Отчет по лабораторной работе №7

Дисциплина: Информационная безопасность

Выполнила Дяченко Злата Константиновна, НФИбд-03-18

Содержание

List of Figures

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 Теоретические вводные данные

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 между элементами гаммы и элементами подлежащего сокрытию текста. 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. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины. Если известны шифротекст и открытый текст, то, чтобы найти ключ, обе части равенства

необходимо сложить по модулю 2 с Pi:

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

$$K_i = C_i \oplus P_i$$

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.

4 Выполнение лабораторной работы

4.1 Шаг 1

Писать программу решила на языке C++. Были написаны функции, показанные на Рисунке 1 (рис. 4.1). Функция convert преобразует символы открытого текста в их коды в ASCCI. Функция printM выводит массив данных (в нашем случае коды символов открытого текста, зашифрованный текст, ключ). Функция keyGen генерирует случайный ключ определенной длины (я генерировала ключ той же длины, что и открытый текст). Функция gammir осуществляет однократное гаммирование - в результате получаем шифротекст. Функция getKey получает ключ, зная шифротекст и открытый текст.

```
void convert(string letter, int array[])
{

    for (int i = 0; i < letter.length(); i++)
    {
        unsigned char x = letter.at(i);
        array[i]= int(x);
    }
}

void printM (int array[], int len){
    for (int j = 0; j < len; j++) {
        cout << setw(4) << array[j];
    }
    cout<<endl;
}

void keyGen (int key[], int len){
    srand(time(NULL));
    for (int i = 0; i < len; i++){
        key[i]= rand() % 256;
    }
}

void gammir(int text[], int key[], int len, int shifrotext[]){
    for (int i=0; i<len; i++){
        shifrotext[i]=text[i]^key[i];
    }
}

void getKey(int shifrotext[], int text[], int len, int key[]){
    for (int i=0; i<len; i++){
        key[i]=shifrotext[i]^text[i];
    }
}
```

Figure 4.1: Функции

4.2 Шаг 2

В функции main создаем переменную text с открытым текстом “С Новым Годом, друзья!” (рис. 4.2). Применяя описанные ранее функции, генерируем ключ и получаем шифротекст.

```
int main (){

    setlocale(LC_ALL, "Russian");
    string text;
    int len;
    text="С Новым Годом, друзья!";
    len=text.size();
    int shifr[len];
    convert(text, shifr);
    cout<<"Текст"<<endl;
    printM(shifr, len);

    int key[len];
    keyGen(key, len);
    cout<<"Ключ:"<<endl;
    printM(key, len);

    int shifrotext[len];
    gammir(shifr, key, len, shifrotext);
    cout<<"Шифротекст:"<<endl;
    printM(shifrotext, len);
```

Figure 4.2: Генерация ключа и осуществление однократного гаммирования

4.3 Шаг 3

Результат шифрования представлен на Рисунке 3 (рис. 4.3).

```
Текст
 209  32 205 238 226 251 236  32 195 238 228 238 236  44  32 228 240 243 231 252 255  33
Ключ:
  25 212 222 126  18 236 210  52 128  53 126 217 214 166  15 185 190 148 152 106  10 211
Шифротекст:
 200 244  19 144 240  23  62  20  67 219 154  55  58 138  47  93  78 103 127 150 245 242
```

Figure 4.3: Открытый текст, сгенерированный ключ и шифротекст

4.4 Шаг 4

Функцию getKey можно проверить, передав в нее полученный ранее шифротекст и тот открытый текст, который был зашифрован. Получим тот же самый ключ (рис. 4.5). Также можем найти ключ, который необходим для того, чтобы расшифровать полученный шифротекст в открытый текст “С Новой бедой, друзья!” (рис. 4.4)

```
    int newKey[len];
    getKey(shifrotext, shifr, len, newKey);
    cout<<"Ключ, полученный, зная шифротекст и открытый текст :"<<endl;
    printM(newKey, len);

    string text2= "С Новой бедой, друзья!";
    int shifr2[len];
    convert(text2, shifr2);
    cout<<"Текст, который хочу получить "<<endl;
    printM(shifr2, len);
    getKey(shifrotext, shifr2, len, newKey);
    cout<<"Ключ, который для этого нужно использовать (шифротекст не изменяется):"<<endl;
    printM(newKey, len);
```

Figure 4.4: Поиск ключа

4.5 Шаг 4

Результаты выполнения функций представлены на Рисунке 5 (рис. 4.5).

```
Ключ, полученный, зная шифротекст и открытый текст :
  25 212 222 126  18 236 210  52 128  53 126 217 214 166  15 185 190 148 152 106  10 211
Текст, который хочу получить
 209  32 205 238 226 238 233  32 225 229 228 238 233  44  32 228 240 243 231 252 255  33
Ключ, который для этого нужно использовать (шифротекст не изменяется):
  25 212 222 126  18 249 215  52 162  62 126 217 211 166  15 185 190 148 152 106  10 211
```

Figure 4.5: Полученный ключ

5 Выводы

В результате работы я освоила на практике применение режима однократного гаммирования. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.