Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №1**

|  |
| --- |
| Простые симметричные шифры |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | Р. С. Шиманович |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 3

4 Теоретические сведения 4

5 Ход выполнения 5

5.1 Блок-схемы алгоритмов шифрования и дешифрования 5

5.2 Составленная программа 7

5.3 Модификация алгоритма 11

5.4 Модифицированные блок-схемы 11

5.5 Модифицированная программа 12

5.6 Доказательство повышения криптостойкости 16

6 Выводы 18

Список использованных источников 19

## **1 Цели**

Ознакомиться с основами симметричного шифрования, ознакомиться с простыми симметричными криптографическими шифрами на основе методов подстановок, перестановок и гаммирования, освоить основные этапы проектирования и реализации симметричных шифров.

## **2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* разработать и составить в виде блок-схемы алгоритмы шифрования и дешифрования текста;
* составить на языке программирования программу, которая реализует данные алгоритмы;
* придумать оригинальный способ модификации шифра.

## **3 Описание задания**

Разработать и составить в виде блок-схемы алгоритмы шифрования и дешифрования текста с помощью шифра Хилла. Убедиться в правильности составления алгоритмов и затем на языке программирования составить программу, которая реализует данные алгоритмы.

На ряде контрольных примеров (не менее 10) открытого текста, состоящего из различного количества символов, проверьте правильность работы алгоритмов шифрования и дешифрования.

Самостоятельно или с помощью преподавателя придумать оригинальный способ модификации шифра с целью повышения его криптостойкости. Внести изменения в исходный алгоритм и программу. Проверить работоспособность алгоритма на тестовых примерах.

Доказать, что предложенный способ модификации действительно повышает криптостойкость.

## 

## **4 Теоретические сведения**

Шифр Хилла – это полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике. Он работает с блоками из n символов, которые представляются в виде n-мерных векторов. Каждому символу алфавита сопоставляется число по модулю m, где m – это размер алфавита. Например, для латинского алфавита можно использовать m = 26 и сопоставить A = 0, B = 1, …, Z = 25. Для шифрования блок исходного текста умножается по модулю m на матрицу ключа размера n x n, которая должна быть обратимой в кольце вычетов по модулю m. Для расшифрованию блок зашифрованного текста умножается на обратную матрицу ключа. Матричное умножение выполняется по правилу: элемент в i-й строке и j-м столбце результирующей матрицы равен сумме произведений элементов i-й строки первой матрицы на элементы j-го столбца второй матрицы по модулю m.

Шифр Хилла может работать в разных режимах в зависимости от выбора параметров n и m. В общем случае, чем больше n и m, тем больше пространство ключей и тем сложнее взломать шифр. Однако, при больших значениях n и m возникают трудности с генерацией обратимых матриц и с выравниванием текста на блоки равной длины. Поэтому на практике часто используются небольшие значения n (2 или 3) и m (26 или 29). Также можно использовать различные числовые схемы для сопоставления символов числам, например, добавить пробелы и знаки пунктуации к алфавиту.

Шифр Хилла не является криптостойким по современным меркам, так как его можно взломать с помощью статистического анализа частоты символов и биграмм (пар символов) в открытом и зашифрованном тексте.

## **5 Ход выполнения**

## **5.1 Блок схемы алгоритмов шифрования и дешифрования**

Блок-схема алгоритма шифрования шифром Хилла представлена на рисунке 1.

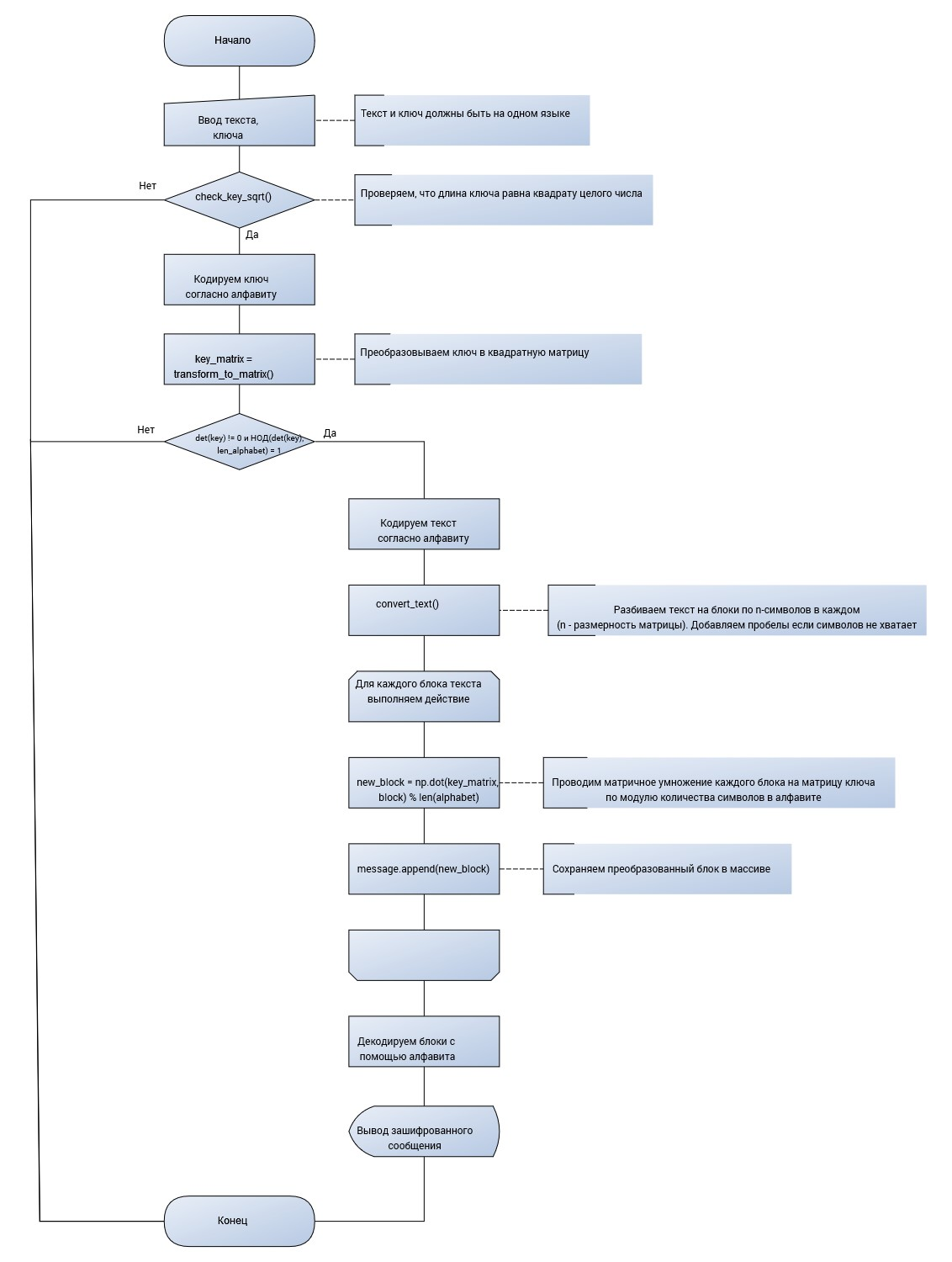


Рисунок 1 – Блок-схема алгоритма шифрования шифром Хилла

Блок-схема алгоритма дешифрования шифром Хилла представлена на рисунке 2.

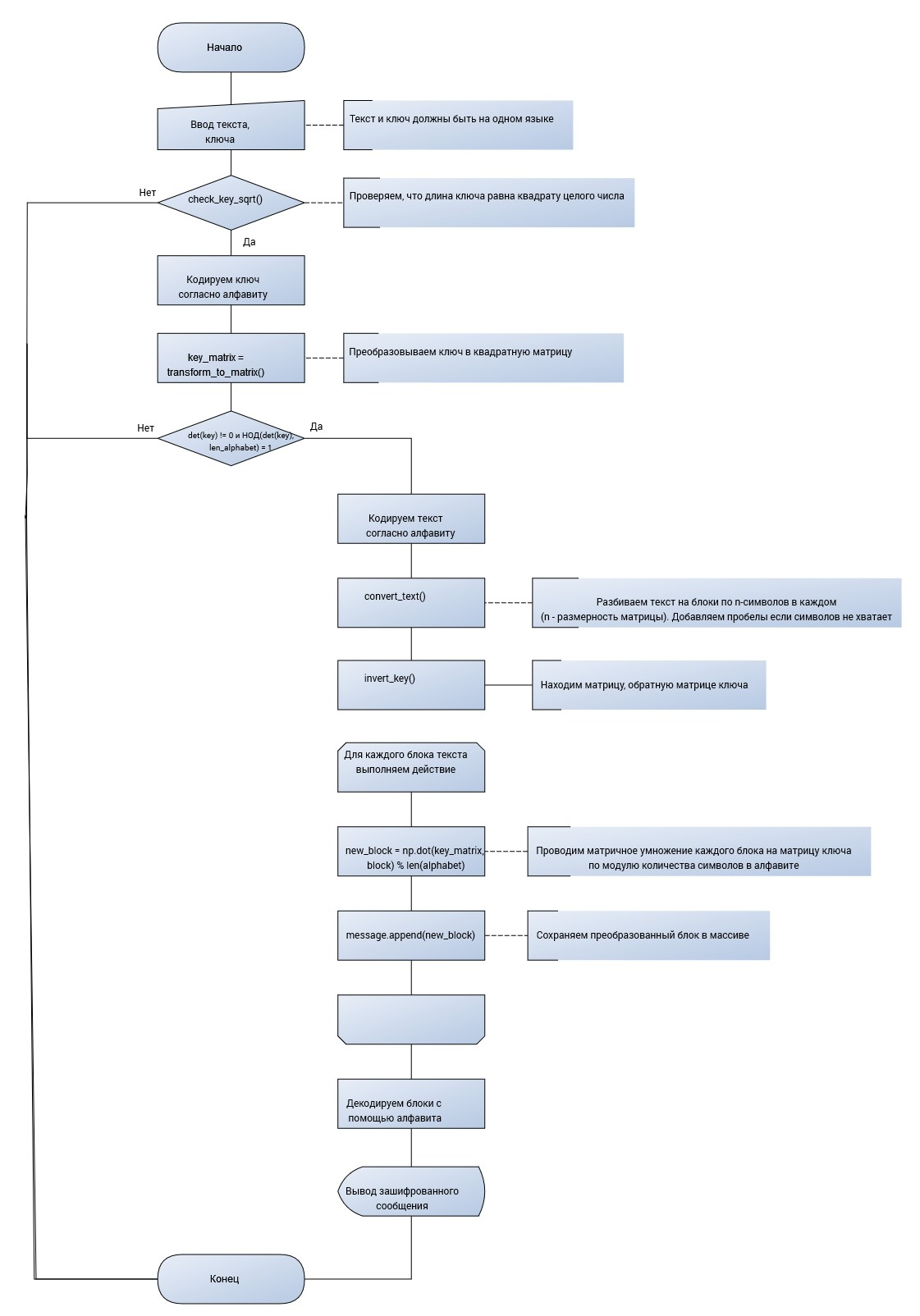


Рисунок 2 – Блок-схема алгоритма дешифрования шифром Хилла

## **5.2 Составленная программа**

Реализуем программу. В листинге 1 представлен код программы, который выполняет шифрование шифром Хилла.

Листинг 1 – Шифрование и дешифрование шифром Хилла

def convert\_text(self):

text\_blocks = []

for index in range(int(len(self.text) / self.key\_sqrt)):

tmp = np.array(self.text[self.key\_sqrt \* index:(self.key\_sqrt \*

index) +

self.key\_sqrt])

tmp = np.reshape(tmp, (self.key\_sqrt, 1))

text\_blocks.append(tmp)

self.text = np.array(text\_blocks)

def prepare\_data(self):

if self.check\_key\_sqrt():

for letter in self.key:

self.key\_matrix.append(self.alphabet[letter])

self.key\_matrix = np.reshape(np.array(self.key\_matrix),

(-1, self.key\_sqrt))

determinant = numpy.linalg.det(self.key\_matrix)

if determinant != 0:

if np.gcd(int(determinant), len(self.alphabet)) == 1:

text\_padded = self.pad\_text()

self.text = []

for letter in text\_padded:

self.text.append(self.alphabet[letter])

self.convert\_text()

return True

else:

QMessageBox.critical(None, "Ошибка!",

"Неверный ключ. Детерминат "

"матрицы ключа"

" и длина алфавита не являются взаимно"

" простыми",

QMessageBox.Ok)

return False

else:

QMessageBox.critical(None, "Ошибка!",

"Неверный ключ. Определитель матрицы ключа"

" равен нулю",

QMessageBox.Ok)

return False

else:

QMessageBox.critical(None, "Ошибка!",

"Неверный ключ. Длина ключа должна быть"

" квадратом целого числа",

QMessageBox.Ok)

return False

def check\_key\_sqrt(self):

self.key\_length = len(self.key)

self.key\_sqrt = math.sqrt(self.key\_length)

if (self.key\_sqrt - int(self.key\_sqrt)) != 0:

return False

else:

self.key\_sqrt = int(self.key\_sqrt)

return True

def pad\_text(self):

text\_len = len(self.text)

if (text\_len % self.key\_sqrt) != 0:

pad\_len = self.key\_sqrt - (text\_len % self.key\_sqrt)

text\_padded = self.text + pad\_len \* ' '

return text\_padded

return self.text

def get\_letter(self, target):

for letter, number in self.alphabet.items():

if number == target:

return letter

def multiplicative\_inverse(self, det):

mul\_inv = -1

for i in range(len(self.alphabet)):

inverse = det \* i

if inverse % len(self.alphabet) == 1:

mul\_inv = i

break

return mul\_inv

def invert\_key(self):

determinant = int(np.round(np.linalg.det(self.key\_matrix)))

determinant\_inverse = self.multiplicative\_inverse(

determinant % len(self.alphabet))

self.key\_matrix = determinant\_inverse \* np.round(

determinant \* np.linalg.inv(self.key\_matrix)).astype(int) % len(

self.alphabet)

def encrypt(self):

message = []

for block in self.text:

tmp = np.dot(self.key\_matrix, block) % len(self.alphabet)

message.append(tmp)

message = np.array(message).flatten()

encrypted\_text = [self.get\_letter(number) for number in message]

return "".join(encrypted\_text)

def encode(self):

if self.prepare\_data():

encrypted\_text = self.encrypt()

return encrypted\_text

else:

Окончание листинга 1

return ""

def decrypt(self):

self.invert\_key()

message = []

for block in self.text:

tmp = np.dot(self.key\_matrix, block) % len(self.alphabet)

message.append(tmp)

message = np.array(message).flatten()

encrypted\_text = [self.get\_letter(int(number)) for number in message]

return "".join(encrypted\_text)

def decode(self):

if self.prepare\_data():

decrypted\_text = self.decrypt()

return decrypted\_text

else:

return ""

На рисунках 3, 4, 5, 6, 7 представлены контрольные примеры работы программы с различными параметрами.

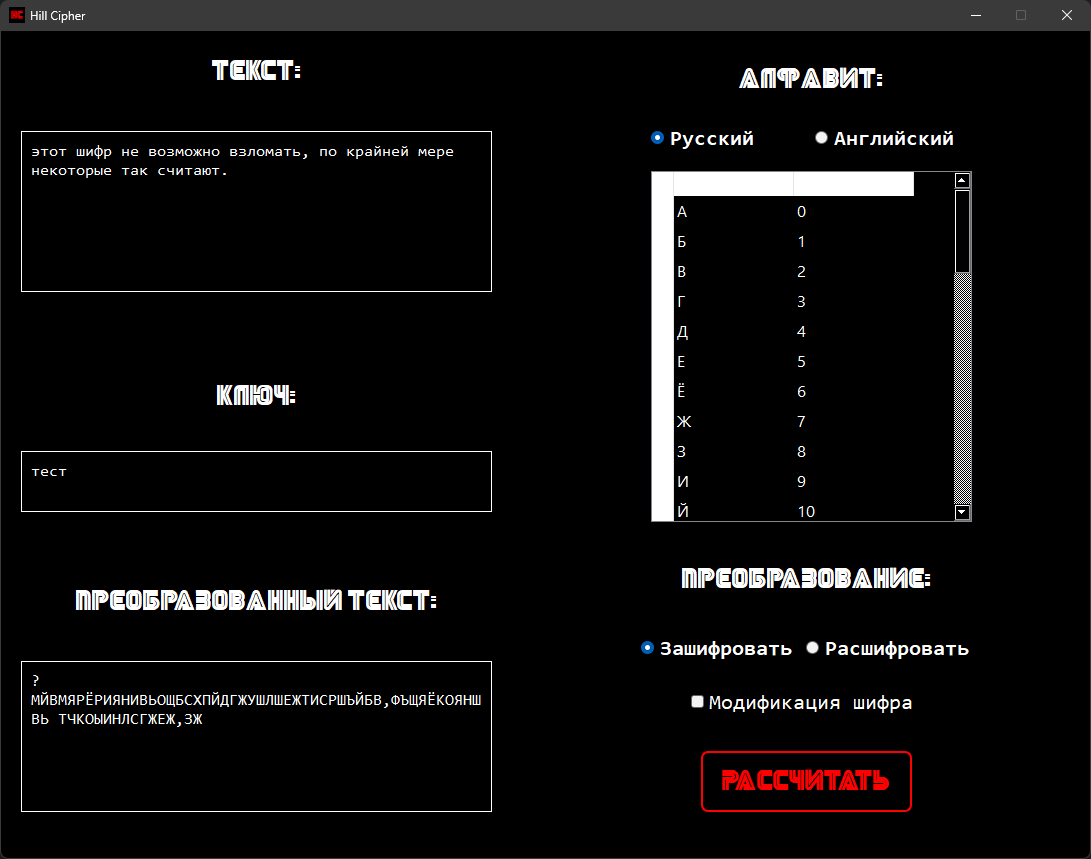


Рисунок 3 – Шифрование текста с ключом

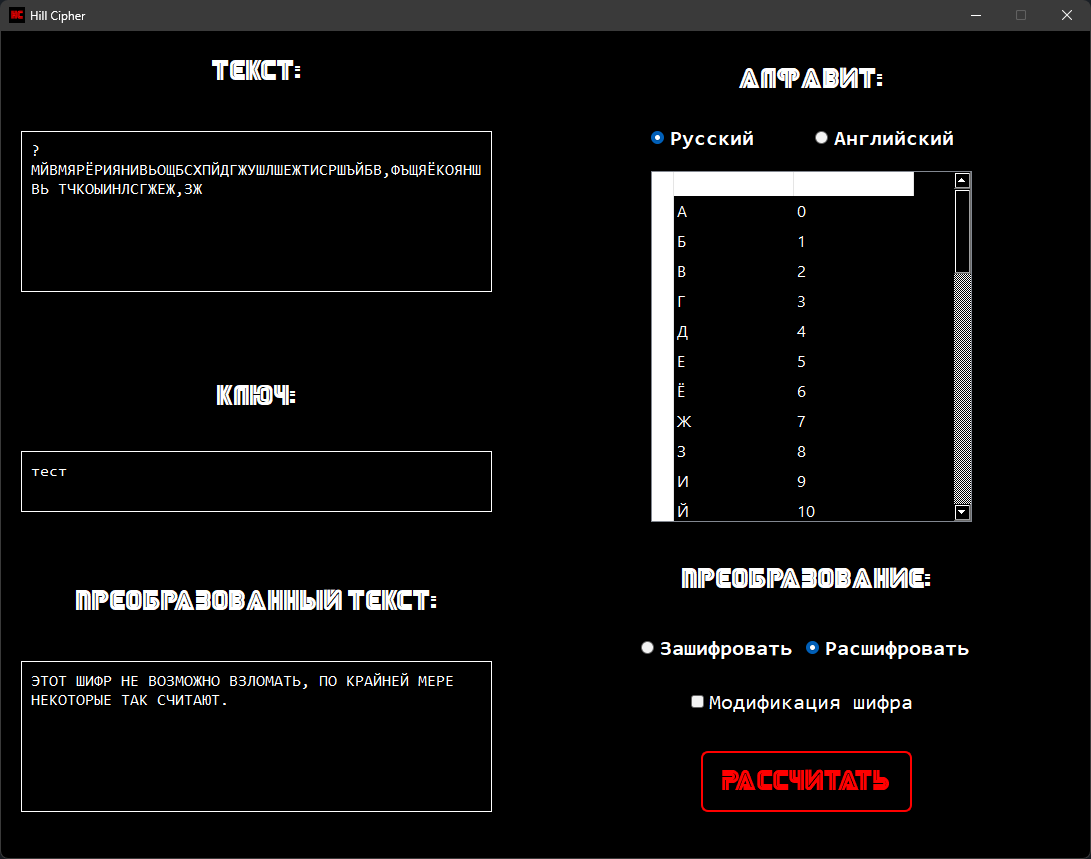


Рисунок 4 – Дешифрование текста с ключом

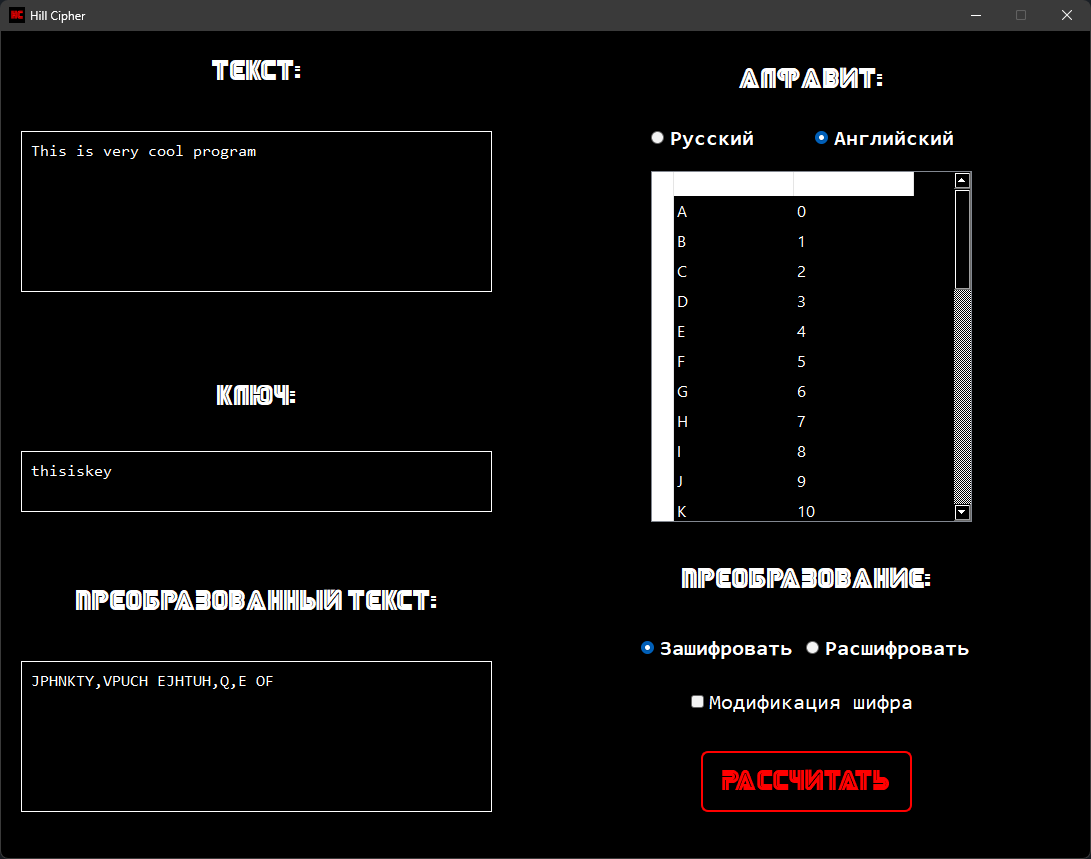


Рисунок 5 – Шифрование на английском языке

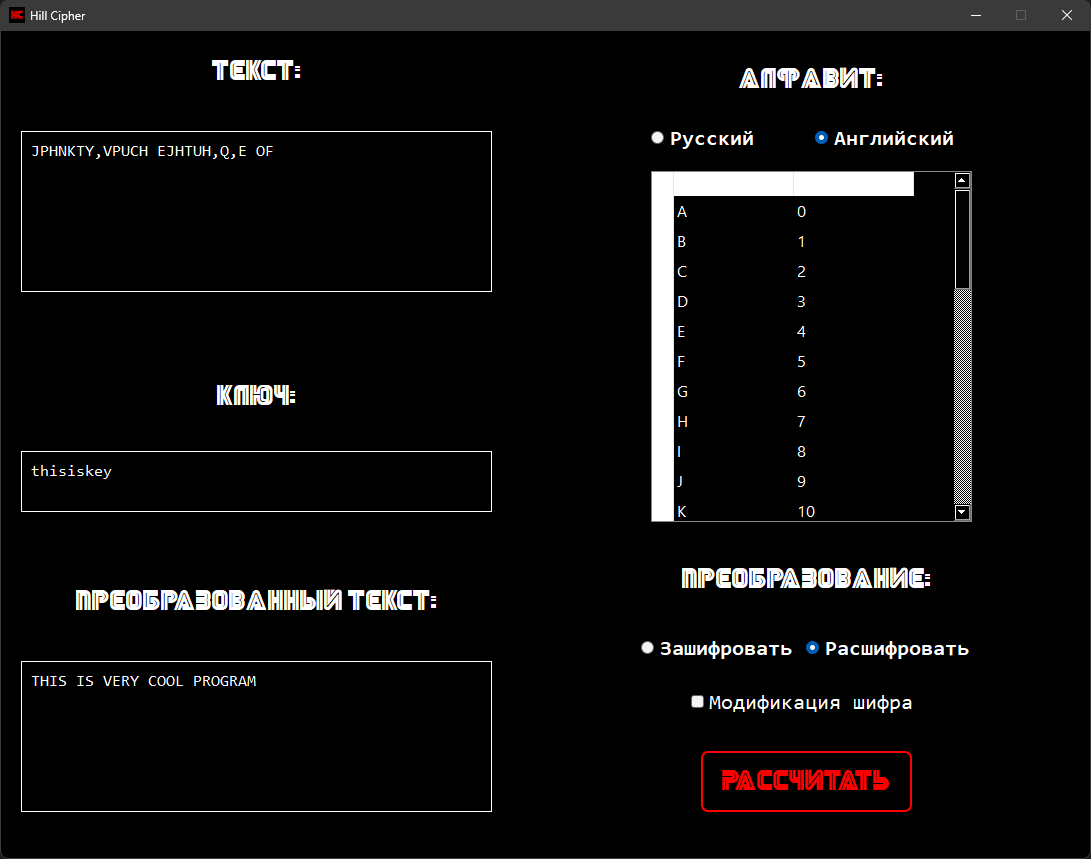


Рисунок 6 – Дешифрование на английском языке

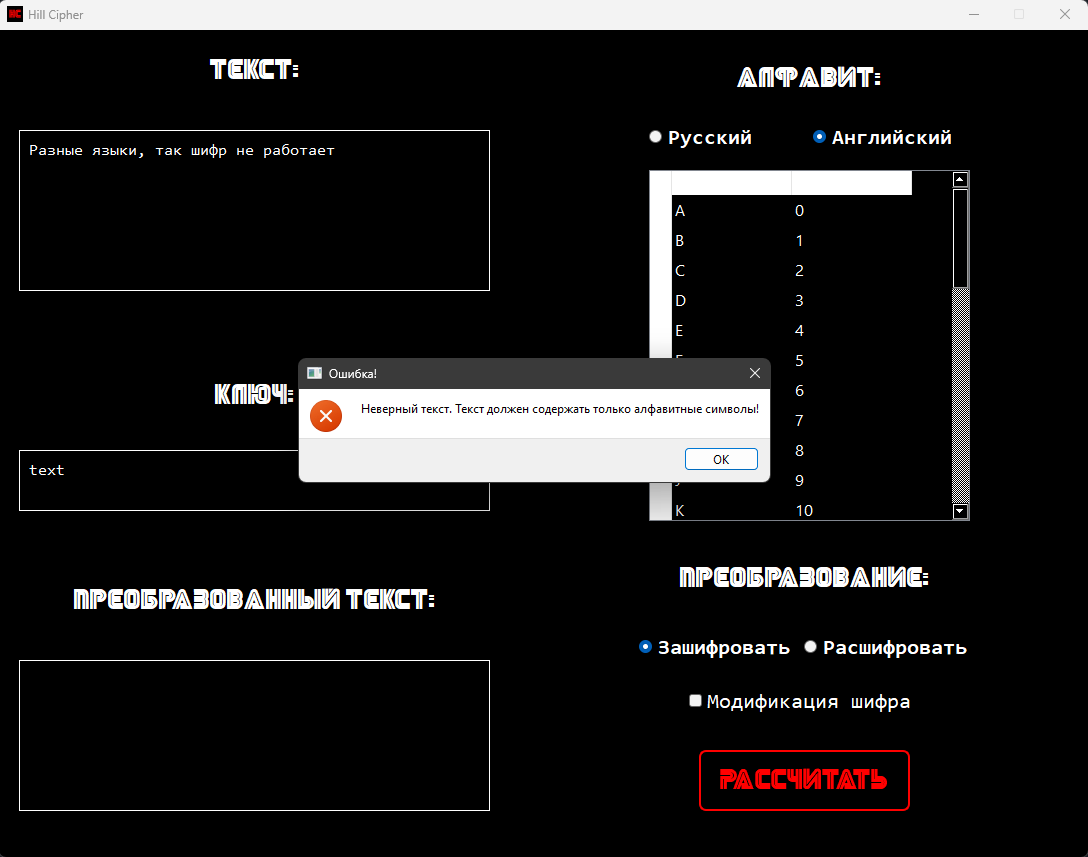


Рисунок 7 – Разный язык ключа и текста

## **5.3 Модификация алгоритма**

В качестве модификации алгоритма было принято решение выполнять с каждым блоком текста перед умножением на матрицу ключа следующие действия: прибавлять к каждому элементу матрицы столбца блока исходного текста соответствующее число из диагонали матрицы ключа, после этого сдвигать элементы в матрице столбце на один элемент влево, чтобы получить циклическую перестановку. Для дешифрования будет необходимо выполнить обратные действия.

## **5.4 Модифицированные блоксхемы**

Блок-схема модифицированного алгоритма шифрования шифром Хилла представлена на рисунке 8.

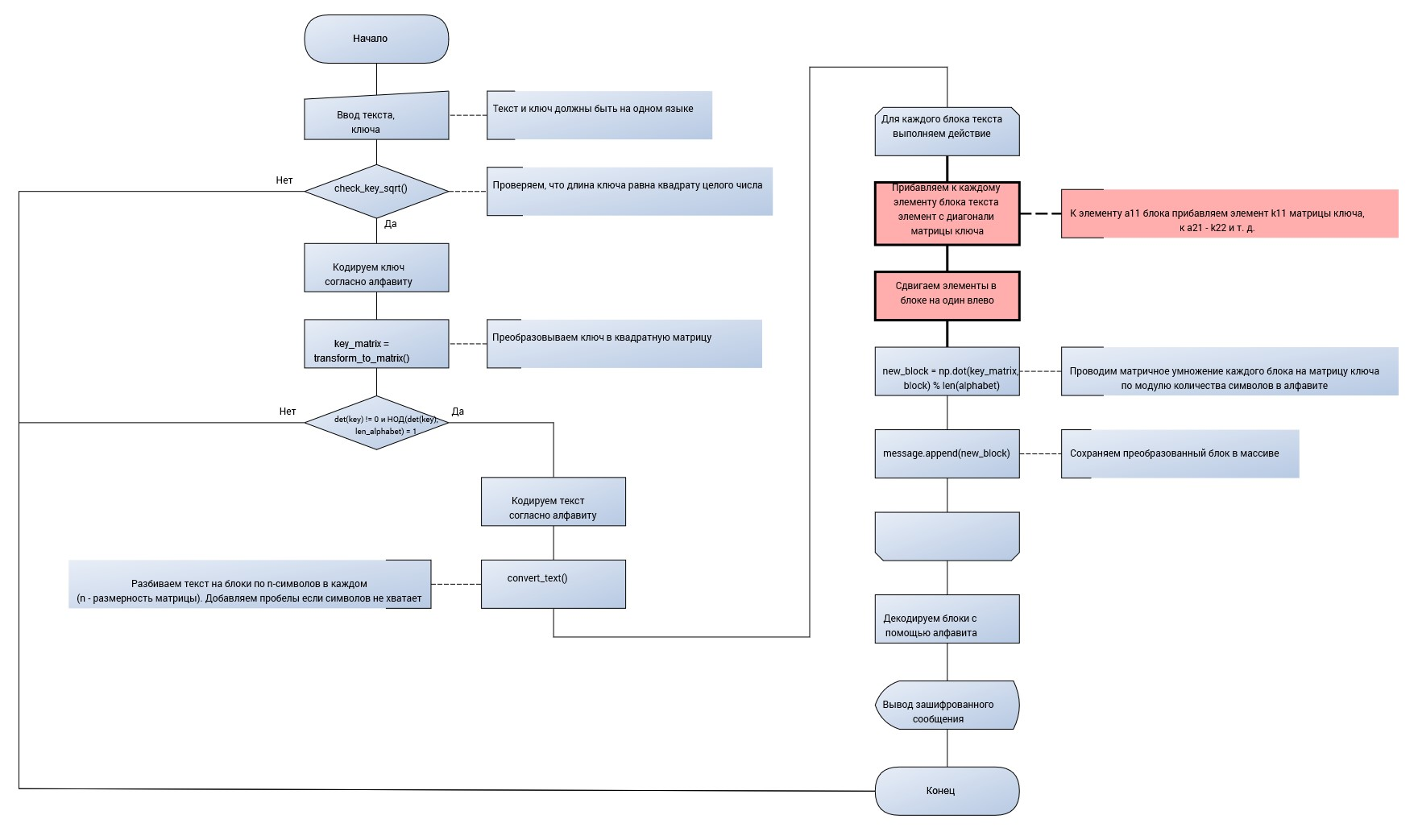


Рисунок 8 – Блок-схема алгоритма шифрования шифром Хилла

Блок-схема модифицированного алгоритма дешифрования шифром Хилла представлена на рисунке 9.

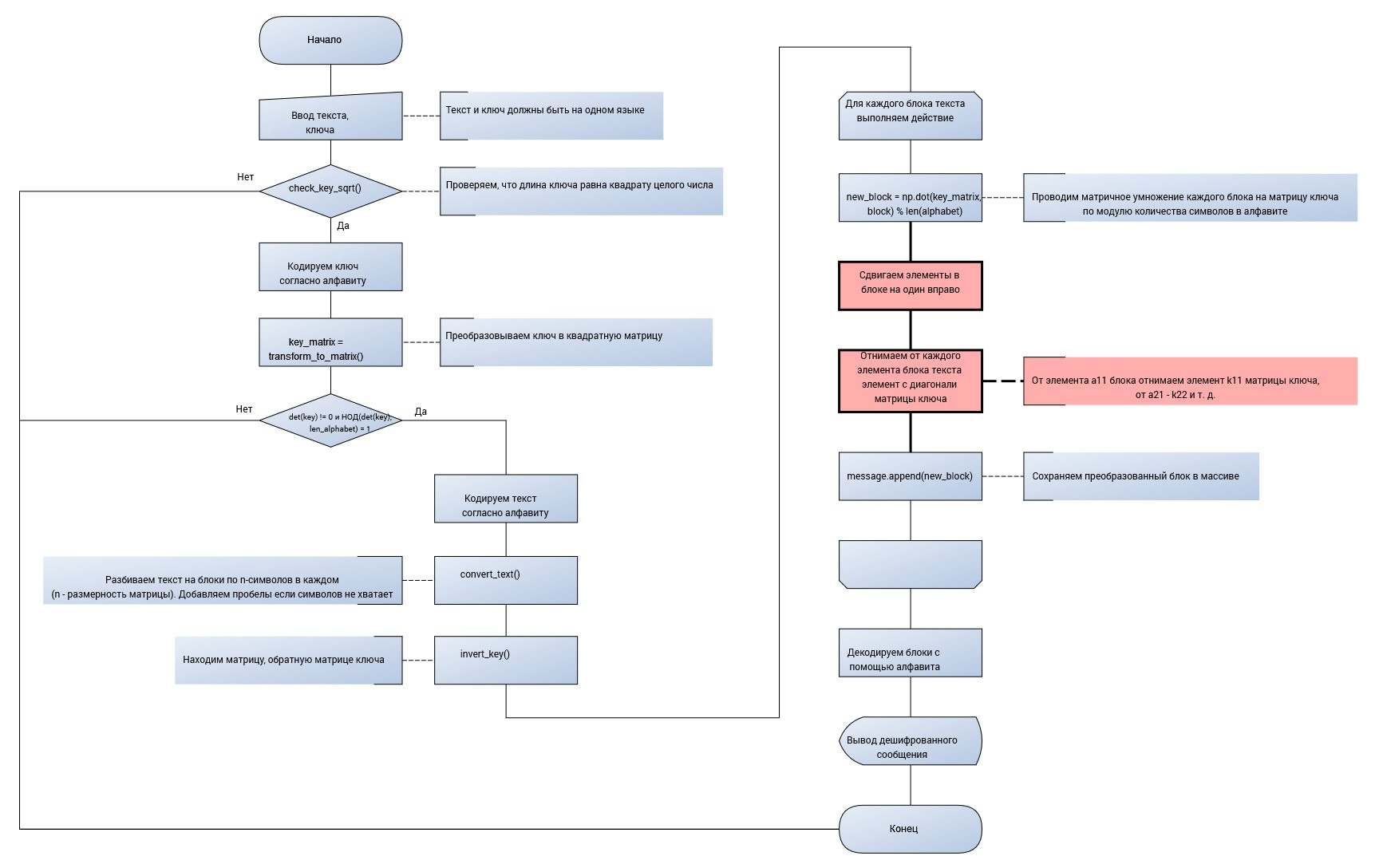


Рисунок 9 – Блок-схема алгоритма дешифрования шифром Хилла

## **5.5 Модифицированная программа**

Реализуем модификацию алгоритма в программе. В листинге 2 представлен код программы, который выполняет модифицированное шифрование шифром Хилла.

Листинг 2 – Изменённое шифрование и дешифрование шифром Хилла

def encrypt\_modification(self):

message = []

for block in self.text:

# Перемешиваем элементы блока в соответствии с функцией перестановки, зависящей от ключа

shuffled\_block = self.shuffle(block)

# Умножаем перемешанный блок на матрицу ключа по модулю размера алфавита

tmp = np.dot(self.key\_matrix, shuffled\_block) % len(self.alphabet)

message.append(tmp)

message = np.array(message).flatten()

encrypted\_text = [self.get\_letter(number) for number in message]

return "".join(encrypted\_text)

def shuffle(self, block):

# Определяем функцию перестановки, например, такую:

# f(x\_1, x\_2, ..., x\_k) = (x\_k + a\_{11}, x\_1 + a\_{22}, ..., x\_{k-1} + a\_{kk}) mod m

# где a\_{ii} - элементы на главной диагонали матрицы ключа

# k - размерность ключа

k = self.key\_sqrt

m = len(self.alphabet)

# Создаем пустой список для хранения перемешанных элементов блока

shuffled\_block = []

# Для каждого элемента блока применяем функцию перестановки и добавляем его в список

for i in range(k):

x\_i = block[i][0]

a\_ii = self.key\_matrix[i][i]

y\_i = (x\_i + a\_ii) % m

shuffled\_block.append(y\_i)

# Сдвигаем список на один элемент влево, чтобы получить циклическую перестановку

shuffled\_block = shuffled\_block[-1:] + shuffled\_block[:-1]

# Возвращаем перемешанный блок в виде вектора-столбца

return np.array(shuffled\_block).reshape(k, 1)

def encode\_modification(self):

if self.prepare\_data():

Окончание листинга 2

encrypted\_text = self.encrypt\_modification()

return encrypted\_text

else:

return ""

def decrypt\_modification(self):

# Находим обратную матрицу ключа

self.invert\_key()

message = []

for block in self.text:

# Умножаем зашифрованный блок на обратную матрицу ключа по модулю размера алфавита

tmp = np.dot(self.invert\_key\_matrix, block) % len(self.alphabet)

# Восстанавливаем элементы блока в соответствии с обратной функцией перестановки, зависящей от ключа

restored\_block = self.restore(tmp)

message.append(restored\_block)

message = np.array(message).flatten()

decrypted\_text = [self.get\_letter(int(number)) for number in message]

return "".join(decrypted\_text)

def restore(self, block):

# Определяем обратную функцию перестановки, например, такую:

# f^{-1}(y\_1, y\_2, ..., y\_k) = (y\_2 - a\_{22}, y\_3 - a\_{33}, ..., y\_1 - a\_{11}) mod m

# где a\_{ii} - элементы на главной диагонали матрицы ключа

# k - размерность ключа

k = self.key\_sqrt

m = len(self.alphabet)

# Создаем пустой список для хранения исходных элементов блока

original\_block = []

# Сдвигаем список на один элемент вправо, чтобы получить обратную циклическую перестановку

block = np.roll(block[:, 0], -1).reshape((-1, 1))

# Для каждого элемента блока применяем обратную функцию перестановки и добавляем его в список

for i in range(k):

y\_i = block[i][0]

a\_ii = self.key\_matrix[i][i]

x\_i = (y\_i - a\_ii) % m

original\_block.append(x\_i)

# Возвращаем исходный блок в виде вектора-столбца

return np.array(original\_block).reshape(k, 1)

def decode\_modification(self):

if self.prepare\_data():

decrypted\_text = self.decrypt\_modification()

return decrypted\_text

else:

return ""

На рисунках 10, 11, 12, 13, 14 представлены контрольные примеры работы модифицированной программы с различными параметрами.

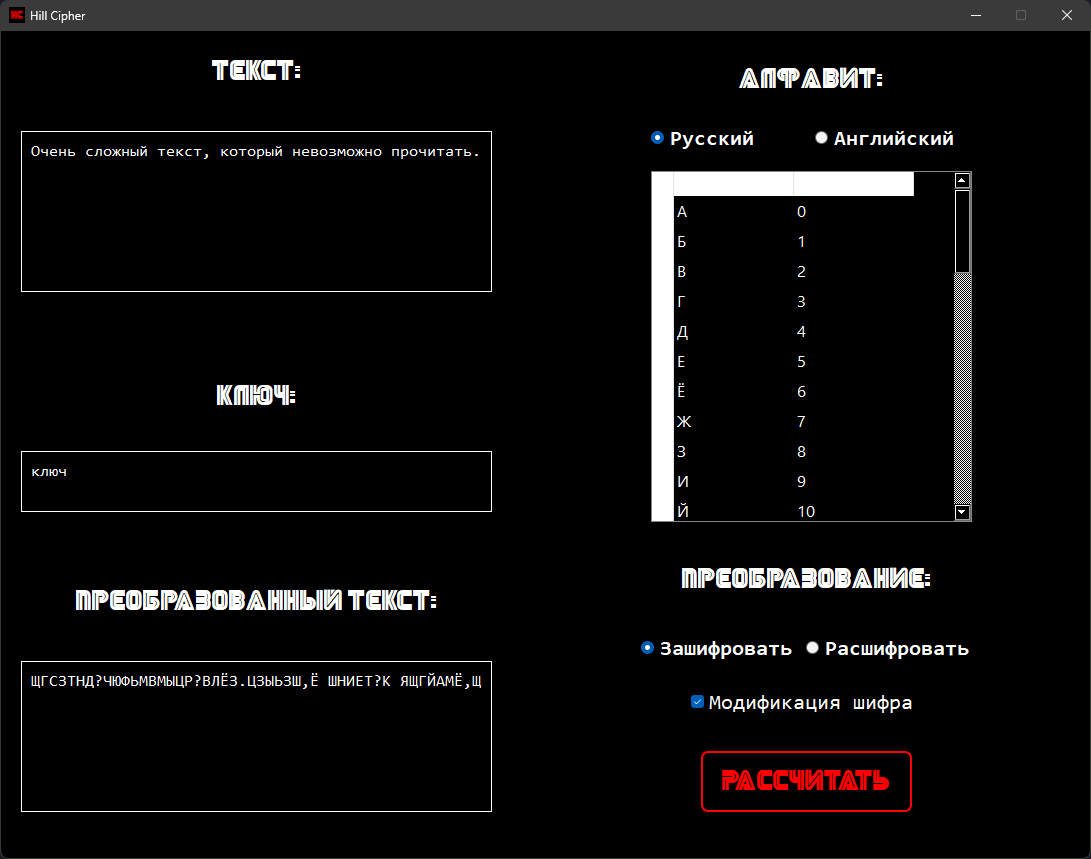


Рисунок 10 – Шифрование текста с ключом

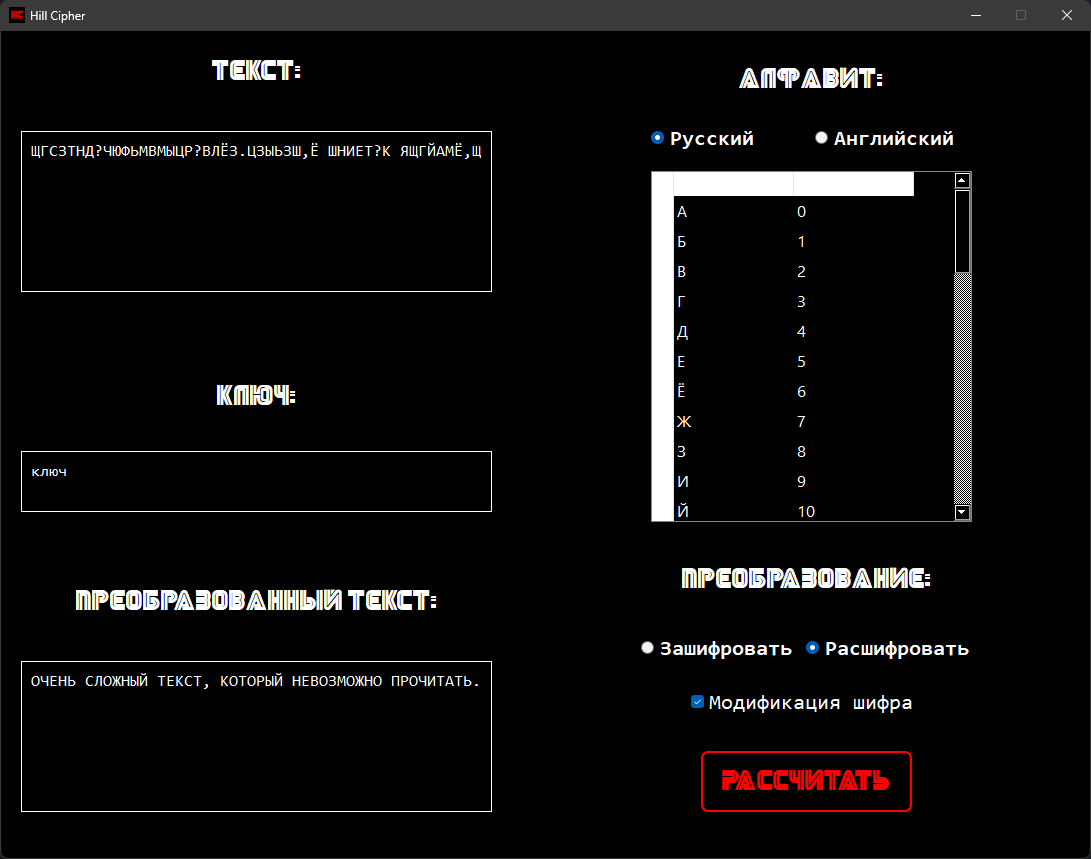


Рисунок 11 – Дешифрование текста с ключом

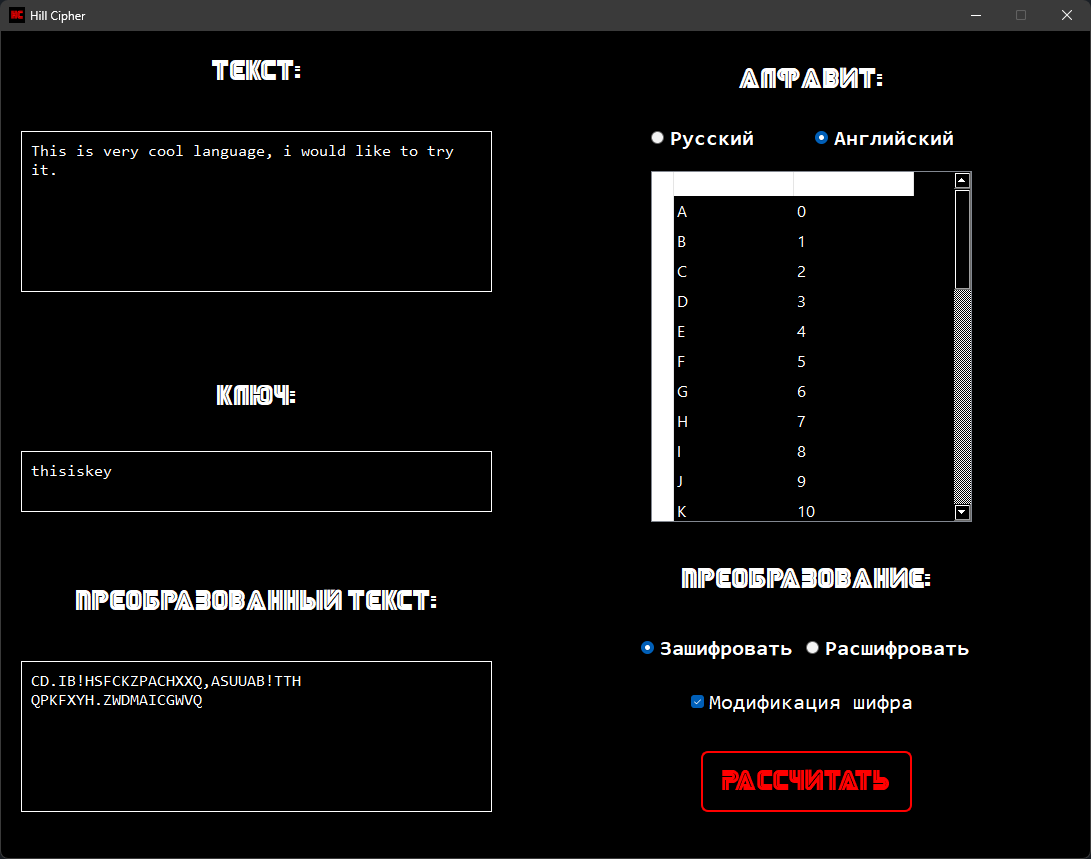


Рисунок 12 – Шифрование на английском языке

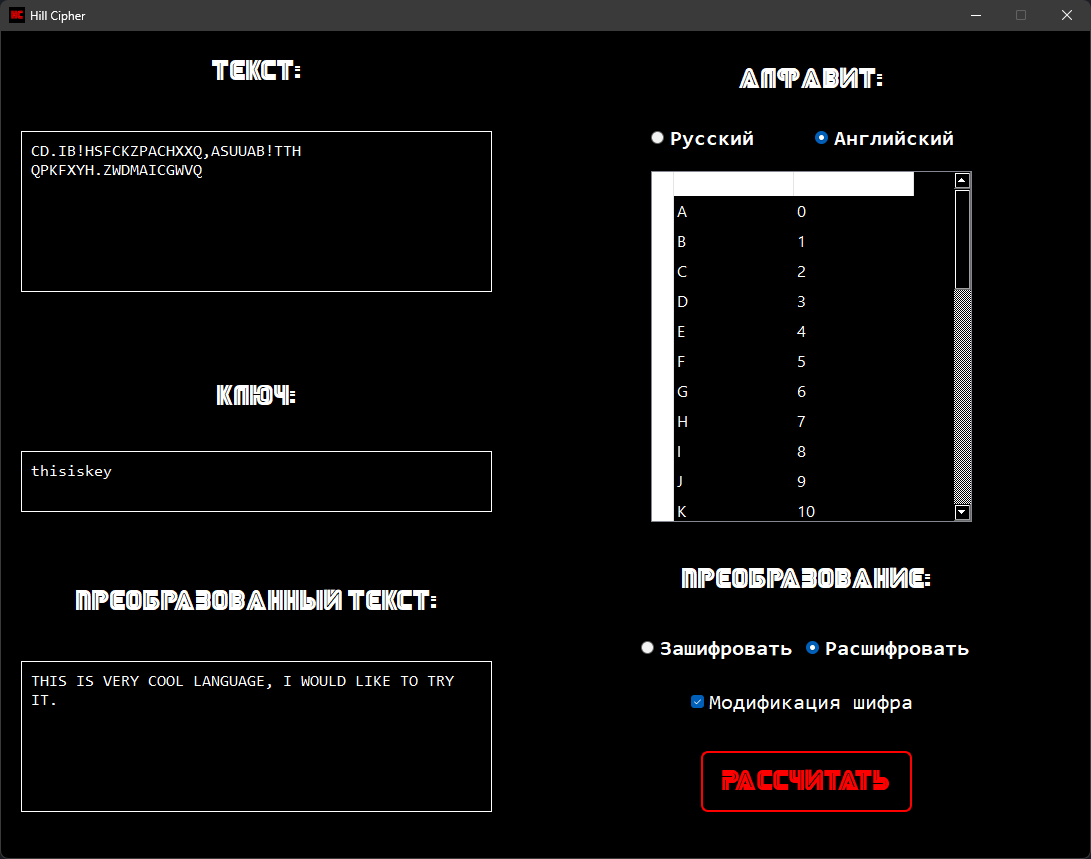


Рисунок 13 – Дешифрование на английском языке

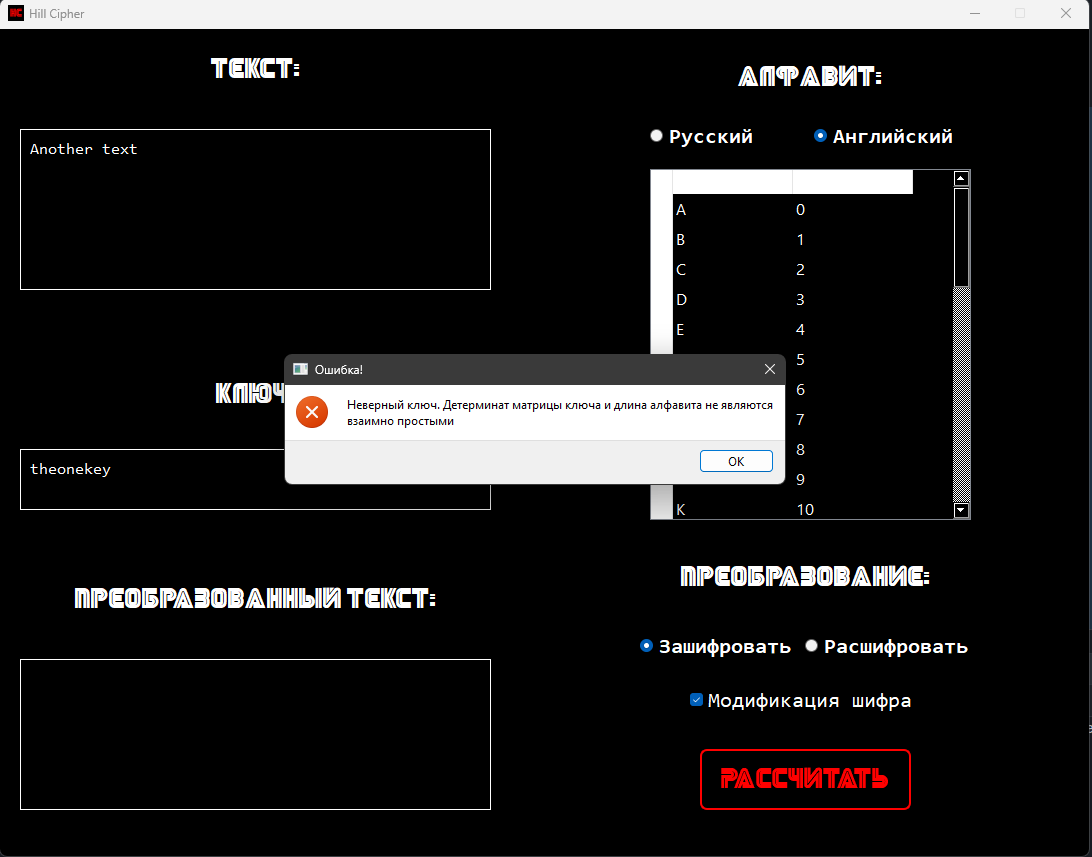


Рисунок 14 – Пример ошибки

## **5.6 Доказательство повышения криптостойкости**

Доказательством того, что предложенная модификация повышает криптостойкость шифра Хилла станет устранение уязвимости исходного шифра, связанной с атакой по парам открытого и зашифрованного текста.

Предположим у нас есть исходный текст: солнце сегодня встало рано, и зашифрованный текст: ЗЭДЕРТИАРЕЫЁУСВЦТО.ЬЭЙВИБР .

Ключ, который мы не знаем, имеет следующую матрицу:

(1)

Попробуем вычислить ключ, тем самым воспользоваться уязвимостью не модифицированного шифра Хилла. Предположим, что мы перехватали 2 пары открытое сообщение (солн) – зашифрованное сообщение (зэде).

Это будут соответствующие матрицы, согласно нашему алфавиту:

(2)

Составим матрицы P и C из этих пар

(3)

(4)

Обратная матрица для P.

(4)

Умножим матрицу C на получившуюся матрицу и найдём ключ.

(5)

Теперь мы можем расшифровать любое перехваченное сообщение. Это является главной уязвимостью в исходном шифре.

Если мы применим модифицированный шифр Хилла, то получим следующие пары.

(6)

Составим матрицы P и C из этих пар.

(7)

(8)

Обратная матрица для P.

(9)

Умножим матрицу C на получившуюся матрицу попробуем найти ключ.

(10)

Мы получили неверный ключ, поэтому дальнейшие вычисления и расшифровка бессмысленны. Уязвимость исходного шифра была закрыта. Повышение криптостойкости при использовании модификации было доказано.

## **6 Выводы**

В ходе выполнения практической работы были изучены виды шифров и принципы их работы. Была разработана программа, которая осуществляет шифрование и дешифрование текста с помощью шифра Хилла.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТУ 7.5-07-2021 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 20.12.2021.