Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Лабораторная работа №1

по дисциплине “Информационная безопасность”

на тему “Основы шифрования данных”

Вариант 3

Выполнила: Студентка гр. P3402

Калугина М. М.

Преподаватель: к.т.н, доцент

Маркина Т.А.

г. Санкт-Петербург

2020 г.

Криптографические системы с секретным ключом

Лабораторная работа № 1

Основы шифрования данных

Вариант

3

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Задание

Реализовать шифрование и дешифрацию файла с использованием метода биграмм. Ключевое слово вводится.

Описание

В начале XVI века аббат из Германии Иоганн Трисемус предложил шифровать по две буквы за раз. Шифры, использующие подобный принцип, были названы биграммными. Открытый текст разбивался на пары букв (биграммы) и текст шифровки строился из него по следующим правилам.

1. Если обе буквы биграммы исходного текста принадлежат одной колонке таблицы, то буквами шифра считаются буквы, которые лежат под ними. Так биграмма ИН дает текст шифровки НЗ. Если буква открытого текста находится в нижнем ряду, то для шифра берется соответствующая буква из верхнего ряда и биграмма НЯ дает шифр ЗИ. (Биграмма из одной буквы или пары одинаковых букв тоже подчиняется этому правилу, и текст ОО дает шифр ГГ).

2. Если обе буквы биграммы исходного текста принадлежат одной строке таблицы, то буквами шифра считаются буквы, которые лежат справа от них. Так биграмма АБ дает текст шифровки НГ. Если буква открытого текста находится в крайней правой колонке, то для шифра берется буква из крайней левой колонки той же строки и биграмма АД дает шифр НА.
3. Если обе буквы биграммы открытого текста лежат в разных рядах и колонках, то вместо них берутся такие две буквы, чтобы вся их четверка представляла прямоугольник. Например, биграмма ЕК шифруется как БЙ (КЕ зашифруется ЙБ).

Заполнение квадрата алфавитом может быть случайным, а может определяться некоторой ключевой фразой, все символы которой (но без повторений) записываются в начале матрицы, а затем по порядку выписываются остальные буквы алфавита.

ключевая фраза – ‘шифрование’

Ш	И	Ф	Р	О	В
А	Н	Е	Б	Г	Д
Ж	З	Й	К	Л	М
П	С	Т	У	Х	Ц
Ч	Щ	Ь	Ы	Ъ	Э
Ю	Я		.	,	-

открытый текст: МЕТОД БИГРАММ.

шифротекст: ЙДХФЕ-НРБОДЖК-

В [30]:

```
import re

lexemes = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ .,-"
```

Ввод ключевого слова

В [80]:

```
# получение ключевого слова
print("Введите ключевое слово кириллицей.")
keyword = input().upper()
temp_size = len(keyword)
print(f'Введенное ключевое слово: "{keyword}"')

# удаление из ключевого слова недопустимых символов
keyword = ''.join(re.findall('[А-Я ,.-]', keyword))
if temp_size != len(keyword):
    print(f'Из ключевой фразы были убраны недопустимые символы. Итоговая фраза: "{keyword}"
```

```
Введите ключевое слово кириллицей.
шахматный этюд
Введенное ключевое слово: "ШАХМАТНЫЙ ЭТЮД"
```

Создание квадрата для шифрования

В [93]:

```
# объявление переменных
encryption_word = ""
encryption_square = [[], [], [], [], [], []]

# удаление дублирующихся символов в строке
temp_size = len(keyword)
keyword = ''.join(sorted(set(keyword), key=keyword.index))

# вывод результата в консоль
if temp_size != len(keyword):
    print("Из ключевой фразы были убраны повторяющиеся символы:")
if len(keyword) != 0:
    print(f'Создание слоавря из ключевой фразы: {keyword}')
else:
    print(f'Введена пустая ключевая фраза')

# создание упорядоченного сета для заполнения квадрата алфовитом.
temp_string = keyword + lexemes
encryption_word = ''.join(sorted(set(temp_string), key=temp_string.index))

# создания квадрата для расшифровки
print("\nСЛОВАРЬ:")
for i in range(0, 6):
    for j in range(0, 6):
        encryption_square[i].append(encryption_word[i*6+j])
    print(encryption_square[i])
```

```
Создание слоавря из ключевой фразы: ШАХМТНЫЙ ЭЮД

СЛОВАРЬ:
['Ш', 'А', 'Х', 'М', 'Т', 'Н']
['Ы', 'Й', ' ', 'Э', 'Ю', 'Д']
['Б', 'В', 'Г', 'Е', 'Ё', 'Ж']
['З', 'И', 'К', 'Л', 'О', 'П']
['Р', 'С', 'У', 'Ф', 'Ц', 'Ч']
['Щ', 'Ъ', 'Ь', 'Я', '.', ',']
```

Шифрование

В [142]:

```
# получение слова
print("Введите фразу для шифрования кириллицей.")
phrase = input().upper()
temp_size = len(phrase)
print(f'Введенная фраза: "{phrase}"')

# удаление из ключевого слова недопустимых символов
phrase = ''.join(re.findall('[А-Я ,.-]', phrase))
if temp_size != len(phrase):
    print(f'Из фразы были убраны недопустимые символы. Итоговая фраза: "{phrase}"')

# если в фразе нечетное количество букв и его невозможно разбить на пары, то добавим пробел
if len(phrase) % 2 == 1:
    phrase = phrase + " "


# функция для поиска позиции буквы в квадрате
# возвращается i, j - строка и столбец найденной буквы
def find_i_j(k):
    for i1 in range(0, 6):
        try:
            j1 = encryption_square[i1].index(k)
            return i1, j1
        except:
            pass


# обработка фразы
encrypted_phrase = ""
for i in range(0, len(phrase), 2):
    # определение местоположения символа
    i1, j1 = find_i_j(phrase[i])
    i2, j2 = find_i_j(phrase[i+1])

    # если оба символа биграмы принадлежат одной колонке (или если это пара одинановых симв
    # --  то буквами шифра считаются буквы, которые лежат под ними.
    if (j1 == j2):
        encrypted_phrase = encrypted_phrase + encryption_square[i1 + 1 if i1 < 5 else 0][j1
        encrypted_phrase = encrypted_phrase + encryption_square[i2 + 1 if i2 < 5 else 0][j2

    # Если обе буквы биграммы исходного текста принадлежат одной строке таблицы,
    # то буквами шифра считаются буквы, которые лежат справа от них
    elif (i1 == i2):
        encrypted_phrase = encrypted_phrase + encryption_square[i1][j1 + 1 if j1 < 5 else 0
        encrypted_phrase = encrypted_phrase + encryption_square[i2][j2 + 1 if j2 < 5 else 0

    # Если обе буквы биграммы открытого текста лежат в разных рядах и колонках,
    # то вместо них берутся такие две буквы, чтобы вся их четверка представляла прямоугольн
    else:
        encrypted_phrase = encrypted_phrase + encryption_square[i1][j2]
        encrypted_phrase = encrypted_phrase + encryption_square[i2][j1]

    print(f'{phrase[i]}{phrase[i+1]}->{encrypted_phrase[i]}{encrypted_phrase[i+1]}')

print(f'Зашифрованная фраза: {encrypted_phrase}')
```

```
Введите фразу для шифрования кириллицей.
```

```
Запись и забаррикадируйся шкафом от хроноса, космоса, эроса, расы, вируса.
Введенная фраза: "ЗАПРИСЬ И ЗАБАРРИКАДИРУЙСЯ ШКАФОМ ОТ ХРОНОСА, КОСМОСА, ЭРО
СА, РАСЫ, ВИРУСА."
ЗА->ИШ
ПР->ЗЧ
ИС->СЪ
Ь ->ХГ
И ->КЙ
ЗА->ИШ
БА->ВШ
РР->ЩЩ
ИК->КЛ
АД->НЙ
ИР->ЗС
УЙ->С
СЯ->ФЪ
 Ш->ЫХ
КА->ИХ
ФО->ЦЛ
М ->ХЭ
ОТ->ЦЮ
 Х->Г
РО->ЦЗ
НО->ТП
СА->ЪЙ
, ->ЬД
КО->ЛП
СМ->ФА
ОС->ИЦ
А,->НЪ
 Э->ЭЮ
РО->ЦЗ
СА->ЪЙ
, ->ЬД
РА->СШ
СЫ->РЙ
, ->ЬД
ВИ->ИС
РУ->СФ
СА->ЪЙ
. ->ЬЮ
Зашифрованная фраза: ИШЗЧСЪХГКЙИШВШЩЩКЛНЙЗСС ФЪЫХИХЦЛХЭЦЮГ ЦЗТПЪЙЬДЛПФАИЦНЪЭ
ЮЦЗЪЙЬДСШРЙЬДИССФЪЙЬЮ
```

Дешифрование фразы

В [143]:

```
# получение слова
print("Введите фразу для дешифрования кириллицей.")
phrase = input().upper()
temp_size = len(phrase)
print(f'Введенная фраза: "{phrase}"')

# удаление из ключевого слова недопустимых символов
phrase = ''.join(re.findall('[А-Я ,.-]', phrase))
if temp_size != len(phrase) or temp_size % 2 == 1:
    raise Exception(f'В зашифрованной фразе встречены недопустимые символы или количество с

decrypted_phrase = ""
for i in range(0, len(phrase), 2):
    # определение местоположения символа
    i1, j1 = find_i_j(phrase[i])
    i2, j2 = find_i_j(phrase[i+1])

    # если оба символа биграмы принадлежат одной колонке (или если это пара одинановых симв
    # --  то буквами шифра считаются буквы, которые лежат под ними.
    # значит, для дешиврования -- буквы, которые лежат над
    if (j1 == j2):
        decrypted_phrase = decrypted_phrase + encryption_square[i1 - 1 if i1 > 0 else 5][j1
        decrypted_phrase = decrypted_phrase + encryption_square[i2 - 1 if i2 > 0 else 5][j2

    # Если обе буквы биграммы исходного текста принадлежат одной строке таблицы,
    # то буквами шифра считаются буквы, которые лежат справа от них
    # значит, для дешиврования -- буквы, которые лежат слева
    elif (i1 == i2):
        decrypted_phrase = decrypted_phrase + encryption_square[i1][j1 - 1 if j1 > 0 else 5
        decrypted_phrase = decrypted_phrase + encryption_square[i2][j2 - 1 if j2 > 0 else 5

    # Если обе буквы биграммы открытого текста лежат в разных рядах и колонках,
    # то вместо них берутся такие две буквы, чтобы вся их четверка представляла прямоугольн
    else:
        decrypted_phrase = decrypted_phrase + encryption_square[i1][j2]
        decrypted_phrase = decrypted_phrase + encryption_square[i2][j1]

    print(f'{phrase[i]}{phrase[i+1]}->{decrypted_phrase[i]}{decrypted_phrase[i+1]}')

print(f'Дешифрованная фраза: {decrypted_phrase}')
```

```
Введите фразу для дешифрования кириллицей.
ИШЗЧСЪХГКЙИШВШЩЩКЛНЙЗСС ФЪЫХИХЦЛХЭЦЮГ ЦЗТПЪЙЬДЛПФАИЦНЪЭЮЦЗЪЙЬДСШРЙЬДИССФЪЙЬЮ
Введенная фраза: "ИШЗЧСЪХГКЙИШВШЩЩКЛНЙЗСС ФЪЫХИХЦЛХЭЦЮГ ЦЗТПЪЙЬДЛПФАИЦНЪЭЮЦЗ
ЪЙЬДСШРЙЬДИССФЪЙЬЮ"
ИШ->ЗА
ЗЧ->ПР
СЪ->ИС
ХГ->Ь
КЙ->И
ИШ->ЗА
ВШ->БА
ЩЩ->РР
КЛ->ИК
НЙ->АД
ЗС->ИР
```

```
С ->УЙ
ФЪ->СЯ
ЫХ-> Ш
ИХ->КА
ЦЛ->ФО
ХЭ->М
ЦЮ->ОТ
Г -> Х
ЦЗ->РО
ТП->НО
ЪЙ->СА
ЬД->,
ЛП->КО
ФА->СМ
ИЦ->ОС
НЪ->А,
ЭЮ-> Э
ЦЗ->РО
ЪЙ->СА
ЬД->,
СШ->РА
РЙ->СЫ
ЬД->,
ИС->ВИ
СФ->РУ
ЪЙ->СА
ЫЮ->.
Дешифрованная фраза: ЗАПРИСЬ И ЗАБАРРИКАДИРУЙСЯ ШКАФОМ ОТ ХРОНОСА, КОСМОСА,
ЭРОСА, РАСЫ, ВИРУСА.
```

Вывод

В ходе выполнения лабораторной работы был изучен метод шифрование и дешифрацию файла с использованием метода биграмм. Была реализована программа для шиврования и дешифрования текста и использованием данного метода и создания квадрата для шифрования при помощи введенного ключевого слова.