Лабораторная работа №8

Цель работы

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

Задание

Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P1 и P2 в режиме однократного гаммирования.

Приложение должно определить вид шифротекстов C1 и C2 обоих текстов P1 и P2 при известном ключе

Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

Теоретическое введение

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» 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. [1].

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.[2]

Гаммирование является симметричным алгоритмом. Поскольку двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, шифрование и дешифрование выполняется одной и той же программой [3].

Оборудование

Лабораторная работа выполнялась дома со следующими характеристиками техники:

– Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81GHz
– ОС Майкрософт Windows 10
– VirtualBox верс. 6.1.26

Выполнение лабораторной работы

1. Разработала приложение, позволяющее шифровать и дешифровать тексты P1 и P2 в режиме однократного гаммирования.

   (рис. -@fig:001)

Однократное гаммирование

Ввод [37]:
```
import random as rnd
import string as strg

p1 = "НаВашисходящийот1204"
p2 = "ВЮжныйфилиалБанка"
k = "".join(rnd.choice(strg.ascii_letters.join(strg.digits)) for i in range(len(p1)))
```

Ввод [38]:
```
#функциия сложение по модулю 2
def xor_string(data, key):
    return "".join(chr(ord(x)^ord(y)) for x, y in zip(data, key))

#функциия переобразования в байты
def butes_print(x, k):
    c = xor_string(x, k)
    print(bytes(c, "UTF-8").hex())
    return c
```

Ввод [39]:
```
c1 = butes_print(p1, k)
c2 = butes_print(p2, k)
```

```
d19bd193d185d1a9d09fd19cd0a9d092d19cd1b8d0a1d09fd1b3d18ed1abd0b45348045b
d194d18dd1a1d1a4d09cd19dd195d09cd19bd088d196d1add1b3d187d1aed1a7d192d187d08ed19f
```

Ввод [40]:
```
c1_c2 = butes_print(c1, c2)
```

```
0f1e240d03017c0e0770777200090553d081d08fd08ad084
```

Ввод [41]:
```
found = xor_string(c1_c2, p1)
print(found)
```

```
ВЮжныйфилиалБанка
```

Приложение написано на python 3. Я запускала его через jupiter Notebook. В данном коде имеется 2 основные функции. 1 - сложение по модулю 2, 2 - представление в байтовом виде.

2. Определим вид шифро-текстов C1 и C2 обоих текстов P1 и P2 при известном ключе. Для того запустим данную часть кода в нашем ноутбуке.

   (рис. -@fig:002)

Ввод [39]:
```
c1 = butes_print(p1, k)
c2 = butes_print(p2, k)
```

```
d19bd193d185d1a9d09fd19cd0a9d092d19cd1b8d0a1d09fd1b3d18ed1abd0b45348045b
d194d18dd1a1d1a4d09cd19dd195d09cd19bd088d196d1add1b3d187d1aed1a7d192d187d08ed19f
```

С помощью функции byte_print() шифруем оба текста p1 и p2 с помощью одного заранее созданного ключа k. byte_print() вызывает xor_string(), которая складывает два предложения по модулю 2. Сохраняем полученные зашифрованные тексты в переменные c1 и c2. Далее функция представляет их в буквенном виде и показывает нам.

3. Далее воспользуемся способом, который даст нам разгадать оба зашифрованных текста, без использования нашего ключа.

   Следуем дальше этой схеме, представленной в инструкции к лабораторной работе.

   (рис. -@fig:003)

$$C_1 \oplus C_2 \oplus P_1 = P_1 \oplus P_2 \oplus P_1 = P_2.$$

Складываем 2 зашифрованных текста по модулю. Получившийся результат складываем с одним из расшифрованных текстов p1. Выводим получившийся результат сложения c1, c2 и p1. У нас получается расшифрованный текст p2, который мы сохраняем. Теперь уже получившийся расшифрованный текст складываем по модулю 2 с c1 и c2. Теперь же у нас получается расшифрованный p1.

(рис. -@fig:004)

```
c1_c2 = butes_print(c1, c2)
```

```
0f1e240d03017c0e0770777200090553d081d08fd08ad084
```

```
found = xor_string(c1_c2, p1)
print(found)
```

```
ВЮжныйныйфилиалБанка
```

```
found2 = xor_string(c1_c2, found)
print(found2)
```

```
НаВашисходящийот1204
```

Т.е. мы можем расшифровать любой текст, если у нас имеется два зашифрованных текста одной гаммой, и один расшифрованный текст. Т. е нам даже не нужно знать шифровальный ключ для данных операций.

Выводы

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

Список литературы

1. Шифры замены и табличного гаммирования // Хабр URL: https://habr.com/ru/post/583616/ (дата обращения: 10.12.2021).
2. Лабораторная работа № 7. Элементы криптографии. Однократное гаммирование // Туис URL: https://esystem.rudn.ru/pluginfile.php/1198312/mod_resource/content/2/007-lab_crypto-gamma.pdf (дата обращения: 9.12.2021).
3. Простейшие методы шифрования с закрытым ключом // НОУ ИНТУТ URL: https://intuit.ru/studies/courses/691/547/lecture/12373?page=4 (дата обращения: 9.12.2021).