

# Лабораторная работа 7.

## Элементы криптографии. Однократное гаммирование.

---

Поленикова Анна Алексеевна

Москва, 2021

Российский Университет Дружбы Народов

## Цель лабораторной работы

Освоить на практике применение режима однократного гаммирования.

Гаммирование - наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

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.



Рис. 1: Схема однократного использования Вернама

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i$$

где  $C_i$  —  $i$ -й символ получившегося зашифрованного послания,  $P_i$  —  $i$ -й символ открытого текста,  $K_i$  —  $i$ -й символ ключа,  $i = 1, m$ .

# Функции программы-шифратора

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4
5  using namespace std;
6
7  string tranform(string toEncrypt) {
8      string transformedMessage;
9      for (int i = 0; i < toEncrypt.size(); ++i) {
10         int ia = toascii(toEncrypt.at(i));
11         transformedMessage.push_back(ia);
12     }
13     return transformedMessage;
14 }
15
16 string keyGeneration(string toEncrypt) {
17     string key;
18     static const char code[] = "0123456789";
19
20     for (int i = 0; i < toEncrypt.size(); ++i) {
21         key += code[rand() % (sizeof(code) - 1)];
22     }
23
24     return key;
25 }
26
27 string encryptDecrypt(string toEncrypt, string key) {
28     string output = toEncrypt;
29
30     for (int i = 0; i < toEncrypt.size(); i++)
31         output[i] = toEncrypt[i] ^ key[i];
32
33     return output;
34 }
```

Рис. 2: Функции программы-шифратора

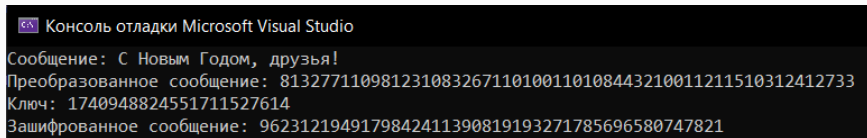
## main функция программы-шифратора

```
36 int main(int argc, const char* argv[])
37 {
38     setlocale(LC_ALL, "Russian");
39
40     string message = "С Новым Годом, друзья!";
41
42     string transformedMessage = tranform(message);
43
44     string key = keyGeneration(transformedMessage);
45
46     string encrypted = encryptDecrypt(transformedMessage, key);
47
48
49     cout << "Сообщение: " << message << "\n";
50     cout << "Преобразованное сообщение: ";
51     for (int i = 0; i < transformedMessage.size(); ++i) {
52         cout << int(transformedMessage[i]);
53     }
54     cout << "\n" << "Ключ: " << key << "\n";
55     cout << "Зашифрованное сообщение: ";
56     for (int i = 0; i < encrypted.size(); ++i) {
57         cout << int(encrypted[i]);
58     }
59
60     return 0;
61 }
```

Рис. 3: main функция программы-шифратора



# Результат работы программы-шифратора



```
Консоль отладки Microsoft Visual Studio  
Сообщение: С Новым Годом, друзья!  
Преобразованное сообщение: 813277110981231083267110100110108443210011211510312412733  
Ключ: 1740948824551711527614  
Зашифрованное сообщение: 9623121949179842411390819193271785696580747821
```

Рис. 4: Результат работы программы-шифратора

В ходе выполнения лабораторной работы было освоено на практике применение режима однократного гаммирования.