование выполняется в обратном порядке: сообщение записывается по столбцам, восстанавливается порядок столбцов и строк, после чего сообщение считывается построчно.

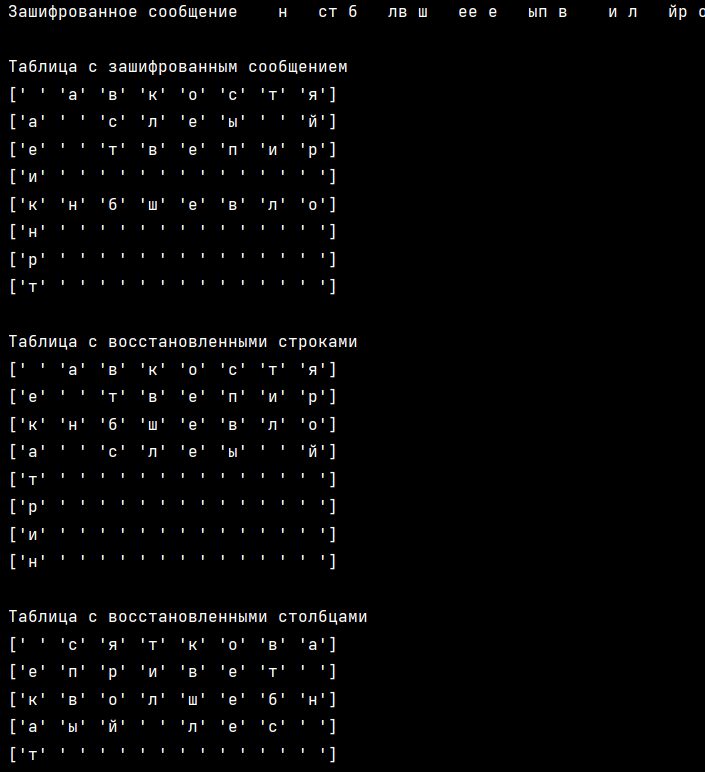
Ниже приведен код основных методов программы, реализующей алгоритм, на языке Python:

import numpy as np  
from datetime import datetime  
  
def encrypt(keyword\_column : str**,** keyword\_row : str**,** message: str) -> str:  
 # удаляем повторяющиеся буквы в ключевых словах  
 keyword\_column = ''.join(sorted(set(keyword\_column)**,** key=keyword\_column.index))  
 keyword\_row = ''.join(sorted(set(keyword\_row)**,** key=keyword\_row.index))  
  
 # формируем матрицу  
 column\_list = list(keyword\_column)  
 row\_list = list(' ' + keyword\_row)  
 row\_list = np.reshape(row\_list**,** (len(keyword\_row) + **1, 1**))  
  
 while len(message) / (len(keyword\_column) \* len(keyword\_row)) > **1**:  
 message = message[:-**1**]  
 while len(message) % (len(keyword\_column) \* len(keyword\_row)) != **0**:  
 message += ' '  
  
 matrix\_message = list(message)  
 matrix\_message = np.reshape(matrix\_message**,** (len(keyword\_row)**,** len(keyword\_column)))  
  
 matrix = np.vstack([column\_list**,** matrix\_message])  
 matrix = np.hstack([row\_list**,** matrix])  
  
 print('\nТаблица с исходным сообщением')  
 for row in matrix:  
 print(row)  
  
 # сортируем строки и столбцы матрицы  
 matrix = matrix[np.argsort(matrix[:**,0**])]  
 print('\nТаблица отстортированная по строкам')  
 for row in matrix:  
 print(row)  
  
 matrix = np.transpose(matrix)  
 matrix = matrix[np.argsort(matrix[:**,0**])]  
 matrix = np.transpose(matrix)  
 print('\nИтоговая отсортированная таблица')  
 for row in matrix:  
 print(row)  
  
 # записываем сообщения по столбцам  
 matrix = matrix[**1**:**, 1**:]  
 res = ''  
 i**,** j = matrix.shape  
 for column in range(j):  
 for row in range(i):  
 res += matrix[row][column]  
 return res  
  
def decrypt(keyword\_column : str**,** keyword\_row : str**,** message: str):  
 keyword\_column = ''.join(sorted(set(keyword\_column)**,** key=keyword\_column.index))  
 keyword\_row = ''.join(sorted(set(keyword\_row)**,** key=keyword\_row.index))  
  
 # формируем матрицу  
 column\_list = sorted(keyword\_column)  
 row\_list = sorted(' ' + keyword\_row)  
 row\_list = np.reshape(row\_list**,** (len(keyword\_row) + **1, 1**))  
 matrix\_message = np.empty((len(keyword\_row)**,** len(keyword\_column))**,** str)  
  
 for i in range(len(keyword\_column)):  
 for j in range(len(keyword\_row)):  
 letter = message[**0**]  
 message = message[**1**:]  
 matrix\_message[j][i] = letter  
  
 matrix = np.vstack([column\_list**,** matrix\_message])  
 matrix = np.hstack([row\_list**,** matrix])  
  
 print('\nТаблица с зашифрованным сообщением')  
 for row in matrix:  
 print(row)  
  
 # сортируем матрицу по ключам  
 order\_rows = []  
 for character in sorted(' ' + keyword\_row):  
 order\_rows.append(list(' ' + keyword\_row).index(character))  
 matrix = matrix[np.argsort(order\_rows)]  
 print('\nТаблица с восстановленными строками')  
 for row in matrix:  
 print(row)  
  
 matrix = np.transpose(matrix)  
 order\_columns = []  
 for character in sorted(' ' + keyword\_column):  
 order\_columns.append(list(' ' + keyword\_column).index(character))  
 matrix = matrix[np.argsort(order\_columns)]  
 matrix = np.transpose(matrix)  
 print('\nТаблица с восстановленными столбцами')  
 for row in matrix:  
 print(row)  
  
 # записывам соощение по строкам  
 matrix = matrix[**1**:**, 1**:]  
 res = ''.join(matrix.flatten())  
 return res  
  
def get\_letters\_amount(seq):  
 letters\_dictionary = {}  
 for i in seq:  
 if i.isalpha():  
 if i not in letters\_dictionary:  
 letters\_dictionary[i] = **0** letters\_dictionary[i] += **1** return dict(sorted(letters\_dictionary.items()))  
  
def multiple\_permutation(keyword\_column : str**,** keyword\_row : str**,** message: str):  
 start\_time = datetime.now()  
 encrypted = encrypt(keyword\_column**,** keyword\_row**,** message)  
 encrypt\_time = datetime.now() - start\_time  
 print('\nЗашифрованное сообщение'**,** encrypted)  
 start\_time = datetime.now()  
 decrypted = decrypt(keyword\_column**,** keyword\_row**,**encrypted)  
 print('\nРасшифрованное сообщение:'**,** decrypted)  
 decrypt\_time = datetime.now() - start\_time  
 print('\nВремя зашифрования:'**,** encrypt\_time)  
 print('Время расшифрования:'**,** decrypt\_time)  
 return get\_letters\_amount(encrypted)

Листинг 2 – Реализация множественных перестановок

С помощью приведенного кода зашифруем и расшифруем сообщение, приведенное в примере:





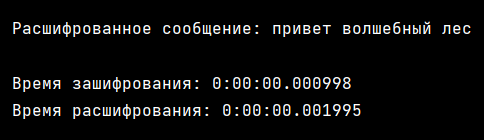


Рисунок 8 – Результат выполнения программы для множественной перестановки

Время зашифрования превысило время дешифрования. Отсюда можно сделать вывод о невысокой криптостойкости шифров перестановки. Это также подтверждает частотный анализ. Частота появления символов не изменилась после зашифрования.

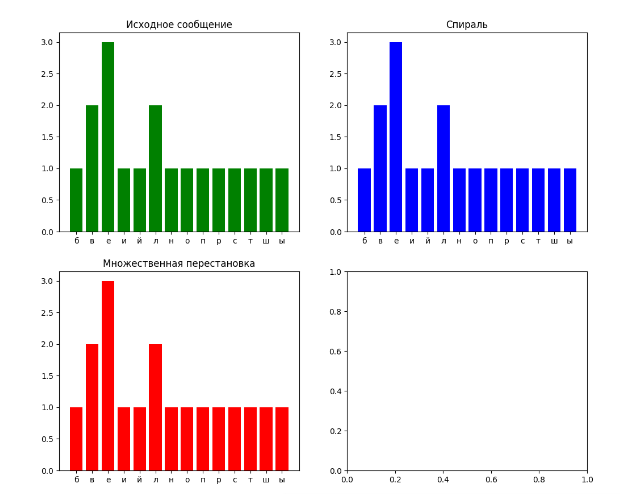


Рисунок 9 – Гистограммы частоты появления символов

**Ответы на вопросы для самоконтроля:**

1. В чем заключается основная идея криптографических преобразований на основе шифров перестановки?

Сущность перестановочного шифрования состоит в том, что исходный текст (М) и зашифрованный текст (С) основаны на использовании одного и того же алфавита, а тайной или ключевой информацией является алгоритм перестановки.

1. Привести классификационные признаки и дать сравнительную характеристику разновидностям перестановочных шифров.

Шифры перестановки можно разделить на два класса:

* Шифры одинарной (простой) перестановки — при шифровании символы открытого текста перемещаются с исходных позиций в новые один раз.
* Шифры множественной (сложной) перестановки — при шифровании символы открытого текста перемещаются с исходных позиций в новые несколько раз.

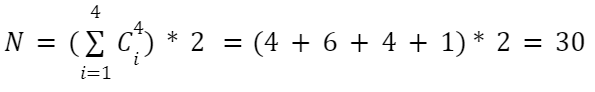
Шифр простой блочной перестановки: при больших значениях *n* приходится работать с таблицами, состоящими из большого числа столбцов; для сообщений разной длинны необходимо создавать разные таблицы перестановок – не очень удобен.

Шифр маршрутной перестановки: представлен геометрической фигурой, обычно прямоугольник или прямоугольная матрица. В ячейки этой фигуры по определенному маршруту (слева направо, сверху вниз или каким-либо иным образом) записывается открытый текст. Для получения шифрограммы нужно записать символы этого сообщения в иной последовательности, т. е. по иному маршруту.

Шифр множественной перестановки: минимум двукратная перестановка символов шифруемого сообщения (перемещения не только столбцов, но и строк); использует 2 основных ключа (для строк и столбцов), а также к ключевой информацию мы можем относить также способы вписывания сообщения и считывания отдельных символов из текущего столбца матрицы.

1. Сколько разновидностей шифров, подобных шифру Цезаря, можно составить для алфавитов русского и белорусского языков?

Существует 4 разновидности шифра Цезаря, но так как мы можем комбинировать несколько шифров, необходимо найти сумму сочетаний, умноженную на 2, так как 2 алфавита:



1. Охарактеризовать криптостойкость перестановочных и подстановочных шифров.

Перестановочные шифры (как и подстановочные) обладают низкой криптостойкостью, так как легко взламываются брутфорсом. Наиболее криптостойким вариантом является сочетание подстановочного и перестановочного шифра.

1. Привести примеры и дать характеристику перестановочным шифрам, не рассмотренным в материалах к данной лабораторной работе.

***Шифрование методом магического квадрата.*** Магический квадрат — квадратная таблица с вписанными в клетки последовательными натуральными числами, начиная с единицы, которые при суммировании по столбцам, строкам и диагоналям дают одно и то же число. Исходный текст вписывают в магический квадрат в соответствии с нумерацией его клеток. Для получения шифртекста таблицу считывают по строкам или столбцам. В качестве ключа для восстановления исходного текста может быть ссылка на номер магического квадрата в некоторой базе, которая известна отправителю и получателю шифртекста.

***Шифрование с помощью решёток.*** Выбирается натуральное число *k* > 1, и квадрат размерности *k×k* построчно заполняется числами 1, 2, ..., *k*. Квадрат поворачивается по часовой стрелке на 90° и размещается вплотную к предыдущему квадрату. Аналогичные действия совершаются еще два раза, так чтобы в результате из четырех малых квадратов образовался один большой с длиной стороны *2k*. алее из большого квадрата вырезаются клетки с числами от 1 до k2, для каждого числа одна клетка. Процесс шифрования происходит следующим образом. Сделанная решетка (квадрат с прорезями) накладывается на чистый квадрат 2*k*×2*k* и в прорези по строчкам (т.е. слева направо и сверху вниз) вписываются первые буквы открытого текста. Затем решетка поворачивается на 90° по часовой стрелке и накладывается на частично заполненный квадрат, вписывание продолжается. После третьего поворота, наложения и вписывания все клетки квадрата будут заполнены.

1. Имеются ли предпочтения в выборе размеров используемой таблицы для перестановочных шифров?

Стойкость простой перестановки однозначно определяется размерами используемой матрицы перестановки. Например, при использовании матрицы 16\*16 число возможных перестановок достигает 1.4E26. Однако, процесс шифрования таких сообщений будет более трудоемким и при шифровании перестановкой полностью сохраняются вероятностные характеристики исходного текста, что облегчает криптоанализ.

1. Охарактеризовать основные методы взлома перестановочных шифров.

***Брутфорс*** – прямой перебор всех возможных вариантов перестановок.

Перебор ключей малой длины – перебор всех возможных ключей длиной до 9 символов.

***Поиск восхождением к вершине*** – берётся часть шифртекста (называемая ключом), и оставшаяся часть расшифровывается с его помощью. Затем вычисляется коэффициент вероятности принадлежности расшифрованного текста к естественному языку. В ключе производятся какие-либо изменения (например, перестановка первых двух букв местами) и снова расшифровывается текст и вычисляется коэффициент. Так повторяется, пока коэффициент не перестанет изменяться.

***Анализ на основе триграмм*** – можно попытаться вычислить примерный алгоритм перестановки, составляя наиболее часто встречающиеся в языке триграммы из символов сообщения.

**Вывод:** в ходе лабораторной работы было разработано приложение, позволяющее шифровать и расшифровывать сообщения с помощью шифров множественной перестановки и маршрутной перестановки (маршрут зигзаг), были построены гистограммы частоты встречаемости символов и измерено время шифрования/дешифования. Временные затраты незначительны.