Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Прикладная математика и компьютерная безопасность

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 11**

по Криптографические методы защиты информации

наименование дисциплины

Криптосистема Хилла

тема

Преподаватель В.И. Вайнштейн

подпись**,** дата инициалы, фамилия

Студент КИ17-01, 031722011 К.А. Василенко

номер группы, зачетной книжки подпись**,** дата инициалы, фамилия

Красноярск 2021

# ВВЕДЕНИЕ

Цель работы: реализовать криптосистему Хилла (шифрование и расшифрование) на любом языке программирования. Предусмотреть графический интерфейс.

Задачи:

1. реализовать программно криптосистему Хилла, предусматривая графический интерфейс;
2. провести тесты на работоспособность программы;
3. сделать отчёт о проделанной работе.

**Реализация шифра**

1. **Описание шифра**

Шифр Хилла — полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике.

Шифр Хилла является полиграммным шифром, который может использовать большие блоки с помощью линейной алгебры. Каждой букве алфавита сопоставляется число по модулю 26. Для латинского алфавита часто используется простейшая схема: A = 0, B = 1, …, Z = 25, но это не является существенным свойством шифра. Блок из n букв рассматривается как n-мерный вектор и умножается по модулю 26 на матрицу размера n × n. Если в качестве основания модуля используется число больше чем 26, то можно использовать другую числовую схему для сопоставления буквам чисел и добавить пробелы и знаки пунктуации. Элементы матрицы являются ключом. Матрица должна быть обратима в Zn26, чтобы была возможна операция расшифрования.

Для n = 3 система может быть описана так:

*C = KP (mod 26)*

где *P* и *C* — векторы-столбцы высоты 3, представляющие открытый и зашифрованный текст соответственно, *K* — матрица 3 × 3, представляющая ключ шифрования. Операции выполняются по модулю 26.

Для того, чтобы расшифровать сообщение, требуется получить обратную матрицу ключа *{\displaystyle K^{-1}}K-1*. Существуют стандартные методы вычисления обратных матриц, но не все матрицы имеют обратную. Матрица будет иметь обратную в том и только в том случае, когда её детерминант не равен нулю и не имеет общих делителей с основанием модуля. Если детерминант матрицы равен нулю или имеет общие делители с основанием модуля, то такая матрица не может использоваться в шифре Хилла, и должна быть выбрана другая матрица (в противном случае шифротекст будет невозможно расшифровать). Тем не менее, матрицы, которые удовлетворяют вышеприведенным условиям, существуют в изобилии.

В общем случае, алгоритм шифрования может быть выражен в следующем виде:

**Шифрование:** *C = E(K, P) = KP (mod 26)*

**Расшифрование:** *P = D(K, C) = K-1C (mod 26) = K-1KP (mod 26) = P*

{\displaystyle c\_{j}=(m\_{j}+k\_{j})\mod {n}}{\displaystyle m\_{j}=(c\_{j}+n-k\_{j})\mod {n}}**Программная реализация**

Программа написана на языке Python с использованием библиотеки PyQt5 и Qt designer для отрисовки графического интерфейса.

1. **Листинг с описанием основного алгоритма программы**

**hill.py**

**from** math **import** sqrt  
**from** numpy **import** linalg, round, delete  
**from** PyQt5 **import** QtWidgets  
  
# text = 'кимоно'  
# K = 'коловорот'  
  
**def** matrix\_algDop(arr, i, j, mod):  
 tmp = delete(delete(arr,i,axis=0), j, axis=1)  
 detTmp = round(linalg.det(tmp))  
 **return** (pow(-1, (i+1)+(j+1)) \* detTmp) % mod  
  
**def** exgcd(a, b):  
 x, xx, y, yy = 1, 0, 0, 1  
 **while** b:  
 q = a // b  
 a, b = b, a % b  
 x, xx = xx, x - xx\*q  
 y, yy = yy, y - yy\*q  
 **return** x  
  
**def** multiply(vector, matr, mod):  
 res = []  
 i = 0 #номер столбца  
 **while** i != len(matr):  
 res1 = 0  
 **for** j **in** range(len(vector)):  
 res1+=((matr[j][i] \* vector[j]) % mod)  
 res.append(res1 % mod)  
 i+=1  
 **return** res  
  
**def** transpose(matr):  
 i = 0  
 res = []  
 **while** i != len(matr):  
 res1 = []  
 **for** j **in** range(len(matr)):  
 res1.append(matr[j][i])  
 res.append(res1)  
 i+=1  
 **return** res  
  
**def** codingHill(text, K):  
 **if not** text **or not** K:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Введите текст!")  
 msgBox.exec\_()  
 **return** ("")  
 **if** sqrt(len(K)) - int(sqrt(len(K))) != 0:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Error1. Длина ключа должна быть квадратом целого числа!")  
 msgBox.exec\_()  
 **return** ("")  
 **if** len(text) % int(sqrt(len(K))) != 0:  
 count = len(text)  
 **while** count % int(sqrt(len(K))) != 0:  
 count += 1  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Error2. Длина текста должна быть кратна sqrt(key)!")  
 msgBox.exec\_()  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Ближайшая длина, кратная sqrt(key) = " + str(count))  
 msgBox.exec\_()  
 **return** ("")  
  
 dict = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890!"№;%:?\*()\_-+**\\**/.,@#$^&[]{}**\'**<>`~|**\t\n** '  
 print(len(dict))  
  
 size = int(sqrt(len(K)))  
 key = []  
 **for** i **in** range(int(size)):  
 key1 = []  
 **for** j **in** range(int(size)):  
 key1.append(dict.index(K[j]))  
 K = K[int(size):]  
 key.append(key1)  
 **for** i **in** key:  
 print('key', i)  
  
 lenText = len(text)  
 textMatr = []  
 **for** k **in** range(lenText//int(size)):  
 P1 = []  
 **for** p **in** range(int(size)):  
 P1.append(dict.index(text[p]))  
 text = text[int(size):]  
 textMatr.append(P1)  
 **for** i **in** textMatr:  
 print('text', i)  
  
 multMatr = []  
 **for** i **in** range(len(textMatr)):  
 multMatr.append(multiply(textMatr[i], key, len(dict)))  
 **for** i **in** multMatr:  
 print(i)  
  
 res = ""  
 **for** i **in** range(len(multMatr)):  
 **for** j **in** range(len(multMatr[i])):  
 res+=dict[multMatr[i][j]]  
 print(res)  
 **return** res  
  
**def** decodingHill(res, K):  
 **if not** K **or not** res:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Введите текст!")  
 msgBox.exec\_()  
 **return** ("")  
 **if** sqrt(len(K)) - int(sqrt(len(K))) != 0:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Error1. Длина ключа должна быть квадратом целого числа!")  
 msgBox.exec\_()  
 **return** ("")  
  
 dict = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890!"№;%:?\*()\_-+**\\**/.,@#$^&[]{}**\'**<>`~|**\t\n** '  
  
 size = int(sqrt(len(K)))  
 text = []  
 **for** i **in** range((len(res)//size)):  
 C2 = []  
 **for** j **in** range(size):  
 C2.append(dict.index(res[j]))  
 res = res[size:]  
 text.append(C2)  
  
 key = []  
 **for** i **in** range(int(size)):  
 key1 = []  
 **for** j **in** range(int(size)):  
 key1.append(dict.index(K[j]))  
 K = K[int(size):]  
 key.append(key1)  
  
 **if** int(linalg.det(key)) % len(dict) == 0:  
 msgBox = QtWidgets.QMessageBox()  
 msgBox.setWindowTitle("Ошибка!")  
 msgBox.setText("Det не может быть равен 0!")  
 msgBox.exec\_()  
 **return** ("")  
  
 keyObr = []  
 **for** i **in** range(len(key)):  
 keyM1 = []  
 **for** j **in** range(len(key[i])):  
 keyM1.append(matrix\_algDop(key, i, j, len(dict)))  
 keyObr.append(keyM1)  
 keyObr = transpose(keyObr)  
 print('keyM', keyObr)  
  
 detKey = round(linalg.det(key)) % len(dict)  
 print('detKey', detKey)  
 detObr = exgcd(detKey, len(dict))  
 print('detObr', detObr)  
  
 keyObrdetObr = []  
 **for** i **in** range(len(keyObr)):  
 tmp = []  
 **for** j **in** range(len(keyObr)):  
 tmp.append((keyObr[i][j] \* detObr) % len(dict))  
 keyObrdetObr.append(tmp)  
  
 multMatr = []  
 **for** i **in** range(len(text)):  
 multMatr.append(multiply(text[i], keyObrdetObr, len(dict)))  
 **for** i **in** multMatr:  
 print(i)  
  
 resText = ""  
 **for** i **in** range(len(multMatr)):  
 **for** j **in** range(len(multMatr[i])):  
 resText += dict[int(multMatr[i][j])]  
 print(resText)  
  
 **return** resText  
  
# codingHill(text, K)  
# decodingHill(codingHill(text, K), K)

1. **Примеры работы программы**

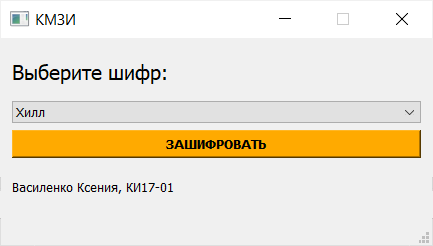


Рисунок 1 – Главное окно.

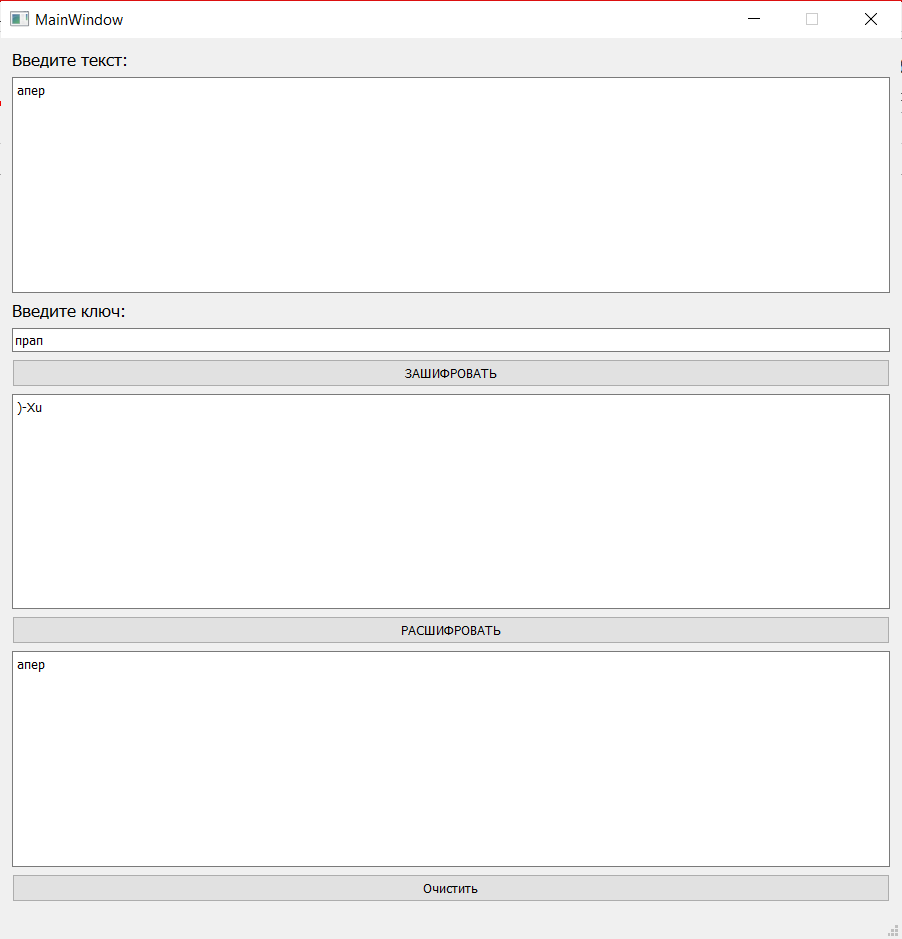


Рисунок 2 – Криптосистема Хилла.

# ЗАКЛЮЧЕНИЕ

Во время выполнения лабораторной работы №11 мною была изучена и реализована программно криптосистема Хилла, а также предусмотрен графический интерфейс с помощью PyQt5 и Qt designer.