

Презентация по лабораторной работе №7

Элементы криптографии. Однократное гаммирование

Танрибергенов Э.

2024 г.

Российский университет дружбы народов, Москва, Россия

Информация

- Танрибергенов Эльдар
- студент 4 курса из группы НПИбд-02-21
- ФМиЕН, кафедра прикладной информатики и теории вероятностей
- Российский университет дружбы народов

Цели и задачи

Освоить на практике применение режима однократного гаммирования.

Задания

- 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.

Результаты

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» 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.

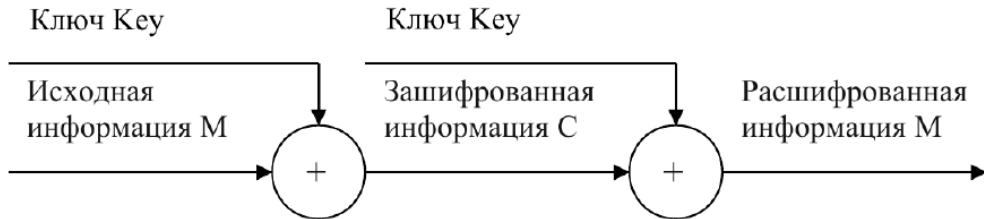


Рис. 1: Схема однократного использования Вернама

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

- Программа написана на языке программирования C++
- Написаны функции: кодирования в 16-ричный код, декодирования, гаммирования и генерации ключа
- Функции размещены в отдельном файле и подключаются при помощи *#include*

```
// Кодировет текст в шестнадцатеричную последовательность
string encode_hex(string norm_txt){
    string hex_txt;
    stringstream ss;
    string::const_iterator cii;
    unsigned char c;
    int cnum;
    char *chr_norm_txt = new char[norm_txt.length()];

    for(cii=norm_txt.begin(); cii!=norm_txt.end(); cii++){
        c = *cii;
        cnum = int(c);
        ss << hex << cnum;
    }
    hex_txt = ss.str();

    return hex_txt;
}
```

Рис. 2: Функция кодирования текста в шестнадцатеричный код

Написание программы

```
// Декодирует шестнадцатеричную последовательность в текст
string decode_hex(string hex_txt){
    string norm_txt, tmp=" ";
    stringstream ss;
    string::const_iterator cii;
    char *chr_hex_txt = new char[hex_txt.length()];
    int n=0;

    for(cii=hex_txt.begin(); cii!=hex_txt.end(); cii++){
        tmp += *cii;
        chr_hex_txt[n] = *cii;
        n++;
    }
    hex_txt = tmp;

    for(int i=0; i<n; i+=2){
        tmp=" ";
        for(int j=i; j<i+2; j++){
            tmp += chr_hex_txt[j];
            ss << char(stoi(tmp,nullptr,16));
        }
        norm_txt = ss.str();

        delete [] chr_hex_txt;
        chr_hex_txt = nullptr;

        return norm_txt;
    }
}
```

Рис. 3: Функция декодирования шестнадцатеричного кода в текст

```
// Гаммирование

string one_time_gamming(string hex_txt, string key){

    string gammed_txt="", str_xor, hex1, hex2;
    stringstream ss;
    bitset<8> bin_xor;
    int int_xor, int_sumnd1, int_sumnd2;
    string::const_iterator cii, cij;
    char *chr_message = new char[key.length()];
    char *chr_key = new char[key.length()];
    int n=0, m=0;

    for (cii=hex_txt.begin(); cii!=hex_txt.end(); cii++){
        chr_message[n] = *cii;
        n++;
    }
    for (cij=key.begin(); cij!=key.end(); cij++){
        chr_key[m] = *cij;
        m++;
    }
}
```

Рис. 4: Функция однократного гаммирования

```
for(int i=0; i<n; i+=2){
    hex1 = "";
    hex2 = "";
    for(int j=i; j<i+2; j++)
    {
        hex1 += chr_message[j];
        hex2 += chr_key[j];
    }
    int_sumnd1 = stoi(hex1, nullptr, 16);
    int_sumnd2 = stoi(hex2, nullptr, 16);
    bin_xor = bitset<8>(int_sumnd1) ^ bitset<8>(int_sumnd2);
    str_xor = bin_xor.to_string();
    int_xor = stoi(str_xor, nullptr, 2);
    if(int_xor < 16)
        ss << hex << 0 << int_xor;
    else
        ss << hex << int_xor;
}
gammed_txt = ss.str();

delete [] chr_message;
delete [] chr_key;
chr_message = nullptr;
chr_key = nullptr;

return gammed_txt;
}
```

Рис. 5: Функция однократного гаммирования


```
// Генерация ключа

string key_gen(string text, int rand_num){

    string key;
    stringstream ss;
    string::const_iterator cii;

    for(cii=text.begin(); cii!=text.end(); cii++)
        ss << hex << (rand()%100 + rand_num)%16;S
    key = ss.str();

    return key;
}
```

Рис. 6: Функция генерации ключа

Написание программы

```
#include <iostream>
#include <string>
#include "functions.h"
using namespace std;

int main(){

    srand(time(NULL));
    string open_txt, hex_txt, key, encrypted_txt, decrypted_txt, decoded_txt;

    open_txt = "Yes";
    cout << "\n Исходное сообщение:\n\n " << open_txt << endl;
    hex_txt = encode_hex(open_txt);
    cout << " " << hex_txt << endl;

    key = key_gen(hex_txt, rand()%10);
    cout << "\n Ключ:\n " << key << endl;

    encrypted_txt = one_time_gamming(hex_txt, key);
    cout << "\n Зашифрованный текст:\n " << encrypted_txt << endl;
```

Рис. 7: Запускающая программу функция

Написание программы

```
key = key_gen(encrypted_txt, rand()%10);

cout << "\n Расшифровка текста...\n\n Пробный ключ:\n " << key << endl;
decrypted_txt = one_time_gamming(encrypted_txt, key);
decoded_txt = decode_hex(decrypted_txt);
cout << "\n Расшифрованное сообщение:\n\n " << decoded_txt << endl << endl;

cout << " Подбор ключа..." << endl;

do{
    key = key_gen(encrypted_txt, rand()%10);
    decrypted_txt = one_time_gamming(encrypted_txt, key);
    decoded_txt = decode_hex(decrypted_txt);

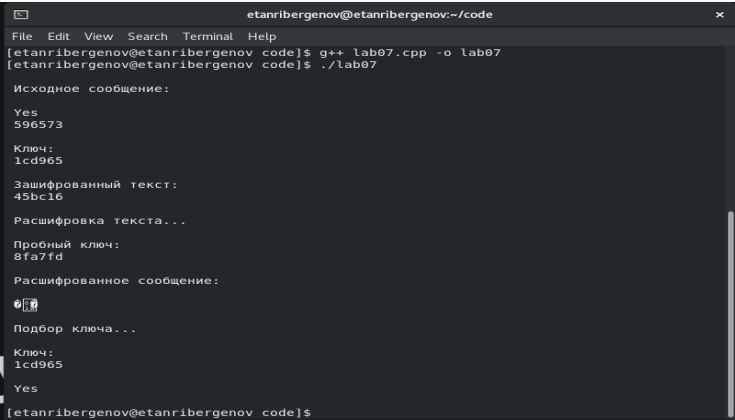
}while(decoded_txt!=open_txt);

    cout << "\n Ключ:\n " << key << endl;
    cout << "\n " << decoded_txt << endl << endl;

return 0;
}
```

Рис. 8: Действия программы

- В качестве открытого текста выбрано маленькое слово “Yes”



```
etanribergenov@etanribergenov:~/code
File Edit View Search Terminal Help
[etanribergenov@etanribergenov code]$ g++ lab07.cpp -o lab07
[etanribergenov@etanribergenov code]$ ./lab07

Исходное сообщение:

Yes
596573

Ключ:
1cd965

Зашифрованный текст:
45bc16

Расшифровка текста...

Пробный ключ:
8fa7fd

Расшифрованное сообщение:

000

Подбор ключа...

Ключ:
1cd965

Yes
[etanribergenov@etanribergenov code]$
```

Рис. 9: Результат

Вывод

В результате выполнения работы я освоил на практике применение режима однократного гаммирования.