ВОРОНЕЖСКИЙ ИНСТИТУТ ВЫСОКИХ ТЕХНОЛОГИЙ – АНОО ВПО

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

По дисциплине «Защита информации»

Лабораторная работа № 5

«ШИФРАЦИЯ МЕТОДАМИ ЗАМЕНЫ И ПЕРЕСТАНОВКИ»

Цель работы: изучить основные методы криптографической защиты данных, типы шифров. Программно реализовать один из описанных алгоритмов шифрации методами перестановки и замены.

Три основных типа шифров.

Среди традиционных методов кодирования имеются два базовых типа - перестановка (transposition) и замена (substitution). Шифры, использующие перестановку, «перемешивают» символы сообщения по определенному правилу. Шифр замены замещает одни символы другими, но сохраняет порядок их следования в сообщении. Оба метода могут быть доведены до любой степени сложности. Кроме того, на их основе можно создать комплексный метод, сочетающий черты каждого из них. Появление компьютеров добавило к существующим двум методам еще один, называемый битовой манипуляцией (bit manipulation). Этот метод изменяет компьютерное представление данных по определенному алгоритму.

Все три метода при желании могут использовать ключ (key). Как правило, ключ представляет собой строку символов, необходимую для того, чтобы декодировать сообщение. Однако, не стоит путать ключ с методом шифрования, поскольку наличие ключа является необходимым, но недостаточным условием успешной расшифровки сообщения. Кроме знания ключа, необходимо знать и алгоритм шифрации. Назначение ключа состоит в «персонализации» сообщения, с тем чтобы прочитать его могли только те, кому оно предназначено, даже несмотря на то, что применяемый для шифрования алгоритм широко известен.

Необходимо научиться различать два базовых понятия - открытый текст (plain text) и шифрованный текст (cipher text). Информация, переданная открытым текстом, представляет собой читаемое сообщение, шифрованный текст представляет собой закодированную версию сообщения.

Шифры замены

Шифр замены представляет собой метод шифрования сообщения путем замены одних символов другими на регулярной основе. Одной из простейших форм такого шифра является циклический сдвиг алфавита на определенное количество символов. Например, если латинский алфавит сдвинуть на три символа, то вместо

abcdefghijklmnopqrstuvwxyz

получим

defghijklmnopqrstuvwxyzabc

Таким образом, a превращается в b, b - в e, и т.д. Обратите внимание, что буквы «abc», находившиеся в начале алфавита, переместились в конец. Для того, чтобы закодировать сообщение, пользуясь этим методом, нужно просто заменить нормальный алфавит на его смещенную версию.

Вышеприведенный алгоритм, основанный на постоянном сдвиге алфавита сможет обмануть разве что совсем неопытного взломщика, поскольку взламывается исключительно просто. В конце концов, если есть всего 26 возможных вариантов сдвига, и все их можно перебрать за сравнительно короткое время вручную. Лучшим вариантом по сравнению с вышеприведенным является использование неупорядоченного алфавита, а не просто сдвига. Еще одним недостатком метода простого постоянного сдвига является то, что сохраняются на своих местах пробелы между словами. Это еще более упрощает задачу взломщика, поэтому пробелы также следует кодировать (еще лучше будет кодировать и знаки препинания). Например, можно задать следующее соответствие строк, одна из которых содержит упорядоченный алфавит, а вторая, задающая преобразование, - его рандомизированную версию:

abcdefghijklmnopqrstuvwxyz<пробел>

qazwsxedcrfvtgbyhnujm ikolp

Дает ли эта рандомизированная версия существенное улучшение по сравнению с предыдущей версией, использовавшей простой постоянный сдвиг ? Ответ будет утвердительным, так как теперь имеется 26! (огромное число $\approx 4 \times 10^{26}$) способов упорядочивания алфавита, а с учетом пробела это число возрастет до 27! ($\approx 1 \times 10^{28}$).

Следует отметить, что даже этот улучшенный алгоритм шифрования с заменой может быть с легкостью взломан при использовании частотных таблиц английского языка, в которых содержится частотная информация по каждой букве алфавита. Далее, чем больше объем кодированного сообщения, тем проще расшифровать его с помощью частотных таблиц. Для того, чтобы замедлить процесс расшифровки сообщения взломщиком, применяющим частотные таблицы, можно воспользоваться шифром со множественными заменами (multiple substitution cipher). В этом случае одна и та же буква открытого текста не обязательно будет преобразовываться в одну и ту же букву зашифрованного сообщения. Этого можно добиться, включив второй рандомизированный алфавит и переключаясь между ними по заранее предопределенному методу (например, при встречающемуся в тексте пробелу).

Этот подход реализует нижеприведенная программа. В качестве второго рандомизированного алфавита используем poi uytrewqasdfghjklmnbvcxz:

```
// Шифр со множественными заменами
#include <iostream.h>
```

```
#include <fstream.h>
#include <ctype.h>
#include <<stdlib.h>

const int SIZE = 28;

void encode (char *input, char *output);
void decode (char *input, char *output);
int find(char *s, char ch);
char sub[SIZE] = "qazwsxedcrfvtgbyhnujm iklop";
char sub2[SIZE] = "poi uytrewqasdfghjklmnbvcxz";
char alphabet[SIZE] = "abcdefghijklmnopqrstuvwxyz";

main(int argc, char *argv[])
{
      if(argc != 4){
            cout << "Usage: input output encode/decode \n";
            exit(1);
      }
      if(toupper(*argv[3]) == 'E')
           encode(argv[1], argv[2]);
      else
           decode(argv[1], argv[2]);
      return 0;
}

// Encode
void encode(char *input, char *output)
{
 int ch, change;
 ifstream in(input, ios :: out | ios :: binary);
 ofstream out(output, ios :: out | ios :: binary);

 if (!in) {
   cout << "Cannot open input file.\n";
   exit(1);
 }

 if (!out) {
   cout << "Cannot open output file.\n";
   exit(1);
 }
 change = 1;
 do {
   ch = in.get();
```

```
   ch = tolower(ch);
   if(isalpha(ch))
            if(change)
                  ch = sub[find(alphabet, ch)];
            else
                  ch = sub2[find(alphabet, ch)];
   if(!in.eof()) out.put((char) ch);
   if(ch == ' ') change = !change;
 } while(!in.eof());
 in.close();
 out.close();
}
// Decode
void decode(char *input, char *output)
{
 int ch, change;
 ifstream in(input, ios :: out | ios :: binary);
 ofstream out(output, ios :: out | ios :: binary);

 if (!in) {
   cout << "Cannot open input file.\n";
   exit(1);
 }

 if (!out) {
   cout << "Cannot open output file.\n";
   exit(1);
 }
 change = 1;
 do {
   ch = in.get();
   ch = tolower(ch);
   if(isalpha(ch))
            if(change)
                  ch = alphabet[find(sub, ch)];
            else
                  ch = alphabet[find(sub2, ch)];
   if(!in.eof()) out.put((char) ch);
   if(ch == ' ') change = !change;
 } while(!in.eof());
 in.close();
 out.close();
}
// Find index
find (char *s, char ch)
```

```
{
 register int t;
 for(t=0;t<SIZE;t++) if(ch==s[t]) return t;
 return -1;
}
```

При использовании шифрования со множественными заменами взломать шифр с помощью частотных таблиц становится намного сложнее. С помощью нескольких рандомизированных алфавитов и совершенного механизма переключения между ними можно добиться построения такого алгоритма, в результате применения которого все алфавитные символы будут появляться с одинаковой частотой. При взломе такого алгоритма частотные таблицы языка будут практически бесполезны.

Задания на лабораторную работу:

1. Составить программу осуществляющую шифрацию/дешифрацию текста пользователя методом перестановки.
2. Составить программу осуществляющую шифрацию/дешифрацию текста пользователя методом замены. Сравнить скорость работы первой и второй программы.

Лабораторная работа № 6

«ШИФРАЦИЯ МЕТОДАМИ БИТОВЫХ МАНИПУЛЯЦИЙ»

Цель работы: изучить методы криптографической защиты данных на основе битовых манипуляций. Познакомиться с программами, реализующими шифрацию файлов (DISKREET, PGP и т.д.). Программно реализовать один из алгоритмов шифрации методами битовых манипуляций.

Шифры битовых манипуляций.

Методы шифрования, приводимые в предыдущей работе представляют собой компьютеризированные версии шифрования, ранее выполнявшегося вручную. Однако, компьютерные технологии дали начало новому методу кодирования сообщений путем манипуляций с битами, составляющими фактические символы нешифрованного сообщения. Как правило, современные компьютеризированные шифры попадают в класс, называемый шифрами бытовых манипуляций (bit manipulating ciphers). Хотя ревнители чистоты теории могут спорить о том, что такие шифры представляют собой просто вариацию шифров методом замены, большинство специалистов соглашается с тем, что концепции и методы, лежащие в основе шифров битовых манипуляций отличаются от всего, что было известно ранее, настолько значительно, что заслуживают выделения в особый класс.

Шифры битовых манипуляций популярны по двум причинам. Во-первых, они идеально подходят для использования в компьютерной криптографии, так как используют операции, которые легко выполняются системой. Вторая причина заключается в том, что полученный на выходе зашифрованный текст выглядит абсолютно нечитаемым - фактически полной бессмыслицей. Это положительно сказывается на безопасности и защищенности, так как важные данные маскируются под поврежденные файлы, доступ к которым просто никому не нужен.

Как правило шифры битовых манипуляций применимы только к компьютерным файлам и не могут использоваться для бумажных копий зашифрованных сообщений. Причина этого заключается в том, что манипуляции с битами имеют тенденцию генерировать непечатаемые символы. Поэтому мы всегда будем полагать, что текст, зашифрованный с помощью битовых манипуляций, всегда будет оставаться в виде электронного документа.

Шифры битовых манипуляций переводят открытый текст в шифрованный с помощью преобразования набора бит каждого символа по определенному алгоритму, используя одну из следующих логических операций или их комбинацию:

AND OR NOT XOR

Простейший (и наименее защищенный) шифр, манипулирующий с битами, использует только оператор первого дополнения. Этот оператор

инвертирует все биты, входящие в состав байта. Таким образом, все нули становятся единицами и наоборот. Поэтому байт, над которым дважды проведена такая операция, принимает исходное значение.

В действительности с этой простой схемой кодирования связаны две основные проблемы. Во-первых, программа шифрования для расшифровки текста не использует ключа. Поэтому любой, кто знает, что используется данный алгоритм и в состоянии написать программу, сможет прочитать файл. Во-вторых (и это самое главное), этот метод отнюдь не тайна для опытных программистов.

Улучшенный метод шифрования методом побитовой манипуляции использует оператор XOR. Результаты выполнения этого оператора приведены в следующей таблице:

XOR	1	0
1	0	1
0	1	0

Иными словами, результат выполнения оператора XOR получает значение ИСТИНА тогда и только тогда, когда один из операндов имеет значение ИСТИНА, а другой - ЛОЖЬ. Именно это и является уникальным свойством оператора XOR - если вы выполните эту операцию на одним байтом, используя другой байт в качестве «ключа», а затем возьмете результат и выполните над ним ту же самую операцию с помощью того же самого ключа, вы снова получите исходный байт. Например:

Исходный байт		11011001
Ключ	XOR	01010011 (ключ)
Зашифрованный байт		10001010

Зашифрованный байт		10001010
Ключ	XOR	01010011 (ключ)
Расшифрованный байт		11011001

Расшифрованный байт равен исходному.

Этот процесс может использоваться для кодирования файлов, так как он решает две основные проблемы с простейшей версией на базе первого дополнения. Во-первых, благодаря использованию ключа, расшифровать файл, имея только программу декодирования нельзя. Во-вторых, используемые манипуляции с битами не настолько просты, чтобы их можно было сразу распознать.

Ключ не обязательно должен иметь длину 1 байт. Фактически, можно использовать ключ, состоящий из нескольких символов, и чередовать эти символы на протяжении всего файла.

Стандарт ГОСТ.

Официально ГОСТ называется «Алгоритм криптографического преобразования данных ГОСТ 28147-89» - это несколько шире, чем просто зашифровывание или расшифровка данных. Все режимы криптопреобразований данных, согласно ГОСТ, базируются на трех циклах алгоритма.

* цикл зашифровывания (32 - З)
* цикл расшифровки (32 - Р)
* цикл выработки имитоприставки (16 - З)

Прежде чем перейти к изучению основных вопросов, рассмотрим дополнительную информацию, используемую ГОСТом, - именно ее секретность обеспечивает секретность шифрованного сообщения. Эта информация представляет собой 2 массива данных - ключ и таблицу замен. Приведем их характеристики.

1. *Ключ* - это массив из 8-ми 32-битовых элементов, обозначаемых в дальнейшем X_i, где i изменяется от 0 до 7. Таким образом, размер ключа составляет 32×8=256 битов или 32 байта.

2. *Таблица замен* - двумерная таблица - набор из 8-ми одномерных массивов (узлов замен), каждый из которых содержит 16 различных 4-битовых чисел (от 0 до 15) в произвольном порядке. Обозначим $K_m(y)$ значение первого элемента в m-ом узле замен. При этом m изменяется в пределах 0...7, а y - в пределах 0...15. Таким образом, общий объем таблицы замен равен 8 узлов×16 элементов×4 бита/элемент = 512 битов = 64 байта.

Рассмотрим основной шаг криптопреобразования. На входе шага заданы два 32-битовых элемента данных - N_1, N_2, с этими элементами выполняются следующие манипуляции:

1) добавление к N_1 элемента ключа - сложение по модулю 2^{32};
2) поблочная замена результата по 4 бита по таблице замен;
3) циклический сдвиг результата на 11 битов влево;
4) побитовое сложение результата по модулю 2 с элементом N_2;
5) перестановка элементов N_2 <- старое, N_1 <- результат;

После этого новые элементы N_1 и N_2 выдаются в качестве результата шага. Так как в основном шаге используется только один элемент ключа, еще одним параметром шага является номер этого элемента.

Рассмотрим базовые циклы криптоалгоритма ГОСТа. Они отличаются друг от друга только числом повторений основного шага и порядком просмотра элементов ключа. В обозначении цикла nn-X первый элемент (nn) - это число повторений основного шага, а второй кодирует порядок просмотра элементов ключа (буква З - порядок зашифровывания, Р -

расшифровки). Кроме того, в конце циклов шифрования предусмотрена дополнительная перестановка элементов. Приведем порядок использования элементов ключа для трех базовых циклов:

* цикл зашифровывания (32 - З) - 3 раза вперед, 1 раз назад:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0
* цикл расшифровки (32 - Р) - 1 раз вперед, 3 раза назад:
0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0,7,6,5,4,3,2,1,0,7,6,5,4,3,2,1,0
* цикл выработки имитоприставки (16 - З) - 2 раза вперед:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7

Каждый из циклов получает на входе 2 32-битовых слова и после серии основных шагов выдает в качестве результата также 2 32-битовых слова.

Основные режимы шифрования.

ГОСТ 28147-89 предусматривает три режима шифрования данных:
1) простая замена;
2) гаммирование;
3) гаммирование с обратной связью;
и дополнительный режим
4) выработка имитоприставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита - именно поэтому ГОСТ относится к блочным шифрам. Кратко опишем основные режимы шифрования.

Простая замена.

Зашифровывание заключается в применении цикла 32-З к бокам открытого текста, расшифровка - в применении цикла 32-Р к блокам шифротекста. Это наиболее простой режим шифрования, и он имеет следующие недостатки:

* с точки зрения стойкости шифра, одинаковые блоки исходных данных дают одинаковые блоки шифротекста; криптологи говорят, что это очень плохо;
* с точки зрения удобства применения, если длина массива информации не кратна 8 байтам, то возникают 2 проблемы:
* чем и как дополнять последний блок до полных 8 байтов.
* после зашифровывания неполного блока в нем все 8 байт станут значащими, то есть вместе с шифротекстом надо хранить количество байтов в последнем блоке исходного текста.

ГОСТ ограничивает возможные случаи применения простой замены шифрованием ключевой информации (ключи и таблицы замен);

Гаммирование.

Этот режим заключается в наложении на открытые данные гаммы с помощью побитовой функции XOR. Зашифровывание и расшифровка в этом режиме не отличаются друг от друга. Блоки гаммы получаются зашифровыванием в режиме простой замены некоторой последовательности 64-битовых блоков, вырабатываемых датчиком псевдослучайных чисел. От этого датчика не требуется обеспечения никаких статистических характеристик выходной последовательности, а нужен лишь максимально возможный период повторения данных.

Гаммирование с обратной связью.

Данный режим похож на режим гаммирования и отличается от него только тем, что для выработки блока гаммы для шифрования следующего блока данных используется блок шифротекста, полученный на предыдущем шаге. Этим достигается зацепление блоков - каждый блок при шифровании зависит от всех предыдущих.

Выработка имитоприставки к массиву данных.

Имитоприставка - это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Цель использования имитоприставки - обнаружение всех изменений в массиве информации. Для потенциального взломщика две следующие задачи, если он не владеет секретным ключом, практически неразрешимы:

* вычисление имитоприставки для заданного открытого массива информации;

подбор открытых данных под заданную имитоприставку.

Задание на лабораторную работу:

1. Составить программу осуществляющую шифрацию/дешифрацию текста пользователя любым методом битовых манипуляций.