Отчёт по лабораторной работе 3. Шифрование гаммированием

Ильин Никита Евгеньевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной работы – научиться программировать Шифрование гаммированием/

2 Задание

1. Реализовать алгоритм шифрование гаммированием

3 Теоретическое введение

Из всех схем шифрования простейшей и наиболее надежной является схема однократного использования (рис. 1). Формируется т- разрядная случайная двоичная последовательность - ключ шифра. Отправитель производит побитовое сложение по модулю два (mod 2) ключа k = k / k 2 . . . K i . . . mK и т- разрядной двоичной последовательности p= P1P2 ...Pi...Pm, соответствующей посылаемому сообщению: Ci = p i @ k ; , i = 1 , m , где pi - i-й бит исходного текста, i - i-й бит ключа, ® - операция побитового сложения (XOR), c; - i-й бит получившейся криптограммы =C C1C2. Ci. Cm• Операция побитного сложения является обратимой, т.е. (x0y)[**y?**] = x, поэтому дешифрование осуществляется повторным применением операции 0 к криптограмме: Ключ к Исходная информация p XOR Pi = c, 0 k , i = 1 , m . Ключ k Зашифрованная информация c XOR Рис. 1 Расшифрованная информация p Основным недостатком такой схемы является равенство объема ключевой информации и суммарного объема передаваемых сообщений. Данный недостаток можно убрать, использовав ключ в качестве «зародыша», порождающего 12 Черны значительно более длинную ключевую последовательность. представлена такая схема, которая и называется гаммированием. На рис. .2 Kuroy k Исходная информация p Зашифрованная информация c Y Расшифрованная информация p F-1 Ключ k GG Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, т.е. псевдослучайной последовательности (ПСП) с выходов генератора G. Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но является детерминированной, т.е. известен алгоритм е формирования.

Чаще Обычно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N - число букв алфавита открытого текста). Простейший генератор • псевдослучайной последовательности можно представить рекуррентным соотношением: Vi = a • V i - 1 + b m o d (m) , i = 1 , m , где Vi - i-й член последовательности псевдослучайных чисел, a, Yo, b - ключевые параметры. Такая последовательность состоит из целых чисел от 0 од т - .1 Если элементы Vi и Y; совпадут, то совпадут и последующие участки: Y+1 Yj+1, Vi+z = Vj+2. Таким образом, ПСП является периодической. Знание периода гаммы существенно облегчает криптоанализ. Максимальная длина периода равна т. Для ее достижения необходимо удовлетворить следующим условиям: 1 b и т - взаимно простые числа; 2. a - 1 делится на любой простой делитель числа т ; 3. a - 1 кратно 4, если т кратно 4. 13

Стойкость шифров, основанных на процедуре гаммирования, зависит от характеристик гаммы - длины и равномерности распределения вероятностей появления знаков гаммы. При использовании генератора ПСП получаем бесконечную гамму. Однако, возможен режим шифрования конечной гаммы. В роли конечной гаммы может выступать фраза. Как и ранее, используется алфавитный порядок букв, т.е. буква «а» имеет порядковый номер 1, «б» - 2 и т.д. Например, зашифруем слово «ПРИКАЗ» («16 17 09 1 01 08») гаммой «ГАММА» («04 01 13 13 01»). Будем использовать операцию побитового сложения по модулю 3 (mod 33). Получаем: C1 = 1 6 + 4 (m o d 3 3) = 2 0 2 c = 7 1 + 1 m(o d 3 3) = 8 1 C3 = 9 + 1 3 (m o d 3 3) = 2 2 C4 = 1 1 + 1 3 (m o d 3 3) = 2 4 = s (1 + 1 m(o d 3 3) 2 = c 6 = 8 + 4 (m o d 3 3) = 1 2 . Криптограмма: «УСХЧБЛ» («20 18 22 24 02 12»).

4 Выполнение лабораторной работы

1. Для начала реализуется алгоритм шифра на языке Python (рис. 4.1).

```
def gamma_encrypt(message, gamma):

    def encrypt(letters_pair):
        idx = (letters_pair[0]+1)+(letters_pair[1]+1) % len(alph)
        if idx > len(alph):
            idx = idx - len(alph)
        return idx - 1

    alph = list(map(chr, range(ord('а'), ord('я')+1)))
    message_clear = list(filter(lambda s: s.lower() in alph, message))
    gamma_clear = list(filter(lambda s: s.lower() in alph, gamma))
    message_ind=list(map(lambda s: alph.index(s.lower()),message_clear))
    gamma_ind=list(map(lambda s: alph.index(s.lower()),gamma_clear))

    for i in range(len(message_ind) - len(gamma_ind)):
        gamma_ind.append(gamma_ind[i])

    print(f'{message.upper()} -> {message_ind}\n{gamma.upper()} -> {gamma_ind}')
    encrypted_ind = list(map(lambda s: encrypt(s), zip(message_ind, gamma_ind)))
    print(f'Зашифрованное сообщение: {encrypted_ind}\n')
    return ''.join(list(map(lambda s: alph[s], encrypted_ind))).upper()
```

Рис. 4.1: Программная реализация шифра Цезаря

2. Зашифрованное сообщение выглядит следующим образом (рис. 4.2).

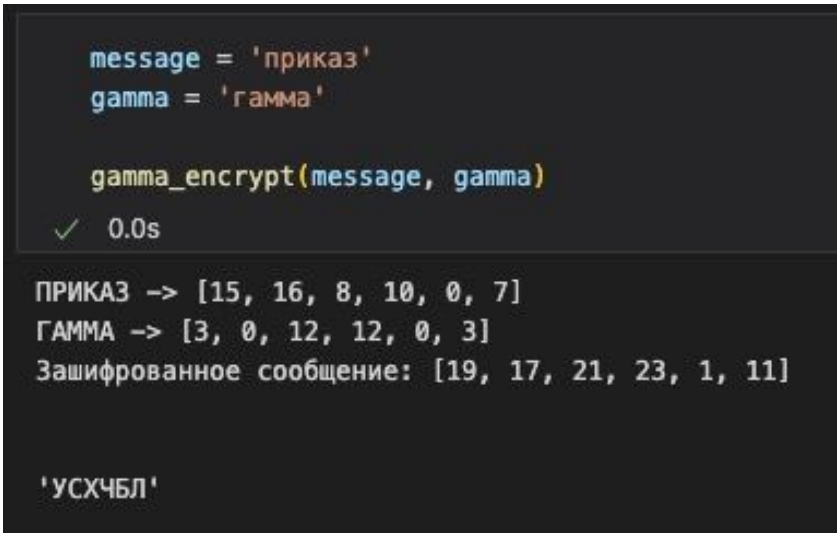

Рис. 4.2: Результат работы программы

5 Выводы

В ходе работы был реализован алгоритм шифрования гаммированием.