

Отчет по лабораторной работе №7

по дисциплине: Информационная безопасность

Ким Михаил Алексеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Создание программы	7
5	Анализ результатов	15
6	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Однократное гаммирование на C++. Листинг. 1	12
4.2	Однократное гаммирование на C++. Листинг. 2	13
4.3	Однократное гаммирование на C++. Листинг. 3	13
4.4	Компиляция и выполнение программы	14

1 Цель работы

Освоить на практике применение режима однократного гаммирования.

2 Задание

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.

3 Теоретическое введение

- Терминал (или «Bash», сокращение от «Bourne-Again shell») — это программа, которая используется для взаимодействия с командной оболочкой. Терминал применяется для выполнения административных задач, например: установку пакетов, действия с файлами и управление пользователями. [1]
- Гаммирование, или Шифр XOR, — 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. Суммирование обычно выполняется в каком-либо конечном поле. Например, в поле Галуа суммирование принимает вид операции «исключающее ИЛИ (XOR)» [2].

4 Выполнение лабораторной работы

4.1 Создание программы

1. Напишем программу на C++, удовлетворяющую всем условиям задания. Программа будет содержать четыре вспомогательные функции: вывод информации в 16-ричной системе счисления, кодирование и декодирование информации, нахождение ключа по исходному тексту и шифротексту. (рис. 4.1, 4.2, 4.3).

```
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

void printHex(vector<unsigned char> vec)
{
    cout << std::hex << std::uppercase;
```

```

    for (size_t i = 0; i < vec.size(); i++)
    {
        cout << static_cast<short>(vec[i]) << ' ';
    }

    cout << std::dec << std::nouppercase << endl;
}

vector<unsigned char> encryptGamming(const vector<unsigned char>& key, const
{
    if (key.size() != pureText.size())
    {
        cout << "Key length and text lenght are not the same!" << endl;
        return vector<unsigned char>('0');
    }

    vector<unsigned char> encryptedText(key.size());

    for (size_t i = 0; i < key.size(); i++)
    {
        encryptedText[i] = pureText[i] ^ key[i];
    }

    return encryptedText;
}

string decryptGamming(const vector<unsigned char>& key, vector<unsigned char>
{

```



```

    if (key.size() != encryptText.size())
    {
        cout << "Key length and encrypt text lenght are not the same!" << endl;
        return "0";
    }

    string decryptedText(key.size(), '0');

    for (size_t i = 0; i < key.size(); i++)
    {
        decryptedText[i] = encryptText[i] ^ key[i];
    }

    return decryptedText;
}

vector<unsigned char> findKey(vector<unsigned char>& encryptText, const string& pureText)
{
    if (encryptText.size() != pureText.size())
    {
        cout << "Pure text length and encrypt text lenght are not the same!" << endl;
        return vector<unsigned char>('0');
    }

    vector<unsigned char> key(encryptText.size());

    for (size_t i = 0; i < encryptText.size(); i++)
    {
        key[i] = encryptText[i] ^ pureText[i];
    }
}

```

```

    }

    return key;
}

int main()
{
    std::setlocale(LC_ALL, "ru");

    vector<unsigned char> exampleKey = {
        0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,
        0x09, 0x2E, 0x22, 0x57, 0xFF, 0xC8, 0x0B, 0xB2, 0x70, 0x54
    };

    string examplePureText("Штирлиц - Вы Герой!!");

    cout << "Example key:" << endl;
    printHex(exampleKey);
    cout << "\nExample decrypted message:\n" << examplePureText << endl;

    vector<unsigned char> encryptedMessage = encryptGamming(exampleKey, exam
    cout << "\nCalculated encrypted message:" << endl;
    printHex(encryptedMessage);

    cout << "\nCalculated decrypted message:" << endl;
    cout << decryptGamming(exampleKey, encryptedMessage) << endl << endl;;
}

```

```

vector<unsigned char> wrongKey = {
    0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,
    0x09, 0x2E, 0x22, 0x55, 0xF4, 0xD3, 0x07, 0xBB, 0xBC, 0x54
};

cout << "\nWrong key:" << endl;
printHex(wrongKey);
cout << "\nDecrypted message from wrong key:\n" << decryptGamming(wrongKey, encryptedMessage);

string wonderedText("Я очень люблю C++!!!");
cout << "\nWondered message:\n" << wonderedText << endl;

vector<unsigned char> predictedKey = findKey(encryptedMessage, wonderedText);
cout << "\nPredicted key for wondered message:" << endl;
printHex(predictedKey);

string calculatedWonderedText = decryptGamming(predictedKey, encryptedMessage);
cout << "\nDecrypted message from this key:\n" << calculatedWonderedText;

string taskMessage("С Новым Годом, друзья");
cout << "\nKey for message '" << taskMessage << "': " << endl;
printHex(findKey(encryptedMessage, taskMessage));

return EXIT_SUCCESS;
}

```

```

1  #include <cstdlib>
2  #include <iostream>
3  #include <vector>
4  #include <string>
5
6  using std::cin;
7  using std::cout;
8  using std::endl;
9  using std::string;
10 using std::vector;
11
12
13 void printHex(vector<unsigned char> vec)
14 {
15     cout << std::hex << std::uppercase;
16
17     for (size_t i = 0; i < vec.size(); i++)
18     {
19         cout << static_cast<short>(vec[i]) << ' ';
20     }
21
22     cout << std::dec << std::nouppercase << endl;
23 }
24
25 vector<unsigned char> encryptGamming(const vector<unsigned char>& key, const string& pureText)
26 {
27     if (key.size() != pureText.size())
28     {
29         cout << "Key length and text lenght are not the same!" << endl;
30         return vector<unsigned char>('0');
31     }
32
33     vector<unsigned char> encryptedText(key.size());
34
35     for (size_t i = 0; i < key.size(); i++)
36     {
37         encryptedText[i] = pureText[i] ^ key[i];
38     }
39
40     return encryptedText;
41 }
42
43 string decryptGamming(const vector<unsigned char>& key, vector<unsigned char>& encryptText)
44 {
45     if (key.size() != encryptText.size())
46     {
47         cout << "Key length and encrypt text lenght are not the same!" << endl;
48         return "0";
49     }
50
51     string decryptedText(key.size(), '0');
52
53     for (size_t i = 0; i < key.size(); i++)
54     {
55         decryptedText[i] = encryptText[i] ^ key[i];
56     }
57
58     return decryptedText;
59 }

```

Рис. 4.1: Однократное гаммирование на C++. Листинг. 1

```

61 vector<unsigned char> findKey(vector<unsigned char>& encryptText, const string& pureText)
62 {
63     if (encryptText.size() != pureText.size())
64     {
65         cout << "Pure text length and encrypt text lenght are not the same!" << endl;
66         return vector<unsigned char>{'0'};
67     }
68     vector<unsigned char> key(encryptText.size());
69     for (size_t i = 0; i < encryptText.size(); i++)
70     {
71         key[i] = encryptText[i] ^ pureText[i];
72     }
73     return key;
74 }
75
76 int main()
77 {
78     std::setlocale(LC_ALL, "ru");
79
80     vector<unsigned char> exampleKey = {
81         0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,
82         0x09, 0x2E, 0x22, 0x57, 0xFF, 0xC8, 0xBB, 0xB2, 0x70, 0x54
83     };
84     string examplePureText("Штирлиц - Вы Герой!!");
85
86     cout << "Example key:" << endl;
87     printHex(exampleKey);
88     cout << "\nExample decrypted message:\n" << examplePureText << endl;
89
90     vector<unsigned char> encryptedMessage = encryptGammig(exampleKey, examplePureText);
91     cout << "\nCalculated encrypted message:" << endl;
92     printHex(encryptedMessage);
93
94     cout << "\nCalculated decrypted message:" << endl;
95     cout << decryptGammig(exampleKey, encryptedMessage) << endl << endl;
96
97     vector<unsigned char> wrongKey = {
98         0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,
99         0x09, 0x2E, 0x22, 0x55, 0xF4, 0xD3, 0x07, 0xBB, 0xBC, 0x54
100     };
101
102     cout << "\nWrong key:" << endl;
103     printHex(wrongKey);
104     cout << "\nDecrypted message from wrong key:\n" << decryptGammig(wrongKey, encryptedMessage) << endl << endl;
105
106     string wonderedText("Я очень люблю C++!!!");
107     cout << "\nWondered message:\n" << wonderedText << endl;
108
109     vector<unsigned char> predictedKey = findKey(encryptedMessage, wonderedText);
110     cout << "\nPredicted key for wondered message:" << endl;
111     printHex(predictedKey);
112
113     string calculatedWonderedText = decryptGammig(predictedKey, encryptedMessage);
114     cout << "\nDecrypted message from this key:\n" << calculatedWonderedText << endl;
115 }

```

Рис. 4.2: Однократное гаммирование на C++. Листинг. 2

```

122 string taskMessage("С Новым Годом, друзья");
123 cout << "\nKey for message '" << taskMessage << "': " << endl;
124 printHex(findKey(encryptedMessage, taskMessage));
125
126 return EXIT_SUCCESS;
127 }
128

```

Рис. 4.3: Однократное гаммирование на C++. Листинг. 3

2. Результат работы программы после компиляции и выполнения (рис. 4.4).

```
Example key:
5 C 17 7F E 4E 37 D2 94 10 9 2E 22 57 FF C8 B B2 70 54

Example decrypted message:
Штирлиц - Вы Герой!!

Calculated encrypted message:
DD FE FF 8F E5 A6 C1 F2 2 30 CB D5 2 94 1A 38 E5 5B 51 75

Calculated decrypted message:
Штирлиц - Вы Герой!!

Wrong key:
5 C 17 7F E 4E 37 D2 94 10 9 2E 22 55 F4 D3 7 BB BC 54

Decrypted message from wrong key:
Штирлиц - Вы Болван!

Wondered message:
Я очень люблю C++!!!

Predicted key for wondered message:
2 DE 11 78 0 4B 3D D2 E9 CE 2A 3E FC B4 59 13 CE 7A 70 54

Decrypted message from this key:
Я очень люблю C++!!!

Key for message 'С Новым Годом, друзья':
C DE 32 61 7 5D 2D D2 C1 DE 2F 3B EE B8 FE C8 16 BC AD 8A
```

Рис. 4.4: Компиляция и выполнение программы

5 Анализ результатов

Работа выполнена без каких-либо серьезных нареканий. Единственным моментом, которому было уделено больше внимания, чем он того заслуживает, стал выбор типа данных для хранения текста, шифротекста и ключа.

6 Выводы

Освоено на практике применение режима однократного гаммирования.

Список литературы

1. Терминал Linux [Электронный ресурс]. URL: [%7Bhttps://www.reg.ru/blog/linux-shpargalka-komandy-terminala-dlya-novichkov/%7D](https://www.reg.ru/blog/linux-shpargalka-komandy-terminala-dlya-novichkov/).
2. Гаммирование [Электронный ресурс]. Wikipedia Inc. URL: https://en.wikipedia.org/wiki/XOR_cipher.