Отчёт по лабораторной работе № 8

Информационная безопасность

Адебайо Ридвануллахи Айофе

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Теоретическое введение

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» (рис. 7.1) является простой, но надёжной схемой шифрования данных.

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование)той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть.Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком $\oplus$) между элементами гаммы и элементами подлежащего сокрытию текста. Напомним, как работает операция XOR над битами: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$.

Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное

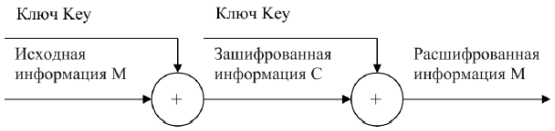

Рис. 2.1: Схема однократного использования Вернама

значение, а шифрование и расшифрование выполняется одной и той же программой.

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i,$$

где C_i—i-й символ получившегося зашифрованного послания, P_i—i-й символ открытого текста, K_i — i-й символ ключа, $i = 1, m$. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с {#fig:001}, а именно, обе части равенства необходимо сложить по модулю 2 с P_i:

$$C_i \oplus P_i = P_i \oplus K_i \oplus P_i = K_i,$$

$$K_i = C_i \oplus P_i,$$

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов.

К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фраг-

ментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P.

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.

3 Выполнение лабораторной работы

3.1 Задание

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочитать оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P_1 и P_2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C_1 и C_2 обоих текстов P_1 и P_2 при известном ключе ; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

3.2 Порядок выполнения работы

1. Определить функцию шифрования и дешифрования

```python
import random
from random import seed
import string
```

```python
def cipher_text_function(text, key):
    #condition to check if the key and text have the same length
    if len(key) != len(text):
        return "Key and text must have the same length"
    cipher_text = ""
    for i in range (len(key)):
        cipher_text_symbol=ord(text[i]) ^ ord(key[i])
        cipher_text+=chr(cipher_text_symbol)
    return cipher_text
```

Рис. 3.1: Функция шифрования

2. Определить вид шифротексты и ключа

In [4]:
```python
P1="НаВашисходящийот1204"
P2="ВСеверныйфилиалБанка"
```

In [5]:
```python
key=""
seed(23)
for i in range(len(P1)):
    key += random.choice(string.ascii_letters + string.digits)
print(key)
```

```
7X8s51fbLtByHwiUmrCa
```

Рис. 3.2: создание ключа той же длины, что и открытый текст

3. Вызов функции шифрования.

In [6]:
```python
cipher_P1 = cipher_text_function(P1, key)
cipher_P2 = cipher_text_function(P2, key)
print(f"Encoded P1: {cipher_P1}")
print(f"Encoded P2: {cipher_P2}")

```

```
Encoded P1: ЬЫЪуЌЉЧЧӨрЙаѰюї3\@sU
Encoded P2: ХѳЙсЄѱЋЩvaОтѰчЋфйяѳуё
```

Рис. 3.3: получение шифротекста

4. Пример обратного шифрования

```
n [7]:  1 print(f"P1: {cipher_text_function(cipher_P1, key)}")

P1: НаВашисходящийот1204

n [8]:  1 print(f"Key: {cipher_text_function(P1, cipher_P1)}")

Key: 7X8s51fbLtByHwiUmrCa

[10]:   1 print(f"P2: {cipher_text_function(cipher_P2, key)}")
        2 print(f"Key: {cipher_text_function(P2, cipher_P2)}")

P2: ВСеверныйфилиалБанка
Key: 7X8s51fbLtByHwiUmrCa
```

Рис. 3.4: обратный шифрование

4 Выводы

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

5 Список литературы

1. Кулябов Д. С. *Лабораторная работа №8**: 007-lab_crypto-key.pdf*