

Лабораторная работа №8

Элементы криптографии. Шифрование (кодирование) различных
исходных текстов одним ключом

Калинина Кристина Сергеевна

Содержание

Цель работы	5
Теоретические сведения	6
Выполнение лабораторной работы	8
Выводы	10
Контрольные вопросы	11
Список литературы	12

List of Figures

0.1	Блок программы с библиотеками и функциями	8
0.2	Блок программы с выполнением задания	9

List of Tables

Цель работы

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

Теоретические сведения

С точки зрения теории криптоанализа метод шифрования однократной случайной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым (далее для краткости авторы будут употреблять термин “однократное гаммирование”, держа в уме все вышесказанное). Обоснование, которое привел Шеннон, основываясь на введенном им же понятии информации, не дает возможности усомниться в этом - из-за равных априорных вероятностей криптоаналитик не может сказать о дешифровке, верна она или нет. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько нибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях. [1]

Логично было бы предположить, что для организации канала конфиденциальной связи в открытых сетях следовало бы воспользоваться именно схемой шифрования однократного гаммирования. Ее преимущества вроде бы очевидны. Есть, правда, один весомый недостаток, который сразу бросается в глаза, - это необходимость иметь огромные объемы данных, которые можно было бы использовать в качестве гаммы. Для этих целей обычно пользуются датчиками настоящих случайных чисел (в западной литературе аналогичный термин носит название True Random Number Generator или TRNG). Это уже аппаратные устройства, которые по запросу выдают набор случайных чисел, генерируя их с помощью очень большого количества физических параметров окружающей среды. Статистические характеристики таких наборов весьма близки к характеристикам “белого шума”, что означает равновероятное появление каждого следующего числа в наборе. А это, в свою очередь, означает

для нас действительно равновероятную гамму. [1]

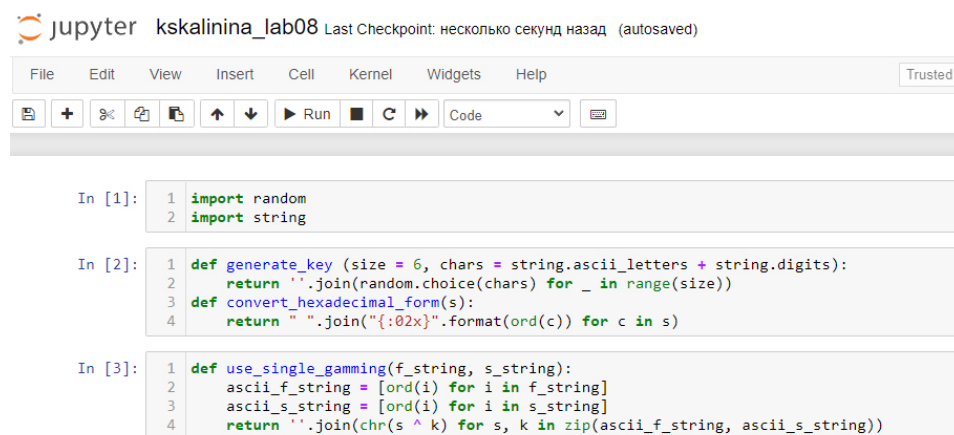
К сожалению, для того чтобы организовать конфиденциальный канал передачи данных, потребуется записать довольно большое количество этих данных и обменяться ими по секретному каналу. [1]

Уже одно это условие делает однократное гаммирование во многих случаях неприемлемым. В самом деле, зачем передавать что-то по открытому незащищенному каналу, когда есть возможность передать все это по секретному защищенному? И хотя на простой вопрос, является ли метод использования однократной случайной равновероятной гаммы стойким к взлому, существует положительный ответ, его использование может оказаться попросту невозможным. [1]

Да и к тому же метод однократного гаммирования криптостоек только в определенных, можно даже сказать, тепличных условиях. [1]

Выполнение лабораторной работы

1. Для реализации приложения было принято решение воспользоваться jupyter notebook в котором я создала файл и на языке python написала программный код.
2. Сначала я подключила необходимые библиотеки: random для использования рандомайзера при генерации ключей и string для использования констант. Затем я написала две функции. Первая - generate_key - генерирует ключ той же длины, что и строка, которую нужно зашифровать, принимая на вход длину этой самой строки. Функция возвращает сгенерированный ключ в виде строки. Вторая - convert_hexadecimal_form - выполняет перевод в шестнадцатичную систему строку, которую принимает на вход. Функция возвращает строку в шестнадцатичном виде (fig. 0.1).



The screenshot shows a Jupyter Notebook window titled 'kskalinina_lab08'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains three code cells:

```
In [1]: 1 import random
        2 import string

In [2]: 1 def generate_key(size = 6, chars = string.ascii_letters + string.digits):
        2     return ''.join(random.choice(chars) for _ in range(size))
        3 def convert_hexadecimal_form(s):
        4     return " ".join("{:02x}".format(ord(c)) for c in s)

In [3]: 1 def use_single_gamming(f_string, s_string):
        2     ascii_f_string = [ord(i) for i in f_string]
        3     ascii_s_string = [ord(i) for i in s_string]
        4     return ''.join(chr(s ^ k) for s, k in zip(ascii_f_string, ascii_s_string))
```

Figure 0.1: Блок программы с библиотеками и функциями

3. В задании нужно было прочитать оба текста, не зная ключ и не стремясь его определить. Поэтому я ввела начальные данные, сгенерировала ключ и зашифровала их. После этого с помощью однократного гаммирования я сложила 2 шифротекста с одним из начальных и получила второй (fig. 0.2).

```
Задание
In [4]: 1 P1 = "НаВашисходящийот1204"
2 P2 = "ВСеверныйфилиалБанка"
3
4 print("P1:", P1)
5 print("P2:", P2)
6
7 K = generate_key(20)
8
9 print("\nK:", K)
10 print("K(16):", convert_hexadecimal_form(K))
11
12 C1 = use_single_gamming(P1, K)
13 C2 = use_single_gamming(P2, K)
14
15 print("\nC1:", C1)
16 print("C1(16):", convert_hexadecimal_form(C1))
17
18 print("\nC2:", C2)
19 print("C2(16):", convert_hexadecimal_form(C2))
20
21 C1C2 = use_single_gamming(C1, C2)
22
23 print("\nВывод текста без ключа:")
24 print("P1:", use_single_gamming(C1C2, P2))
25 print("P2:", use_single_gamming(C1C2, P1))

P1: НаВашисходящийот1204
P2: ВСеверныйфилиалБанка

K: YwJEZhWnE4wfy5a3gNe6
K(16): 59 77 4a 45 32 68 57 6e 45 34 77 66 79 35 61 33 67 4e 65 36

C1: фчјѵ0ѐЖюЁиЯскѹѵV|UѰ
C1(16): 444 447 458 475 47a 450 416 42b 47b 400 438 42f 441 40c 45f 471 56 7c 55 02

C2: ыіѵѵIШXѰѵЯйсSыTіouI
C2(16): 44b 456 47f 477 407 428 46a 425 47c 470 44f 45d 441 405 45a 422 457 473 45f 406

Вывод текста без ключа:
P1: НаВашисходящийот1204
P2: ВСеверныйфилиалБанка
```

Figure 0.2: Блок программы с выполнением задания

Выводы

Таким образом я успешно освоила на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Контрольные вопросы

1. Как, зная один из текстов (P_1 или P_2), определить другой, не зная при этом ключа?

Нужно сложить по модулю два оба шифротекста, полученный результат сложить по модулю два с одним из начальных текстов. Таким образом получится второй начальный текст.

2. Что будет при повторном использовании ключа при шифровании текста?

Два текста получатся взаимосвязанными, что плохо скажется на безопасности шифрования.

3. Как реализуется режим шифрования однократного гаммирования одним ключом двух открытых текстов?

Каждый текст отдельно гаммируется одним и тем же ключом.

4. Перечислите недостатки шифрования одним ключом двух открытых текстов.

Увеличивается шанс взлома, т.к. зная один исходный текст, можно получить второй, при этом не нужно знать ключ.

5. Перечислите преимущества шифрования одним ключом двух открытых текстов.

Уменьшение затрат при передаче ключей.

Список литературы

1. Использование однократного гаммирования. // bugtraq.ru URL: <https://bugtraq.ru/library/boo> (дата обращения 18.12.2021).
2. Д. С. Кулябов, А. В. Королькова, М. Н. Геворкян. Информационная безопасность компьютерных сетей: лабораторные работы. // Факультет физико-математических и естественных наук. М.: РУДН, 2015. 64 с..