Отчёт по лабораторной работе №7

Дисциплина: Информационная безопасность

Евдокимов Иван Андреевич

Содержание

Техническое оснащение: **5**

Цель работы: **6**

Постановка задачи **7**

Код программы **8**

Список литературы **13**

Список иллюстраций

1 шифровка и дишифровка текста 11
2 шифровка фрагмента текста 11
3 шифровка и дишифровка текста на английском 11
4 шифровка фрагмента текста на английском 11

Список таблиц

Техническое оснащение:

- Персональный компьютер с операционной системой Windows 10;
- OBS Studio, использующийся для записи скринкаста лабораторной работы;
- Приложение Visual Studio Code для редактирования файлов формата *md*, а также для конвертации файлов отчётов и презентаций;

Цель работы:

Освоить на практике применение режима однократного гаммирования.

Постановка задачи

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

Код программы

```
# Импортируем модули для работы со строками и для генерации случайных чисел
import random
import string


# Создаем класс для кодирования и декодирования текста
class TextEncoding:

    @staticmethod
    # Метод для определения алфавита, который следует использовать для генерации
    def determine_alphabet(text):
        # Если первый символ текста в английском алфавите (в нижнем регистре),
        # то возвращаем английский алфавит и цифры
        if text[0] in string.ascii_lowercase:
            return string.ascii_lowercase + string.digits
        else:
            # В противном случае возвращаем русский алфавит и цифры
            return "абвгдеёжзийклмнопрстуфхцчшщъыьэюя" + string.digits

    @staticmethod
    # Метод для генерации ключа
    # Ключ состоит из случайных символов алфавита (определенного в методе determi
```

```
def generate_key(size, alphabet):
    return "".join(random.choice(alphabet) for _ in range(size))

@staticmethod
# Метод для преобразования строки в шестнадцатеричный формат
# Каждый символ кодируется в шестнадцатеричную систему и объединяется в строк
def to_hex(coding):
    return " ".join(hex(ord(character))[2:] for character in coding)

@staticmethod
# Метод для кодирования строки
# Происходит применение операции XOR между кодами символов текста и ключа
def encode_string(text, key):
    return "".join(chr(ord(char) ^ ord(key_char)) for char, key_char in zip(t

@staticmethod
# Метод для поиска возможных ключей
# Принимает на вход строку текста и фрагмент этого текста
# Создает список возможных ключей, которые могут декодировать зашифрованный т
def find_possible_keys(text, fragment):
    key_length = len(fragment)
    possible_keys = []

    # Проходим по всему тексту с шагом, равным длине фрагмента
    for index in range(len(text) - key_length + 1):
        # Получаем возможный ключ путем применения операции XOR между очередн
        key = [chr(ord(char) ^ ord(key_char)) for char, key_char in zip(text[
        # Предполагаемый расшифрованный текст получаем путем кодирования заши
        presumed_plaintext = TextEncoding.encode_string(text, key)
```

```
            # Если известный фрагмент присутствует в предполагаемом расшифрованно
            if fragment in presumed_plaintext:
                possible_keys.append(''.join(key))

        return possible_keys

# Получаем от пользователя открытый текст
plaintext = input("Введите открытый текст: ")
# Определяем алфавит для генерации ключа
alphabet = TextEncoding.determine_alphabet(plaintext)
# Генерируем ключ
key = TextEncoding.generate_key(len(plaintext), alphabet)

# Выводим сгенерированный ключ и его шестнадцатеричное представление
print(f"Ключ: {key}", f"Ключ в 16 бит: {TextEncoding.to_hex(key)}", sep='\n')

# Кодируем открытый текст с помощью сгенерированного ключа
ciphertext = TextEncoding.encode_string(plaintext, key)
# Выводим зашифрованный текст и его шестнадцатеричное представление
print(f"Зашифрованный текст: {ciphertext}", f"Зашифрованный текст в 16 бит: {Text
      sep='\n')

# Декодируем зашифрованный текст с помощью сгенерированного ключа
decrypted_text = TextEncoding.encode_string(ciphertext, key)
# Выводим расшифрованный текст
print("Расшифрованный текст:", decrypted_text)

# Получаем от пользователя известный фрагмент открытого текста
```

```
known_fragment = input("Введите фрагмент открытого текста: ")
# Ищем возможные ключи для шифротекста
possible_keys = TextEncoding.find_possible_keys(ciphertext, known_fragment)
# Выводим найденные ключи
print("Возможные ключи для шифротекста:", possible_keys)
```

вывод запуска программы 1 (шифровка и дишифровка текста).

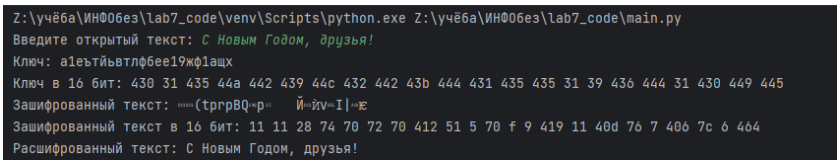

Рис. 1: шифровка и дишифровка текста

вывод запуска программы 2 (шифровка фрагмента текста).

Рис. 2: шифровка фрагмента текста

вывод запуска программы 3 на английском (шифровка и дишифровка текста на английском).

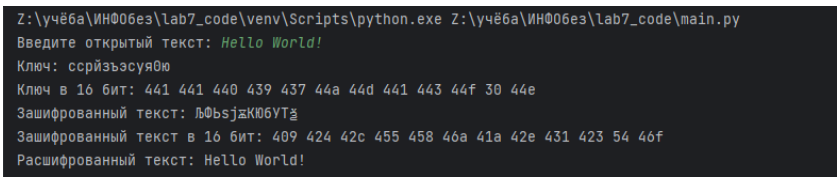

Рис. 3: шифровка и дишифровка текста на английском

вывод запуска программы 4 на английском (шифровка фрагмента текста на английском).

Рис. 4: шифровка фрагмента текста на английском

на видео к выполнения работы будут представленны более удачные варианты запуска программы

Выводы:

Мною были освоино на практике применение режима однократного гаммирования.

Список литературы

1. Официальный сайт VirtualBox
2. Материал для выполнения лабораторной
3. Официальный сайт CentOS