# Информационная безопасность. Отчет по лабораторной работе №7

**Элементы криптографии. Однократное гаммирование**

Горбунова Ярослава Михайловна

# Содержание

# List of Figures

# List of Tables

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования [1].

# 2 Указание к работе

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» (fig. 2.1) является простой, но надёжной схемой шифрования данных. Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, 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. Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком ⊠) между элементами гаммы и элементами подлежащего сокрытию текста. Напомним, как работает операция XOR над битами: 0⊠0 = 0, 0⊠1 = 1, 1⊠0 = 1, 1⊠1 = 0. Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное

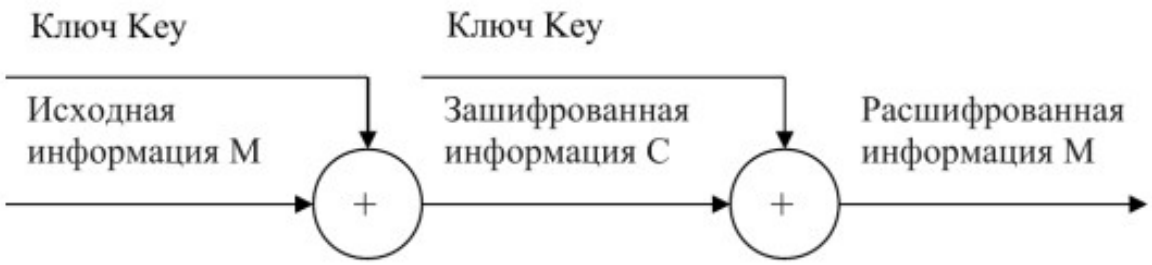


Figure 2.1: Схема однократного использования Вернама

значение, а шифрование и расшифрование выполняется одной и той же программой. Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила (fig. 2.2):

$$C_i = P_i \oplus K_i,$$

Figure 2.2: Формула 7.1

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. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины. Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с (fig. 2.2), а именно, обе части равенства необходимо сложить по модулю 2 с P_i (fig. 2.3):

$$C_i \oplus P_i = P_i \oplus K_i \oplus P_i = K_i,$$
$$K_i = C_i \oplus P_i.$$

Figure 2.3: Формула 7.2

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов.

К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P .

Необходимые и достаточные условия абсолютной стойкости шифра: – полная случайность ключа; – равенство длин ключа и открытого текста; – однократное использование ключа.

Рассмотрим пример.

Ключ Центра:

05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54

Сообщение Центра:

Штирлиц – Вы Герой!!

D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C3 E5 F0 EE E9 21 21

Зашифрованный текст, находящийся у Мюллера:

DD FE FF 8F E5 A6 C1 F2 B9 30 CB D5 02 94 1A 38 E5 5B 51 75

Дешифровальщики попробовали ключ:

05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 55 F4 D3 07 BB BC 54

и получили текст:

D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C1 EE EB E2 E0 ED 21

Штирлиц - Вы Болван!

Другие ключи дадут лишь новые фразы, пословицы, стихотворные строфы, словом, всевозможные тексты заданной длины.

# 3 Выполнение лабораторной работы

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно: 1. Определить вид шифротекста при известном ключе и известном открытом тексте. 2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

Для выполнения работы была написана программа (fig. 3.1 - fig. 3.4) с помощью языка программирования C++, которая получает на вход открытый текст “Stirlis - Vy Geroy!!” и ключ “05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54”, затем шифрует открытый текст методом однократного гаммирования и получает шифротекст. После этого методом сложения по модулю 2 или дешифровки методом однократного гаммирования определяет ключ, с помощью которого из полученного шифротекста можно получить целевой фрагмент текста, указанный в задании, “S Novym Godom drysya” (в данном сообщении были убраны запятая и восклицательный знак, чтобы добиться одинаковой размерности открытого и целевого текста, также сообщения транскрипированы на латиницу).

```cpp
/*
Gorbunova Y.M., NFI-01-19, 2022

Single gamming
Principle:
1) text ^ gamma = ciphertext;
2) ciphertext ^ gamma = text;
3) gamma = ciphertext ^ text;
where "^" is additions modulo 2 (XOR)
*/


#include <iostream>
#include <string>
#include <cstring>
#include <bitset>

using namespace std;

const int m = 1024;  // array dimension
const int n = 8;  // bitset dimension

// Print for unmodified text
void print_bitset_text(char arr[]) {
    for (unsigned int i = 0; i < strlen(arr); i++)
        cout << bitset<n>((unsigned char)arr[i]) << " ";
}

// Print for modified text
void print_bitset_gamma(char arr[], char text[]) {
    for (unsigned int i = 0; i < strlen(text); i++)
        cout << bitset<n>((unsigned char)arr[i]) << " ";
}
```

Figure 3.1: Программа (1)

```cpp

int main()
{
    // Introduce variables
    //char text[m] = "Штирлиц - Вы Герой!!";
    char text[m] = "Stirlis - Vy Geroy!!";
    //char target_text[m] = "С Новым Годом друзья";
    char target_text[m] = "S Novym Godom drysya";
    int gam[m] = {
        0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,
        0x09, 0x2E, 0x22, 0x57, 0xFF, 0xC8, 0x0B, 0xB2, 0x70, 0x54
    };  // given key
    char gamma_1[m];  // given key
    char gamma_2[m];  // target key to obtain target_text from text
    char gam_text[1024];

    // Convert from integer to char
    for (unsigned int i = 0; i < size(gam); i++) {
        gamma_1[i] = (char)gam[i];
    }

    cout << "------------------------------------------------------------" << endl << "Part 1" << endl;

    cout << "    Text for single gamming: \n" << text << endl;

    cout << "    Bitset of the Text :\n";
    print_bitset_text(text);
    cout << endl;

    cout << "    Bitset of the given key (gamma_1) :\n";
    print_bitset_text(gamma_1);
    cout << endl;

    cout << "    Ciphertext computation...\n";
```

Figure 3.2: Программа (2)

```
68
69      // Convert text to ciphertext
70      for (unsigned int i = 0; i < strlen(text); i++) {
71          gam_text[i] = text[i] ^ gamma_1[i];
72      }
73      cout << "    Bitset of the Ciphertext (gam_text) :\n";
74      print_bitset_gamma(gam_text, text);
75      cout << endl;
76      cout << "-----------------------------------------------------------------" << endl;
77      cout << endl << "-----------------------------------------------------------------" << endl << "Part 2" << endl;
78      cout << "    Target text for single gamming: \n" << target_text << endl;
79      cout << "    Bitset of the Target text :\n";
80      print_bitset_text(target_text);
81      cout << endl;
82      cout << "    Key computation...\n";
83
84      // Find key to obtain target_text from text
85      for (unsigned int i = 0; i < strlen(target_text); i++) {
86          gamma_2[i] = gam_text[i] ^ target_text[i];
87      }
88
89      cout << "    Bitset of the obtained key (gamma_2) :\n";
90      print_bitset_gamma(gamma_2, target_text);
91      cout << endl;
92
93      cout << "    Check of correct computation work...\n    Obtained target text after single gamming" << endl;
94      for (unsigned int i = 0; i < strlen(text); i++) {
95          cout << static_cast<char>(bitset<n>((unsigned char)(gam_text[i] ^ gamma_2[i])).to_ulong() + 256);
96      }
97
98      cout << endl << "-----------------------------------------------------------------" << endl;
99  }
100
```

Figure 3.3: Программа (3)

```
-----------------------------------------------------------------
Part 1
    Text for single gamming:
Stirlis - Vy Geroy!!
    Bitset of the Text :
01010011 01110100 01101001 01110010 01101100 01101001 01110011 00100000 00101101 00100000 01010110 01111001 00100000 01000111 01100101 01110010 01101111 01111001 00100001 00100001
    Bitset of the given key (gamma_1) :
00000101 00001100 00010111 01111111 00001110 01001110 00110111 11010010 10010100 00010000 00001001 00101110 00100010 01010111 11111111 11001000 00001011 10110010 01110000 01010100
    Ciphertext computation...
    Bitset of the Ciphertext (gam_text) :
01010110 01111000 01111110 00001101 01100010 00100111 01000100 11110010 10111001 00110000 01011111 01010111 00000010 00010000 10011010 10111010 01100100 11001011 01010001 01110101
-----------------------------------------------------------------

-----------------------------------------------------------------
Part 2
    Target text for single gamming:
S Novym Godom drysya
    Bitset of the Target text :
01010011 00100000 01001110 01101111 01110110 01111001 01101101 00100000 01000111 01101111 01100100 01101111 01101101 00100000 01100100 01110010 01111001 01110011 01111001 01100001
    Key computation...
    Bitset of the obtained key (gamma_2) :
00000101 01011000 00110000 01100010 00010100 01011110 00101001 11010010 11111110 01011111 00111011 00111000 01101111 00110000 11111110 11001000 00011101 10111000 00101000 00010100
    Check of correct computation work...
    Obtained target text after single gamming
S Novym Godom drysya
-----------------------------------------------------------------
```

Figure 3.4: Вывод работы программы

## 3.1 Контрольные вопросы

1. Поясните смысл однократного гаммирования. – Дан открытый текст и ключ для шифрования открытого текста, с помощью клуча можно как шифровать открытый текст для получения шифротекста, так и дешифровать шифротекст для получения открытого текста.

2. Перечислите недостатки однократного гаммирования. – Необходимость иметь огромные объемы данных, которые можно было бы использовать в качестве гаммы.

3. Перечислите преимущества однократного гаммирования. – 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.

4. Почему длина открытого текста должна совпадать с длиной ключа? – Потому что в данном методе для каждого символа открытого текста предполагается символ ключа, наложение которорого приводит к шифровке символа открытого текста.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности? – Сложения по модулю 2 (XOR). Метод шифрования является симметричным, так как двойное при- бавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

6. Как по открытому тексту и ключу получить шифротекст? – Шифротекст = открытый текст XOR ключ.

7. Как по открытому тексту и шифротексту получить ключ? – Ключ = открытый текст XOR шифротекст.

8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра? – Необходимые и достаточные условия абсолютной стойкости шифра: полная случайность ключа; равенство длин ключа и открытого текста; однократное использование ключа.

# 4 Выводы

Освоено на практике применение режима однократного гаммирования.

# 5 Список литературы

1. Методические материалы курса