Отчёт по лабораторной работе №8

Попов Олег Павлович

Содержание

List of Tables

List of Figures

1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

2 Выполнение лабораторной работы

Перед реализацией новой функции в приложении нужно изучить теорию:

Доказательство абсолютной стойкости Шеннона [править | править код]

Клод Шеннон доказал, что при определённых свойствах гаммы этот метод шифрования является абсолютно стойким (то есть не поддающимся взлому).

Пусть X, Y и Z — дискретные случайные величины.

Пусть:

- X — значение бита открытого текста; то есть, переменная X (бит) способна принимать два значения: 0 и 1;
- p — вероятность события, заключающегося в том, что переменная X примет значение 0;
- $(1-p)$ — вероятность противоположного события (то есть, вероятность того, что переменная X примет значение 1).

Запишем закон распределения значений X:

X	0	1
Pi	p	$(1-p)$

Используем p и $(1-p)$, так как вероятность встретить одну букву в разных словах различна.

Пусть:

- Y — бит псевдослучайной последовательности (гаммы); то есть, переменная Y (бит) способна принимать два значения: 0 и 1;
- каждое из значений Y равновероятно; то есть, вероятности получить 0 или 1 равны 1/2.

Запишем закон распределения значений Y:

Y	0	1
Pi	$\frac{1}{2}$	$\frac{1}{2}$

Иными словами, в качестве гаммы (Y) подаётся одинаковое количество нулей и единиц, или значения переменной Y имеют симметричный закон распределения.

Пусть:

- Z — бит закрытого текста; то есть, переменная Z (бит) способна принимать два значения: 0 и 1;
- значение Z вычисляется на основе значений X и Y по формуле:
  $Z = X + Y \pmod 2$
  или
  Z = xor(X, Y)
  или
  $Z = X \oplus Y$

Найдём следующие вероятности:

- $P(Z=0)$ — вероятность события, заключающегося в том, что переменная Z принимает значение 0;
- $P(Z=1)$ — вероятность события, заключающегося в том, что переменная Z принимает значение 1.

Используем формулы:

- сложения вероятностей несовместных событий:
  $P(A+B)=P(A)+P(B)$;
- умножения вероятностей независимых событий:
  $P(A*B)=P(A)*P(B)$.

Вероятность того, что переменная Z примет значение 0:

$$P(Z=0)=P(X=0,Y=0)+P(X=1,Y=1)=P(X=0)*P(Y=0)+P(X=1)*P(Y=1)=p*1/2+(1-p)*1/2=1/2.$$

Вероятность того, что переменная Z примет значение 1:

$$P(Z=1)=1-P(Z=0)=1/2.$$

Так как $P(Z=0)$ и $P(Z=1)$ не зависят от p, p может принимать любое значение.

Запишем закон распределения значений переменной Z:

Y	0	1
Pi	$\frac{1}{2}$	$\frac{1}{2}$

Закон распределения Z оказался симметричным, как и закон распределения гамма (Y) или шум. То есть, Z не содержит никакую информацию из X (в Z нет p). Это доказывает, что шифр является абсолютно стойким.

В итоге можно сделать вывод, что расшифровать код невозможно без ключа или одного из исходных текстов сообщений. Значит, можно реализовать функцию, которая будет расшифровывать сообщение с помощью кодов всех сообщений и текста одного из сообщений.

Функция:

```
# расшифровка сообщения через коды и сообщение
def decryptWithoutKey(code1, code2, str1):
    xor_code = xor(code1, code2)
    str_code = createCode(str1)
    mes_code = xor(xor_code, str_code)
    mes = decryptCode(mes_code)
    return mes
```

Эта функция основана на функциях, написанных в седьмой лабораторной: сначала через xor получаем код, потом переводим сообщение в его исходный код, снова перемножаем коды через xor и дешфруем полученный в итоге код.

Код main:

```python
import codecreate
import numpy as np

mes1 = input("Введите первое сообщение: ")
mes2 = input("Введите второе сообщение: ")

key = codecreate.generateDecryptionCode(len(mes1))

code1 = np.array([])
code1 = codecreate.xor(codecreate.createCode(mes1), key)
print("XOR code 1:", code1, "\n")

code2 = np.array([])
code2 = codecreate.xor(codecreate.createCode(mes2), key)
print("XOR code 2:", code2, "\n")

print('Первое расшифрованное сообщение (если известно второе):', codecreate.decryptWithoutKey(code1, code2, mes2))
print('Второе расшифрованное сообщение (если известно первое):', codecreate.decryptWithoutKey(code1, code2, mes1))
```

Сначала на вход подаются две строки, генерируется ключ, строки шифруются через ключ и в конце через написанную функцию переводим строки из шифрованных кодов.

Вывод:

```
Введите первое сообщение: Привет, Антон!
Введите второе сообщение: Hello, Alex!!!
XOR code 1: ['0x449' '0x494' '0x470' '0x4c7' '0x4c3' '0x4a1' '0xda' '0x1a' '0x4a4'
 '0x4f2' '0x489' '0x48b' '0x4ab' '0x5a']

XOR code 2: ['0x1e' '0xb1' '0x24' '0x99' '0x99' '0xcf' '0xd6' '0x7b' '0xd8' '0xaa'
 '0xb3' '0x94' '0xb7' '0x5a']

Первое расшифрованное сообщение (если известно второе): Привет, Антон!
Второе расшифрованное сообщение (если известно первое): Hello, Alex!!!

Process finished with exit code 0
```

Как мы видим, из зашифрованных кодов и известного сообщения можно найти второе сообщение.

3 Выводы

В ходе данной работы я освоил на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.