МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Государственное образовательное учреждение высшего профессионального образования   
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

Кафедра компьютерных технологий и программной инженерии (№43)

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| канд. техн. наук, доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Фаткиева Р. Р. |
|  | подпись, дата |  |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

ШИФР ГАММИРОВАНИЯ

По дисциплине: ЗАЩИТА ИНФОРМАЦИИ

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |
| --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4931 |  | Кинько А. А. |
|  |  | подпись, дата | фамилия, инициалы |

**Санкт-Петербург, 2023**

**Лабораторная работа №2**

**«Шифр гаммирования»**

**I. Цель работы**

Целью работы является освоение принципов шифрования гаммирования, изучение свойств генератора псевдослучайных чисел, программная реализация метода гаммирования.

**II. Постановка задачи**

Порядок выполнения лабораторной работы:

1. Выбрать параметры генератора ПСЧ: в соответствии с вариантом №12:  
   Вид генератора ПСЧ – гаммирование с обратной связью.
2. Разработать программу шифрования и дешифрования текста.
3. Произвести шифрование исходного текста, получить шифрограмму, осуществить ее дешифрование и сравнение с исходным текстом. Рекомендуется для представления символов исходного текста использовать стандартную кодировку символов.
4. Произвести изменение одного или несколько параметров генератора случайных чисел, осуществить получение шифрограммы и сравнение ее с предыдущим вариантом.
5. Результат работы оформить в виде отчета.

**III. Ход работы**

**Описание используемого алгоритма**

Принцип шифрования гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы шифра на открытые данные обратимым образом (используя операцию сложения по модулю 2).

где – бит исходного текста; – бит зашифрованного текста; - бит гаммы.

Процесс дешифрования сводится к повторной генерации гаммы шифра при известном ключе и наложении такой же гаммы на зашифрованные данные.

При использовании обратной связи значение зашифрованного символа зависит не только от гаммы, но и от предыдущих символов. Процесс шифрования представляется следующими шагами:

1. Генерация сегмента гаммы и наложение его на соответствующий участок шифруемых данных.

2. Подсчет контрольной суммы участка, соответствующего сегменту гаммы .

3. Генерация с учетом контрольной суммы уже зашифрованного участка данных следующего сегмента гамм .

4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных и т.д.

Под контрольной суммой понимают функцию где – -е слово шифруемых данных.

Чтобы получить линейные последовательности элементов гаммы, длина которых не превышает размер шифруемых данных, используют датчики ПСЧ. Одним из хороших конгруэнтных генераторов является линейный конгруэнтный датчик ПСЧ. Он вырабатывает последовательности псевдослучайных чисел , описываемые соотношением:

где – константы, – исходная величина, выбранная в качестве порождающего числа. Очевидно, что эти три величины образуют ключ. Значение обычно устанавливается равным где – длина машинного слова в битах. Необходимо выбирать числа и так, чтобы период был максимальный, что достигается, когда – нечетное и .

Для кодирования символов в слове составим таблицу, где первые 30 символов будут представлять большие русские буквы (все за исключением «Ь», «Ъ» и «Ы»), затем все прописные, следом за ними большие и прописные буквы латинского алфавита, а также точка и знак подчеркивания.

Зашифруем исходный текст «кот», представленный в виде последовательности 0101001, 0101101, 0110001. Тогда . В качестве контрольной суммы будем брать количество единиц в бинарном представлении буквы. Тогда сегменту соответствует участок , контрольная сумма равна 3. Далее, для буквы «о» проделываем такие же действия, но принимая . В итоге получаем гамму шифра: 0000011, 0110010, 0111101.

Проделав операцию сложения по модулю 2, получаем шифрограмму: 0101010, 0011111, 0001100, которая соответствует тексту «лбЛ».

**IV. Результаты работы программы**

Для исходных данных:

--- Кодируем строку: Какой\_прекрасный\_день

Полученные коды букв текста: ['0001011', '0011110', '0101001', '0101101', '0101000', '1111110', '0101110', '0101111', '0100011', '0101001', '0101111', '0011110', '0110000', '0101100', '0111010', '0101000', '1111110', '0100010', '0100011', '0101100', '0111011']

Соответствующие контрольные суммы: [3, 4, 3, 4, 2, 6, 4, 5, 3, 3, 5, 4, 2, 3, 4, 2, 6, 2, 3, 3, 5]

Полученная гамма: ['0000011', '0110010', '0111101', '0110010', '0111101', '0100111', '1010011', '0111101', '1001000', '0110010', '0110010', '1001000', '0111101', '0100111', '0110010', '0111101', '0100111', '1010011', '0100111', '0110010', '0110010']

Полученные закодированные буквы: ['0001000', '0101100', '0010100', '0011111', '0010101', '1011001', '1111101', '0010010', '1101011', '0011011', '0011101', '1010110', '0001101', '0001011', '0001000', '0010101', '1011001', '1110001', '0000100', '0011110', '0001001']

Закодированный текст: ЗнУбФa.СsЭЯXМКЗФayДаИ

--- Декодируем текст: ЗнУбФa.СsЭЯXМКЗФayДаИ

Декодированный текст: Какой\_прекрасный\_день

Для измененных параметров:

--- Кодируем строку: Какой\_прекрасный\_день

Полученные коды букв текста: ['0001011', '0011110', '0101001', '0101101', '0101000', '1111110', '0101110', '0101111', '0100011', '0101001', '0101111', '0011110', '0110000', '0101100', '0111010', '0101000', '1111110', '0100010', '0100011', '0101100', '0111011']

Соответствующие контрольные суммы: [3, 4, 3, 4, 2, 6, 4, 5, 3, 3, 5, 4, 2, 3, 4, 2, 6, 2, 3, 3, 5]

Полученная гамма: ['0001000', '0001110', '0010001', '0001110', '0010001', '0001011', '0010111', '0010001', '0010100', '0001110', '0001110', '0010100', '0010001', '0001011', '0001110', '0010001', '0001011', '0010111', '0001011', '0001110', '0001110']

Полученные закодированные буквы: ['0000011', '0010000', '0111000', '0100011', '0111001', '1110101', '0111001', '0111110', '0110111', '0100111', '0100001', '0001010', '0100001', '0100111', '0110100', '0111001', '1110101', '0110101', '0101000', '0100010', '0110101']

Закодированный текст: ГПщеъ3ъяшигЙгихъ3цйдц

--- Декодируем текст: ГПщеъ3ъяшигЙгихъ3цйдц

Декодированный текст: Какой\_прекрасный\_день

Так, можно заметить, что от изменения параметров генератора псевдослучайных чисел, меняется получаемая шифрограмма – во втором случае большее количество прописных букв. Также из примеров видно, что контрольные суммы не зависят от параметров установки, а только от введенного текста.

**V. Выводы**

В ходе выполнения данной лабораторной работы были получены навыки шифрования текста методом гаммирования, в частности, был изучен алгоритм гаммирования с обратной связью, а также реализована программа на высоком языке программирования, выполняющая как кодирование, так и декодирования исходного текста.

**Приложение А. Листинг программы.**

Листинг программы также доступен по ссылке: <https://github.com/ArtemKinko/SUAI-labs-spring-2023/tree/main/DP/LAB-2_DP>

Файл «lab2-DP.py»:

import io  
  
  
A = 3  
C = 5  
T0 = 1  
b = 7  
M = 128  
  
  
def get\_alphabet\_from\_file(alphabet\_file\_name):  
 with io.open(alphabet\_file\_name, encoding="utf-8") as alphabet\_file:  
 alphabet = []  
 for \_ in range(128):  
 letter = alphabet\_file.read().rsplit()  
 alphabet.append(letter)  
 alphabet.append(" ")  
 return alphabet[0]  
  
  
def encode\_text(text\_string, alphabet):  
 print("\n--- Кодируем строку:", text\_string)  
 indexes = []  
 gamma = []  
 prev\_T = T0  
 control\_summary = []  
 for letter in text\_string:  
 indexes.append(alphabet.index(letter))  
 current\_gamma = (A \* prev\_T + C) % M  
 prev\_T = format(alphabet.index(letter), 'b').count('1')  
 control\_summary.append(prev\_T)  
 gamma.append(current\_gamma)  
 binary\_indexes = [format(x, '07b') for x in indexes]  
 print("Полученные коды букв текста:\t", binary\_indexes)  
 print("Соответствующие контрольные суммы:", control\_summary)  
 binary\_gamma = [format(x, '07b') for x in gamma]  
 print("Полученная гамма:\t\t\t\t", binary\_gamma)  
 encoded\_indexes = [indexes[i] ^ gamma[i] for i in range(len(gamma))]  
 binary\_encoded = [format(x, '07b') for x in encoded\_indexes]  
 print("Полученные закодированные буквы:", binary\_encoded)  
 encoded\_text = ""  
 for encoded\_index in encoded\_indexes:  
 encoded\_text += alphabet[encoded\_index]  
 print("Закодированный текст:", encoded\_text)  
 return encoded\_text, gamma  
  
  
def decode\_text(text\_string, alphabet, gamma):  
 print("\n--- Декодируем текст:", text\_string)  
 decoded\_text = ""  
 for i in range(len(text\_string)):  
 index = alphabet.index(text\_string[i])  
 decoded\_text += alphabet[index ^ gamma[i]]  
 print("Декодированный текст:", decoded\_text)  
 return decoded\_text  
  
  
letters = get\_alphabet\_from\_file("symbols.txt")  
string\_to\_decode = "Какой\_прекрасный\_день"  
encoded\_text, gamma\_text = encode\_text(string\_to\_decode, letters)  
decoded\_text = decode\_text(encoded\_text, letters, gamma\_text)

Файл «symbols.txt»:

А  
Б  
В  
Г  
Д  
Е  
Ё  
Ж  
З  
И  
Й  
К  
Л  
М  
Н  
О  
П  
Р  
С  
Т  
У  
Ф  
Х  
Ц  
Ч  
Ш  
Щ  
Э  
Ю  
Я  
а  
б  
в  
г  
д  
е  
ё  
ж  
з  
и  
й  
к  
л  
м  
н  
о  
п  
р  
с  
т  
у  
ф  
х  
ц  
ч  
ш  
щ  
ъ  
ы  
ь  
э  
ю  
я  
A  
B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q  
r  
s  
t  
u  
v  
w  
x  
y  
z  
1  
2  
3  
4  
5  
6  
7  
8  
9  
0  
.  
\_