Лабораторная работа 1

Тема:

“Метод рассечения-разнесения данных”

Выполнил:

студент группы 22207 Гордеев Никита

Задание:

- Составить программу, которая будет выполнять процедуру рассечения-разнесения и обратную операцию — сборку, для строки текста произвольной длины, содержащего символы переноса строки.
- Разделяемыми элементами являются символы.

Значения параметров разбиения (Вариант 1):

- Количество блоков: 12
- Количество столбцов: 4
- Ключ для столбцов: {3-2-4-1}
- Ключ для строк: 12 / 4 → 3 {3-1-2}

О методе:

- Общепринятой практикой в области облачных технологий является хранение информации (данных) у одного поставщика услуг в одном физическом и географическом местоположении. При этом можно задействовать внушительный набор инструментов по организационным и техническим мерам обеспечения защиты информации в «облаке», что, как правило, сводит на нет преимущества облачного хранилища. Поэтому ценная критически важная информация обязательно шифруется перед её отправкой в «облако» на стороне пользователя. Для этого можно использовать средства как простые (например, создание архивов, защищенных паролем), так и более продвинутые (ПО для создания шифрованных разделов). Для пользователя помимо обычного кодирования своей информации, существуют методы разделения информации, которые позволяют разделить их сообщение и отправить его по разным внешним хранилищам, что существенно лучше, чем просто отправить оригинал с резервными копиями по разным хранилищам. К таким методам можно отнести, например, метод рассечения-разнесения
- Рассмотренный же в данной статье метод рассечения разнесения позволяет разбить исходный файл на несколько частей и каждую хранить в разных местах хранения и если злоумышленник сможет добыть одну часть и успешно взломать её, то достичь цели крипто атаки всё равно не сможет.
- Набор защищаемых данных разбивается на блоки, которые разносятся по нескольким другим наборам данных.
- Каждый отдельный блок не несет сколько-нибудь значимой информации, и даже доступ к полной совокупности блоков не позволяет легко восстановить исходный набор данных без знания способа разбиения.

Решение:

1) Решение на бумаге

Ключи	3	2	4	1
3	М	Е	Т	О
1	Д	я	Р	А
2	С	С	Е	Ч
3	Е	Н	И	Я
1	-	Р	А	З
2	Н	Е	С	Е
3	Н	И	Я	.

n
4

Номер блока	Содержимое
1	АЗ
2	яР
3	Д-
4	РА
5	ЧЕ
6	СЕ
7	СН
8	ЕС
9	ОЯ.
10	ЕНИ
11	МЕН
12	ТИЯ

К	11	10	12	9
	3	2	4	1
	7	6	8	5
	11	10	12	9
	3	2	4	1
	7	6	8	5
	11	10	12	9

2) Алгоритм программы:

1. Получение сведений о программе

1.1. Получение сообщения

```
10  print('\n' + "2) Введите сообщение без пробелов, например: МЕТОДяРАССЕЧЕНИЯ-РАЗНЕСЕНИЯ.")
11  message = "МЕТОДяРАССЕЧЕНИЯ-РАЗНЕСЕНИЯ."
12  message = FillMessage(message)
13
```

1.1.1.Проверка на корректность введённых данных

```
25  # Дозаполнение сообщения пробелами до размера кратного 4м
26  def FillMessage(input_message):
27      if (len(input_message) % 4 == 0):
28          return input_message
29      else:
30          input_message += ' ' * (4 - len(input_message) % 4)
31          return input_message
```

1.2. Получение ключей столбцов и строк

```
14  print('\n'+ "3) Введите ключи 4х столбцов в произвольной комбинации, например: [3, 2, 4, 1]")
15  column_key = [3, 2, 4, 1]
16  CheckListValues(column_key, 4)
17
18  print('\n' + "4) Введите ключи 3х строк, например: [3, 1, 2]")
19  string_key = [3, 1, 2]
20  CheckListValues(string_key, 3)
```

1.2.1.Проверка на корректность введённых данных

```
1   # Импорт библиотек
2   from email import message
3   import sys
4
5   # Проверка, что значение не привосходит размер
6   def CheckLess(string_key, limit):
7       return (all(x <= limit for x in string_key))
8
9   # Проверка уникальности значений списка
10  def CheckUniqueObjects(input_list):
11      unique_input_set = set(input_list)
12      unique_out_list = list(unique_input_set)
13      if (len(unique_out_list) == len(input_list)):
14          return True
15      else:
16          return False
17
18  # Комплексная проверка значений списка
19  def CheckListValues(check_list, limit):
20      if (type(check_list) == list and len(check_list) == limit and CheckLess(check_list, limit) and CheckUniqueObjects(check_list)):
21          print("Ключи введены верно")
22      else:
23          sys.exit("Ключи введены не верно")
24
```

1.3. Получение количества строк и столбцов

```
# Обработка входных данных
message_length = len(message)
number_blocks = 12
number_column = 4
number_string = int(number_blocks/number_column)

matrix_column = number_column+1
matrix_string = int(message_length//number_column)+1
```

2. Шифрование сообщения

2.1. Создание нулевой матрицы с заголовками

```
# Создание нулевой матрицы с заголовками
matrix = [[0 for x in range(matrix_column)] for y in range(matrix_string)]
matrix[0] = [0] + column_key
j = 0  # позиция в ключах по строкам
for i in range(1, matrix_string):
    matrix[i][0] = string_key[j]
    j += 1
    if j >= len(string_key):
        j = 0
```

2.2. Заполнение матрицы буквами и сохранение позиций букв

```
# Определение позиции для заполнения матрицы
def DefinePosition(i, division):
    return (i//division + 1)
```

```
# заполение матрицы буквами и сохранение позиций букв
blocks = []
m = 0
for i in range(1, matrix_string):
    for j in range(1, matrix_column):
        matrix[i][j] = message[m]
        add_column = matrix[0][j]
        add_string = matrix[i][0]
        position = DefinePosition(i-1, number_string)
        blocks.append([message[m], add_string, add_column, position])
        m += 1
```

2.3. Создание матрицы блоков

```
# Функция шифрование матрицы в блоки
def EncryptSeparationDissectionMethod(matrix, number_matrix_column, number_matrix_blocks, matrix_string, matrix_column):
    # Шаблон для формирования блоков
    blocks = [[] for i in range(number_matrix_blocks)]
```

```
# Шифрование сообщения в блоки
print("Сформированные, зашифрованные блоки:")
cipher = EncryptSeparationDissectionMethod(matrix, number_column, number_blocks, matrix_string, matrix_column)
```

2.4. Распределение букв из матрицы в блоки

```
# Выбор номера блока, согласно выражению
def DefineBlock(r, s, n):
    K = n * (r-1) + s
    return K
```

```
    # Формирование блоков
    for i in range(1, matrix_string):
        for j in range(1, matrix_column):
            position_column = matrix[0][j]
            position_string = matrix[i][0]
            position = DefineBlock(position_string, position_column, number_matrix_column)
            blocks[position-1].append(matrix[i][j])
```

2.5. Возврат блоков

```
# Печать блоков на консоль
def OutputBlocksToConsole(blocks):
    i = 1
    for position in blocks:
        print("блок " + str(i) + " содержимое", position)
        i += 1
```

```
    OutputBlocksToConsole(blocks)
    return blocks
```

3. Расшифрование блоков

3.1. Создание матрицы с заголовками

```
# Функция расшифровки блоков в матрицу
def Decrypt(string_key, column_key, cipher, blocks, matrix_string, matrix_column):
    # Суммарное количество букв в блоках
    letters_in_message = sum(sum(1 for i in string if i) for string in cipher)

    # Задание размеров расшфированной матрицы
    matrix_column = len(column_key)+1
    matrix_string = int(letters_in_message/len(column_key))+1

    # Создание нулевой матрицы с заголовками
    decoded_matrix = [[0 for x in range(matrix_column)] for y in range(matrix_string
    decoded_matrix[0] = [0] + column_key # названия колонок
    j = 0 # позиция в ключах по строкам
    for i in range(1, matrix_string):
        decoded_matrix[i][0] = string_key[j]
        j += 1
        if j >= len(string_key):
            j = 0
```

3.2. Распределение букв в блоках по ячейкам матрицы

```
# Получение строки ключей
def StringKeys(matrix_rows):
    string = []
    for x in matrix_rows:
        string.append(x[0])
    return string
```

```
    # Сопоставление блока ячейке в матрице
    string_keys = StringKeys(decoded_matrix)
    for i in range(len(blocks)):
        column_index = decoded_matrix[0].index(blocks[i][2])
        index = [m for m, x in enumerate(string_keys) if x == blocks[i][1]]
        string_index = index[blocks[i][3]-1]

        # Добавление элемента в расшифрованную матрицу
        decoded_matrix[string_index][column_index] = blocks[i][0]
```

3.3. Печать расшифрованной матрицы на консоль

```
# Печать блоков на консоль
def OutputMatrixToConsole(matrix):
    for string in matrix:
        print(string)
```

3.4. Сборка строки обратного сообщения

```
# Генерация исходного сообщения
decrypted = DecryptedMessage(matrix_string, matrix_column, decoded_matrix)
return decrypted
```

```
# Сборка расшифрованного сообщения
def DecryptedMessage(matrix_string, matrix_column, decoded_matrix):
    decrypted = ''
    for i in range(1, matrix_string):
        for j in range(1, matrix_column):
            decrypted += decoded_matrix[i][j]
    return decrypted
```

3.5. Вывод на консоль пользователя

```
# Расшифрованное сообщение сообщения в блоки
print('\n' + "Сформированная, расшифрованная таблица:")
decrypted = Decrypt(string_key, column_key, cipher, blocks, matrix_string, matrix_column)
print('\n' + "Расшифрованное сообщение:", decrypted)
```

Материалы:

- ЗАЩИТА ИНФОРМАЦИИ ВО ВНЕШНИХ ХРАНИЛИЩАХ ДАННЫХ МЕТОДОМ РАССЕЧЕНИЯ-РАЗНЕСЕНИЯ // ЭЛЕКТРОННЫЙ НАУЧНЫЙ ЖУРНАЛ "МОЛОДАЯ НАУКА СИБИРИ" URL: https://mnv.irgups.ru/sites/default/files/articles_pdf_files/protection_of_information_in_external_data_stores_0.pdf (дата обращения: 12.10.2022).
- Python: проверьте, является ли переменная списком // DevGang URL: https://dev-gang.ru/article/python-proverte-javljaetsja-li-peremennaja-spiskom-jgs9sxwhc2/ (дата обращения: 12.10.2022).

- Get unique values from a list in python [duplicate] // stack overflow URL: https://stackoverflow.com/questions/12897374/get-unique-values-from-a-list-in-python) (дата обращения: 13.10.2022).