Отчет по лабораторной работе “Реализация блочного шифра Магма”

Работу выполнила Смирнова Екатерина Андреевна, ККСО-03-17,1 курс

Задача: реализовать шифратор, который использует блочный шифр “Магма” на языке C и Python3.

Теоретическая часть:

Полное название — «ГОСТ 34.12-15 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования». Является примером DES-подобных криптосистем, созданных по классической итерационной схеме Фейстеля.

ГОСТ 34.12-15 — блочный шифр с 256-битным ключом и 32 циклами (называемыми раундами) преобразования, оперирующий 64-битными блоками. Основа алгоритма шифра — сеть Фейстеля. Выделяют четыре режима работы:

- простой замены
- гаммирование
- гаммирование с обратной связью
- режим выработки имитовставки.

В своей работе я использовала режим работы простой замены

Для зашифровывания в этом режиме 64-битный блок открытого текста сначала разбивается на две половины (младшие биты — A, старшие биты — B). На i-ом цикле

используется подключ K_i.

Для генерации подключей исходный 256-битный ключ разбивается на восемь 32-битных блоков: $K_1 \ldots K_8$.

Ключи $K_9 \ldots K_{24}$ являются циклическим повторением ключей $K_1 \ldots K_8$ (нумеруются от младших битов к старшим). Ключи $K_{25} \ldots K_{32}$ являются ключами $K_8 \ldots K_1$.

После выполнения всех 32 раундов алгоритма, блоки A_{33} и B_{33} склеиваются (обратите внимание, что старшим блоком становится A_{33}, а младшим — B_{33}) — результат есть результат работы алгоритма.

Расшифровывание выполняется так же, как и зашифровывание, но инвертируется порядок подключей K_i.

Функция шифрования(расшифровки) вычисляется следующим образом:

A_i и K_i складываются по модулю 2^{32}.

Результат разбивается на восемь 4-битовых подпоследовательностей, каждая из которых поступает на вход своего *узла таблицы замен* (в порядке возрастания старшинства битов), называемого ниже *S-блоком*. Общее количество S-блоков стандарта — восемь, то есть столько же, сколько и подпоследовательностей. Каждый *S-блок* представляет собой перестановку чисел от 0 до 15 (конкретный вид S-блоков в стандарте не определен). Первая 4-битная подпоследовательность попадает на вход первого S-блока, вторая — на вход второго и т. д.

Если узел *S-блока* выглядит так:

```
1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12
```

и на входе S-блока 0, то на выходе будет 1, если 4, то на выходе будет 5, если на входе 12, то на выходе 6 и т. д.

Выходы всех восьми S-блоков объединяются в 32-битное слово, затем всё слово циклически сдвигается влево (к старшим разрядам) на 11 битов.

Режим простой замены имеет следующие недостатки:

- Может применяться только для шифрования открытых текстов с длиной, кратной 64 бит
- При шифровании одинаковых блоков открытого текста получаются одинаковые блоки шифротекста, что может дать определенную информацию криптоаналитику.

Таким образом, применение «Магмы» в режиме простой замены желательно лишь для шифрования ключевых данных.

Практическая часть:

В моем коде на языке C использовано 3 функции. На языке Python3 использованы идентичные методы, но с более легкой конструкцией, так как можно было избежать использование функций.

Первая функция-fillkey (). С ее помощью пользователь может ввести свой ключ, а функция выполнит разбиения ключа на итерационные ключи Ki, i = 1, 2, …, 32

```
void fillkey(){
   long int key,k0,k1,k2,k3,k4,k5,k6;
   fscanf(stdin, "%08x %08x %08x %08x %08x %08x %08x %08x",&key,&k0,&k1,&k2,&k3,&k4,&k5,&k6);
   K[0]=key;K[1]=k0;K[2]=k1;K[3]=k2;K[4]=k3;K[5]=k4;K[6]=k5;K[7]=k6;
   K[8]=key;K[9]=k0;K[10]=k1;K[11]=k2;K[12]=k3;K[13]=k4;K[14]=k5;K[15]=k6;
   K[16]=key;K[17]=k0;K[18]=k1;K[19]=k2;K[20]=k3;K[21]=k4;K[22]=k5;K[23]=k6;
   K[24]=k6;K[25]=k5;K[26]=k4;K[27]=k3;K[28]=k2;K[29]=k1;K[30]=k0;K[31]=key;
}
```

Вторая функция-fest(&a,&b,i); Принимает номер раунда и исходные данные. Далее выполняет алгоритм шифрования или расшифрования. В данной функции присутствуют сложение по модулю 32, выполнение замены символов по специальной таблице, циклический сдвиг влево на 11 битов, и если раунд не равен 31, то a и b меняются местами.

Ниже приведена таблица подстановок

```
int
table[8][16]=
                {
                  12,   4,   6,   2,   10,   5,  11,   9,  14,   8,  13,   7,   0,   3,
                15,   1,
                   6,   8,   2,   3,    9,  10,   5,  12,   1,  14,   4,   7,  11,  13,
                0,  15,
                  11,   3,   5,   8,    2,  15,  10,  13,  14,   1,   7,   4,  12,   9,
                6,   0,
                  12,   8,   2,   1,   13,   4,  15,   6,   7,   0,  10,   5,   3,  14,
                9,  11,
                   7,  15,   5,  10,    8,   1,   6,  13,   0,   9,   3,  14,  11,   4,
                2,  12,
                   5,  13,  15,   6,    9,   2,  12,  10,  11,   7,   8,   1,   4,   3,
                14,   0,
                   8,  14,   2,   5,    6,   9,   1,  12,  15,   4,  11,   0,  13,  10,
                3,   7,
                   1,   7,  14,  13,    0,   5,   8,   3,   4,  15,  10,   6,   9,  12,
                11,   2
                };
```

Третья функция-getkey(i). Принимает номер раунда и выбирает подходящий итерационный ключ.

```
int getkey(int i){
   if (mode=='1')
      x=K[(i)];
   if (mode=='2')
      x=K[(31-i)];
}
```

Mode-режим работы, при значении равном единице будет выполняться шифрование данных, а при двойке-расшифровка.

Заключение: поставленная задача успешно выполнена, разработан программный комплекс, с помощью которого можно реализовать шифратор на основе блочного шифра «Магма»