# 3. Шифр Виженера

3.1 алгоритм шифрования

Алгоритм шифрования осуществляет работу над комутативным кольцом вычетов по модулю n, где n- мощность алфавита используемого языка.

Введем обозначения:

Отождествим буквы алфавита с числами:

Алгоритм шифрования в веденных обозначениях выглядит следующим образом:

Алгоритм расшифрования можно путем выражения из уравнения шифрования символа открытого текста:

Расмотрим програмную реализацию шифрования и расшифрования.

Данные алгоритмы реализует функция:

void crypt(FILE \*source, char\* mode, char\* key);

FILE \*source - дескриптор файла, подлежащий шифровке или расшифровке;

char\* mode - модификатор работы функции;

char\* key - указатель на область памяти в которой хранится ключ;

//я в ожидание окончания работы

3.2 Криптоанализ шифра Вижинера

Криптоанализ шифра Виженера проходит в два этапа:

1. Поиск длины ключа
2. Поиск по длине ключа самого ключевого слова

Для определения длины ключа воспользуемся индексом совпадения.

**Индексом совпадения** называют вероятность того, что две выбранные случайно буквы текста совпадут.

Тогда вероятность совпадения двух случайно выбранных букв с будет равна .

Тогда индекс совпадений примет вид:

В случае, когда текст известен можно посчитать численное выражение для введенных величин.

Пусть символ алфавита встречается в тексте раз, при этом обозначим общее количество элементов текста за N, в таком случае:

Для русского и английских языков данный индекс посчитан и составляет для английского языка 0,0662, а для русского 0,529.

Вернемся к определению длины ключа. Посмотрим, как шифруется текст. Для это разобьем открытый текст M по следующему правилу:

В таком случае рассматривая преобразования символов в определенном получим, что для всех символов алфавита определено некоторое смещение равное постоянное для всех элементов перестановки . В таком случае можем сказать, что для шифрования символов лежащих в реализуется подстановка:

где – перестановка шифрованных символов полученных из .

При этом индекс совпадений для не изменится, так как в общей сумме останутся те же слагаемые, но в этом случае они будут относится к другим буквам определенным подстановкой , а в силу биективности каждое слагаемое будет присутствовать и в новой сумме.

В таком случае алгоритм поиска длины ключа будет сводится к разбиению зашифрованного текста на соответствующие перестановки и подсчитывания для них индекса совпадения. Если индекс совпадения равен с определенной точностью заранее определенным индексам для используемого языка, то используемую в данной итерации длину ключа можно считать предположительно верной. После достаточного анализа закрытого текста из всех выбранных длин выберем ту, у которой самый большой индекс совпадений и при этом она делит большинство выбранных значений.

Данный алгоритм реализуется функцией:

int find\_Key\_lenth(FILE\* file, char mod, char\*\* kod);

FILE \*file - дескриптор файла, подлежащий шифровке или расшифровке;

char\* mod - модификатор работы функции;

char\*\* kod - указатель на область памяти в которой будет хранится зашифрованный текст для последующего анализа;

Рассмотрим алгоритм, по которому работает данная функция.

Узнаем размер файла, с которым нам предстоит работать

fseek(file, 0, SEEK\_END);

int lenth\_kod = ftell(file);

fseek(file, 0, SEEK\_SET);

Введем этот файл в заранее заданную область в виде массива

\*kod = (char\*)calloc((lenth\_kod + 2) \* sizeof(char), (lenth\_kod + 2) \* sizeof(char));

fread(\*kod, sizeof(char), lenth\_kod, file);

Так как функция работает по-разному в зависимости от используемого языка, то для этих целей был введен в определение принимаемых значений модификатор работы функции, который может принимать значения `e` и `r` обозначающие английский и русский языки соответственно.

Определим мощность языка lenth\_alf и позицию первого символа в таблице ASCI otclonenie\_table.

if (mod == 'r') {//русский

otclonenie\_table = 192;

lenth\_alf = 33;

}

else if (mod == 'e') {//англ

otclonenie\_table = 65;

lenth\_alf = 26;

}

else

return 0;//в случае если модификатор введен неверно

Уберем из заданного массива знаки, не участвующие в процессе шифрования

for (int i = 0; i + otclonenie < lenth\_kod;)

{

if (!((unsigned char)(\*kod)[i + otclonenie] >= otclonenie\_table && (unsigned char)(\*kod)[i + otclonenie] <= otclonenie\_table + lenth\_alf || (unsigned char)(\*kod)[i + otclonenie] == 168))

{

otclonenie++;

}

else

{

(\*kod)[i] = (\*kod)[i + otclonenie];

i++;

}

}

После чего добавим в конец файла нуль символ и переопределим размер массива

(\*kod)[lenth\_kod - otclonenie] = 0;

lenth\_kod = lenth\_kod - otclonenie;

Также определим массив

int\* kol\_vo\_simbols = 0;

kol\_vo\_simbols = (int\*)malloc(lenth\_alf \* sizeof(int));

В таком случае заданные величины в программе и в теоретической части проекта будут соответствовать:

По алгоритму, описанному выше найдем длину ключа

for (int i = 2; i < lenth\_kod; i++)

{

По переменной i перейдем к перебору всевозможных длин ключа, подразумевая, что она меньше заявленного текста

for (int f = 0; f < i && f < 100; f++)

{

По переменой f будем перемещается между различными определенными выше

Далее обнулим массив kol\_vo\_simbols содержащий количество элементов для , определенного первыми двумя циклами

for (int q = 0; q < lenth\_alf; q++)

{

kol\_vo\_simbols[q] = 0;

}

index\_covpadeniu = 0;

Заполним массив kol\_vo\_simbols

for (int j = f; j < lenth\_kod; j = j + i)

{

if ((unsigned char)(\*kod)[j] >= otclonenie\_table && (unsigned char)(\*kod)[j] < otclonenie\_table + lenth\_alf)

kol\_vo\_simbols[(unsigned char)(\*kod)[j] - otclonenie\_table]++;

else if ((unsigned char)(\*kod)[j] == 168)

kol\_vo\_simbols[lenth\_alf - 1]++;

}

Подсчитаем индекс совпадения

for (int j = 0; j < lenth\_alf; j++)

{

index\_covpadeniu = index\_covpadeniu + (kol\_vo\_simbols[j] \* (kol\_vo\_simbols[j] - 1)) / ((float)((lenth\_kod) / i) \*((lenth\_kod) / i - 1));

}

Проверим выполнение условий, накладываемых на индекс совпадения

if ((index\_covpadeniu < right\_wall\_fo\_E && index\_covpadeniu > left\_wall\_fo\_E && mod == 'e') || ((index\_covpadeniu < right\_wall\_fo\_R && index\_covpadeniu > left\_wall\_fo\_R) && mod == 'r'))

{

Выберем длину ключа smeshenie по условиям, указанным в описании алгоритма

if (index\_covpadeniu > MAX\_index\_covpadeniu && (i % smeshenie) != 0 || proverka)

{

smeshenie = i;

if (proverka)proverka--;

MAX\_index\_covpadeniu = index\_covpadeniu;

}

if ((i % smeshenie) == 0)MAX\_index\_covpadeniu \*= 1.06;

}

}

}

// я не знаю, что писать

Следующий этап в криптоанализе шифра Виженера, это определение самого ключа. Для этого воспользуемся индексом взаимного совпадения.

**Взаимный индекс совпадения** называют вероятность того, что две выбранные случайно буквы из двух текстов совпадут.

вероятность совпадения повторно выбранной из другого текста буквы с

Пусть символ алфавита встречается в первом тексте раз, при этом обозначим общее количество элементов первого текста за N, а для второго текста возьмём аналогичные обозначения :

В таком случае взаимный индекс совпадения примет вид:

Разобьем С на по ранее указанному правилу, в таком случае текст разобьется на следующие перестановки:

Введем для каждого такой перестановки перестановку по следующему правилу:

- количество символов в перестановке .

Вернемся к взаимному индексу совпадения. Для осмысленных текстов написанных на английском и русском языках индекс посчитан и составляет 0,066 , 00000 соответственно. В случае если один из текстов представляет набор случайных (подразумевается неосмысленный набор букв) символов, то индекс будет принимать значения близкие к числу 0,04 или меньше. Посмотрим выражение взаимного индекса для нашего разбиения:

Рассмотрим взаимный индекс совпадения для

При этом вероятность совпадения i-ой буквы алфавита в соответствующих открытых текстах для равна , в таком случае взаимный индекс совпадения примет вид:

Произведем замену индекса суммирования и так как индекс берется как остаток от деления, в таком случае получим

В таком случае получим окончательную формулу для вычисления взаимного индекса совпадения

В таком случае, поиск ключевого слова сводится к перебору всевозможных d для двух таких перестановок и поиска среди них удовлетворяющей условиям:

1. приблизительно совпадает со значением взаимного индекса совпадения для осмысленных текстов написанных на данном языке;

Для реализации алгоритма поиска ключевого слова зафиксируем значение первого символа ключа, а для остальных символов найдем их сдвиг относительно первого. В таком случае пользователю остается из переложенных n фраз выбрать самую подходящую.

Данный алгоритм реализуется функцией:

void attack(FILE\* f);

FILE \*file - дескриптор файла, подлежащий шифровке или расшифровке;