**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: **«Основы защиты информации»**

на тему: **Шифрование и дешифрование на основе классических криптосистем**

Выполнил: студент гр. ИТП-41

Солодков М.А

Принял: ст. преподаватель

Титова Л.К.

Гомель 2020

**Цель**: освоить основные понятия криптологии, научиться применять простейшие криптосистемы.

**Ход работы**

**Вариант 13**

**Задание:** разработать приложение, выполняющее шифрование и дешифрование текста в соответствии с заданным алгоритмом (табл. 1.1).

Протестировать работу приложения на не менее чем трех текстах от 10 до 20 символов.

Таблица 1.1 – Задание

|  |  |
| --- | --- |
| Криптографическая система | Вариант |
| **Задание 1** | |
| Шифр Цезаря (шаг сдвига соответствует варианту). | Четные варианты |
| Цифровая система тайнописи (соответствие цифр и букв выбрать самостоятельно). | Нечетные варианты |
| **Задание 2** | |
| Квадрат Полибия (для латинского алфавита) | 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31 |
| Квадрат Полибия (для русского алфавита) | 2, 5, 8, 11, 14, 17, 20, 23, 26, 29 |
| Магический квадрат | 3, 6, 9, 12, 15, 18, 21, 24, 27, 30 |

**Результат работы программы:**

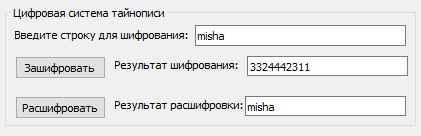


Рисунок 1 – Шифрование и дешифрование с помощью цифровой системой тайнописи

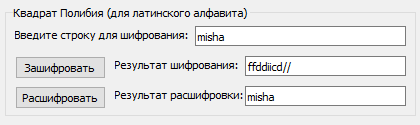


Рисунок 2 – Шифрование и дешифрование с помощью квадрата Полибия

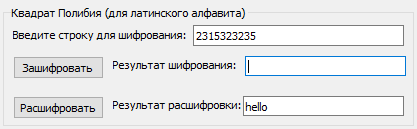


Рисунок 3 – Тест правильности шифрования методом квадрат Полибия

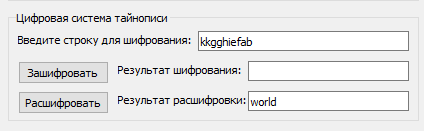


Рисунок 4 – Тест правильности шифрования методом цифровой системы тайнописи

**Вывод:** в результате выполнения лабораторной работы были изучены основные понятия криптологии, а также применения простейших криптосистем.

**Приложение А**

**Исходный код программы**

**Main.py**

import sys

from math import floor

from PyQt5 import QtWidgets

from layout import Ui\_MainWindow

class window(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

super(window, self).\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.encoder = None

self.digitalCryptographySystem = None

self.ui.pushButton.clicked.connect(lambda: self.polybius\_code())

self.ui.pushButton\_2.clicked.connect(lambda: self.polybius\_decode())

self.ui.pushButton\_3.clicked.connect(lambda: self.digitalCryptographySystem\_code())

self.ui.pushButton\_4.clicked.connect(lambda: self.digitalCryptographySystem\_decode())

def polybius\_code(self):

self.encoder = Encoder(self.ui.lineEdit.text())

self.ui.lineEdit\_3.setText(self.encoder.code())

def polybius\_decode(self):

self.encoder.decode()

self.ui.lineEdit\_2.setText(self.encoder.decode\_text)

def digitalCryptographySystem\_code(self):

self.digitalCryptographySystem = DigitalCryptographySystem(self.ui.lineEdit\_4.text())

self.digitalCryptographySystem.code()

self.ui.lineEdit\_5.setText(self.digitalCryptographySystem.encode\_text)

def digitalCryptographySystem\_decode(self):

self.digitalCryptographySystem.decode()

self.ui.lineEdit\_6.setText(self.digitalCryptographySystem.decode\_text)

class Encoder:

def \_\_init\_\_(self, text):

self.text = text

self.encode\_text = ""

self.decode\_text = ""

self.polybius\_dictionary = {

'a': '11', 'b': '12', 'c': '13', 'd': '14', 'e': '15',

'f': '21', 'g': '22', 'h': '23', 'i': '24', 'j': '25',

'k': '31', 'l': '32', 'm': '33', 'n': '34', 'o': '35',

'p': '41', 'q': '42', 'r': '43', 's': '44', 't': '45',

'u': '51', 'v': '52', 'w': '53', 'x': '54', 'y': '55',

'z': '61'

}

def set\_text(self, text):

self.text = ""

self.text = text

def code(self):

self.encode\_text = ""

list\_text = list(self.text.lower())

for char in list\_text:

if char in self.polybius\_dictionary:

self.encode\_text += self.polybius\_dictionary.get(char)

else:

self.encode\_text += (char + char)

return self.encode\_text

def decode(self):

new\_text = ""

list\_text = []

step = 2

for i in range(0, len(self.encode\_text), 2):

list\_text.append(self.encode\_text[i:step])

step += 2

keys\_polybius\_dictionary = list(self.polybius\_dictionary.keys())

values\_polybius\_dictionary = list(self.polybius\_dictionary.values())

for char in list\_text:

if char in values\_polybius\_dictionary:

i = values\_polybius\_dictionary.index(char)

new\_text += keys\_polybius\_dictionary[i]

else:

new\_text += char[0:1]

self.decode\_text = new\_text

class DigitalCryptographySystem:

def \_\_init\_\_(self, text):

self.text = text

self.encode\_text = ""

self.decode\_text = ""

self.dictionary = {

'a': '2', 'b': '3', 'c': '4', 'd': '5', 'e': '6',

'f': '7', 'g': '8', 'h': '9', 'i': '10', 'j': '11',

'k': '12', 'l': '13', 'm': '14', 'n': '15', 'o': '16',

'p': '17', 'q': '18', 'r': '19', 's': '20', 't': '21',

'u': '22', 'v': '23', 'w': '24', 'x': '25', 'y': '26',

'z': '27', '/': '1', '\*': '0'

}

def set\_text(self, text):

self.text = ""

self.text = text

def code(self):

text\_list = list(self.text.lower())

keys\_dictionary = list(self.dictionary.keys()) #letter

values\_dictionary = list(self.dictionary.values()) #numbers

for char in text\_list:

if char in keys\_dictionary:

value = int(self.dictionary[char])

if value % 2 == 0:

first\_code\_value = str(int(value / 2))

second\_code\_value = str(int(value / 2))

else:

first\_code\_value = str(floor(value / 2))

second\_code\_value = str(floor((value / 2)) + 1)

index\_first\_code\_value = values\_dictionary.index(first\_code\_value)

index\_second\_code\_value = values\_dictionary.index(second\_code\_value)

self.encode\_text += keys\_dictionary[index\_first\_code\_value] + keys\_dictionary[

index\_second\_code\_value]

return self.encode\_text

def decode(self):

new\_text = []

step = 2

for i in range(0, len(self.encode\_text), 2):

new\_text.append(self.encode\_text[i:step])

step += 2

keys\_dictionary = list(self.dictionary.keys()) # letter

values\_dictionary = list(self.dictionary.values()) # numbers

for char in new\_text:

if char != ' ':

first\_char = list(char)[0]

second\_char = list(char)[1]

number\_value\_first\_char = int(self.dictionary[first\_char])

number\_value\_second\_char = int(self.dictionary[second\_char])

res\_number\_value = str(number\_value\_first\_char + number\_value\_second\_char)

index\_char = values\_dictionary.index(res\_number\_value)

self.decode\_text += keys\_dictionary[index\_char]

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication([])

application = window()

application.show()

sys.exit(app.exec())