Отчёт по лабораторной работе №3

Шифр гаммирования

Бакундукизе Эжид Принц НФИмд-01-21

Содержание

List of Figures

1 Цель работы

Изучение алгоритма шифрования гаммированием.

2 Теоретические сведения

2.1 Шифр гаммирования

Гаммирование — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных [1].

Схема шифрования гаммированием представлена на рис.1.

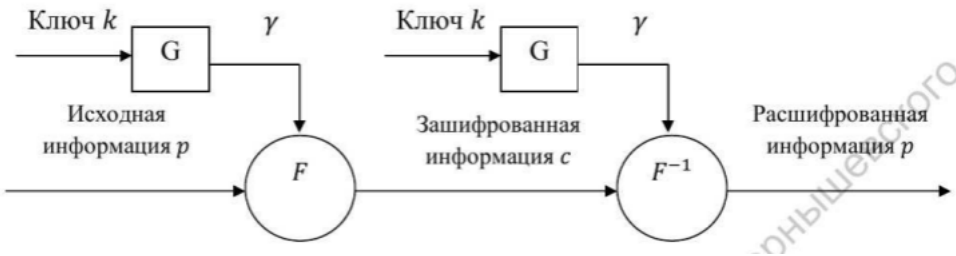

Figure 2.1: Схема шифрования гаммированием

Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, т.е. псевдослучайной последовательности выходов генератора G. Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но является детерминированной, т.е. известен алгоритм ее формирования. Обчно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N - число букв алфавита открытого текста).

Простейший генератор пседвослучайной последовательности можно предста-

вить рекуррентным соотношением (рис.2):

$$\gamma_i = a \cdot \gamma_{i-1} + b \; mod(m), i = \overline{1, m},$$

где γ_i – i-й член последовательности псевдослучайных чисел, a, γ_0, b – ключевые параметры. Такая последовательность состоит из целых чисел от 0 до $m - 1$. Если элементы γ_i и γ_j совпадут, то совпадут и последующие участки: γ_{+1i}=γ_{j+1}, $\gamma_{i+2} = \gamma_{j+2}$. Таким образом, ПСП является периодической. Знание периода гаммы существенно облегчает криптоанализ. Максимальная длина периода равна m. Для ее достижения необходимо удовлетворить следующим условиям:

1. b и m – взаимно простые числа;
2. $a - 1$ делится на любой простой делитель числа m;
3. $a - 1$ кратно 4, если m кратно 4.

Figure 2.2: Простейший генератор псевдослучайной последовательности

Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей и изменяется случайным образом для каждого шифруемого слова. Если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа) [2].

Стойкость шифров, основанных на процедуре гаммирования, зависит от характеристик гаммы - длины и равномерности распределения вероятностей появления знаков гаммы. При использовании генератора псевдослучайных последовательностей получаем бесконечную гамму. Однако, возможен режим шифрования конехной гаммы. В роли конечной гаммы выступать фраза.

Метод гаммирования становится бессильным, если известен фрагмент исходного текста и соответствующая ему шифрограмма. В этом случае простым вычитанием по модулю 2 получается отрезок псевдослучайной последовательности и по нему восстанавливается вся эта последовательность.

3 Выполнение работы

3.1 Реализация алгоритма шифрования гаммирования конечной гаммой на Python

```
def gamma():
    #создаем алфавит
    dict = {"а" :1, "б" :2 , "в" :3 ,"г" :4 ,"д" :5 ,"е" :6,"ж": 7, "з": 8, "и":
            "м": 13, "н": 14, "о": 15, "п": 16,
            "р": 17, "с": 18, "т": 19, "у": 20, "ф": 21, "х": 22, "ц": 23, "ч": 2
            "ы": 28, "ь": 29, "э": 30, "ю": 31, "я": 32
            }

    # меняем местами ключ и значение, такой словарь понадобится для расшифровки
    dict2 = {v: k for k, v in dict.items()}

    g = input("Гамма: ").lower()
    text = input("Текст для шифрования: ").lower()

    list_text = list() #числа букв из текста
    list_gamma = list() #числа букв для гаммы

    #запишем числа в список
    for i in text:
```

```
    list_text.append(dict[i])
print("Числа текста: ", list_text)

#запишем числа для гаммы

for i in g:
    list_gamma.append(dict[i])
print("числа гаммы: ", list_gamma)

result = list() #сюда будем записывать результат

ch = 0
for i in text:
    try:
        a = dict[i] + list_gamma[ch]
    except:
        ch=0
        a = dict[i] + list_gamma[ch]
    if a >=33:
        a = a%33
    ch+=1
    result.append(a)

print("Числа зашифрованного текста: ", result)

# теперь обратно числа представим в виде букв
text2=""
for i in result:
    text2 += dict2[i]
```

```
print("Зашифрованный текст: ", text2)

#Дешифровка

list_text2 = list()

for i in text2:
    list_text2.append(dict[i])

ch = 0
list_text2_2 = list()
for i in list_text2:
    try:
        a = i - list_gamma[ch]
    except:
        ch=0
        a = i - list_gamma[ch]
    if a < 1:
        a = 33 + a

    list_text2_2.append(a)
    ch+=1

text_decrypted = ""
for i in list_text2_2:
    text_decrypted+=dict2[i]

print("Дешифровка: ", text_decrypted)
```

3.2 Контрольный пример

```
In [24]: gamma()

Гамма: гамма
Текст для шифрования: приказ
Числа текста:  [16, 17, 9, 11, 1, 8]
числа гаммы:  [4, 1, 13, 13, 1]
Числа зашифрованного текста:  [20, 18, 22, 24, 2, 12]
Зашифрованный текст:  усхчбл
Дешифровка:  приказ
```

Figure 3.1: Работа алгоритма гаммирования

4 Выводы

В ходе выполнения данной лабораторной работы мы познакомились с алгоритмом шифрования гаммирования. В рамках задания программно были реализованы алгоритмы шифрования и дешифрования гаммированием конечной гаммы.

Список литературы

1. Шифрование методом гаммирования
2. Режим гаммирования в блочном алгоритме шифрования