Министерство образования и науки Российской Федерации

Санкт-Петербургский государственный политехнический университет

—

Факультет технической кибернетики

**Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Анализ степенного шифра методом сдвига**

по дисциплине «Теория итерированных шифров»

Выполнил

студент гр. 5088/1 И.И. Сиваков

<*подпись*>

Проверил

ст. преп. О.В. Шемякина

<*подпись*>

Санкт-Петербург

2011

# Задача

Изучить криптоанализ степенного шифра методом сдвига, способы оптимизации метода и способы оценки стойкости шифра.

# Выполненная работа

Размер блока исследуемого шифра 32 бита. Зашифрование выполняется на четырех циклах. На каждом цикле используется один и тот же ключ, выполняются действия:

1. XOR блока текста с блоком ключа.
2. Подстановка S блоков по 4 бита.
3. 32-х битовая перестановка P.

S: (2, 11, 13, 0, 9, 7, 4, 14, 1, 12, 8, 15, 6, 10, 3, 5)

P:

P: (11, 16, 21, 26, 31, 4, 9, 14, 19, 24, 29, 2, 7, 12, 17, 22, 27, 0, 5, 10, 15, 20, 25, 30, 3, 8, 13, 18, 23, 28, 1, 6)

Пусть *F* – один цикл шифрования. Необходимо найти пару текстов таких, чтобы выполнялись условия.

Тогда, выполнив обратную перестановку и обратную подстановку , можно найти ключ.

Основная идея метода анализа, позволяющая сделать его эффективнее, чем перебор заключается в предварительной отбраковке. Чем больше пар можно отбраковать на этом этапе, тем меньше перебор.

Для предварительной отбраковки можно использовать невозможные дифференциалы подстановки. Так как на одном цикле шифрования используется 8 блоков подстановки, и половина дифференциалов невозможны, то вероятность случайному тексту попасть в возможный дифференциал равна . Значит для заданного текста отсеется в среднем кандидатов. Тексты, для которых критерий выполняется можно проверить, попытавшись выполнить первый цикл расшифрования и найти ключ.

Была написана программа, выполняющая проверку критерия для найденной пары и вскрытие ключа.

В результате выполнения программы ключ был найден.

# Выводы

Для использования метода сдвига необходимо чтобы шифр был степенным, то есть используется одинаковая цикловая функция *F* с одинаковым ключом. Метод криптоанализа не зависит от числа циклов шифрования. Необходимым условием практического использования метода является предварительная отбраковка большого числа заведомо ложных пар. Предложенный метод отбраковки не сработает, если в процессе шифрования после выполнения всех циклов будет выполнена еще одна операция XOR с ключом.

Для того чтобы шифр был стойким по отношению к атаке, необходимо чтобы он не был степенным. Например, на каждом цикле можно сдвигать ключ на 1 бит влево. Также после выполнения всех циклов необходимо выполнить XOR с ключом. Это затруднит отбраковку.

# Приложение

//lab5.cpp

#include "stdafx.h"

int main()

{

srand (time(NULL));

//------------------------------------------------

vector<int> my\_s, my\_p;

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

//------------------------------------------------

my\_s = vector<int>(my\_sub, my\_sub+16);

my\_p = vector<int>(32);

for (int i = 0; i < 32; i++) my\_p[i] = (i\*5 + 11)%32;

//------------------------------------------------

cout << "Working.." << endl;

Cipher ciph(my\_s,my\_p);

unsigned char x1[] = "Text", x2[4];

unsigned char y1[4],y2[4];

unsigned char key[] = "keykeykeykeykey";

//"находим" открытый текст, являющийся результатом первого цикла шифрования

memcpy(x2, x1, 4);

ciph.CryptCycleBlock(x2, key);

//зашифруем тексты

memcpy(y1, x1, 4);

memcpy(y2, x2, 4);

for(int i = 0; i < CYCLES; i++)

{

ciph.CryptCycleBlock(y1, key);

ciph.CryptCycleBlock(y2, key);

}

/\*

//если после циклов ксорить с ключом метод отсева не работает

ciph.CryptBlock(y1, key);

ciph.CryptBlock(y2, key);

/\*\*/

//проверяем критерий отбраковки

if(ciph.CheckCriterion(x1, x2, y1, y2))

{

cout << "Pair sieved" << endl;

\_getch();

return 0;

}

//вскрываем ключ

unsigned char findkey[] = "\x00\x00\x00\x00\x00";

ciph.DecryptCycleBlock(x2, findkey);

for(int i = 0; i < 4; i++)

{

findkey[i] = x1[i]^x2[i];

}

cout << "Key finded: " << findkey << endl;

\_getch();

return 0;

}