Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №4

«Исследование криптографических шифров на основе подстановки (замены) символов»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

Минск 2020

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации подстановочных шифров.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости подстановочных шифров.

2. Ознакомиться с особенностями реализации и свойствами различных подстановочных шифров на основе готового программного средства (L\_LUX).

3. Разработать приложение для реализации указанных преподавателем методов подстановочного зашифрования/расшифрования.

4. Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.

5. Оценить скорость зашифрования/расшифрования реализованных способов шифров.

6. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Сущность подстановочного шифрования состоит в том, что, исходный текст (из множества М) и зашифрованный текст (из множества С) основаны на использовании одного и того же или разных алфавитов, а тайной или ключевой информацией является алгоритм подстановки. Если исходить из того, что используемые алфавиты являются конечными множествами, то в общем случае каждой букве ax алфавита AM (axAM) для создания сообщения Мi (Мi M) соответствует буква ay или множество букв{АхС} для создания шифртекста Сi. Важно, чтобы во втором случае любые два множества (например, {АхС}b и {АхС}n, b≠n, 1 ≤ b, n, x, y ≤ N, N – мощность алфавита), используемые для замены разных букв открытого текста, не пересекались: {АхС}b ∩ {АхС}n = 0.

Если в сообщении Мi содержится несколько букв ax, то каждая из них заменяется на символ ay либо на любой из символов {АхС}. За счет этого с помощью одного ключа можно сгенерировать различные Сi для одного и того же Мi. Так как множества {АхС}b и {АхС}n попарно не пересекаются, то по каждому символу Сi можно однозначно определить, какому множеству он принадлежит, и, следовательно, какую букву открытого сообщения Мi он заменяет. В силу этого открытое сообщение восстанавливается из зашифрованного однозначно.

Приведенные утверждения справедливы для следующих типов подстановочных шифров:

* моноалфавитных (шифры однозначной замены или простые подстановочные),
* полиграммных,
* омофонических (однозвучные шифры или шифры многозначной замены),
* полиалфавитных.

В моноалфавитных шифрах операция замена производится только над каждым одиночным символом сообщения Мi. Для наглядной демонстрации шифра простой замены достаточно выписать под заданным алфавитом тот же алфавит, но в другом порядке или, например, со смещением. Записанный таким образом алфавит называют алфавитом замены. Максимальное количество ключей для любого шифра этого вида не превышает N!, где N – количество символов в алфавите. Для математического описания криптографического преобразования предполагаем, что зашифрованная буква ay, соответствующая символу aх, находится на позиции y = x + k (mod N), x, y – индекс (порядковый номер, начиная с 0) символа в используемом алфавите, k – ключ.

Для расшифрования сообщения Сi необходимо произвести расчеты обратные (2.1), т. е.: х = у – k (mod N).

Система шифрования Цезаря с ключевым словом (лозунгом) также является одноалфавитной системой подстановки. Особенностью этой системы является использование ключевого слова (лозунга) для смещения и изменения порядка символов в алфавите подстановки (желательно, чтобы все буквы ключевого слова были различными). Ключевое слово пишется в начале алфавита подстановки.

Метод можно видоизменить, если ключевое слово записывать начиная не с первого символа (нулевой индекс) во второй строке, а в соответствии с некоторым числом а: 0 ≤ a < N.

Применяя одновременно операции сложения и умножения по модулю n над элементами множества (индексами букв алфавита), можно получить систему подстановок, которую называют аффинной системой подстановок Цезаря. Определим процедуру зашифрования в такой системе: y = ax + b (mod N), где a и b – целые числа. При этом взаимно однозначные соответствия между открытым текстом и шифртекстом будут иметь место только при выполнении следующих условий: 0 ≤ a, b< N, наибольший общий делитель (НОД) чисел a, b равен 1, т.е. эти числа являются взаимно простыми.

В полиграммных шифрах одна подстановка соответствует сразу нескольким символам исходного текста. Первым известным шифром этого типа является шифр Порты. Шифр представляется в виде таблицы. Наверху горизонтально и слева вертикально записывался стандартный алфавит. В ячейках таблицы записываются числа в определенном порядке.

Шифрование выполняется парами букв исходного сообщения. Первая буква пары указывает на строку, вторая – на столбец. В случае нечетного количества букв в сообщении Мi к нему добавляется вспомогательный символ, например, «А». Другими известными полиграммными шифрами являются шифр Плейфера и шифр Хилла. С точки зрения криптостойкости рассматриваемый тип шифров имеет преимущества перед моноалфавитными шифрами. Это связано с тем, что распределение частот групп букв значительно более равномерное, чем отдельных символов. Во-вторых, для эффективного частотного анализа требуется больший размер зашифрованного текста, так как число различных групп букв значительно больше, чем мощность алфавита.

Омофонические шифры (омофоническая замена) или однозвучные шифры подстановки создавались с целью увеличить сложность частотного анализа шифртекстов путем маскировки реальных частот появления символов текста с помощью омофонии. В 1401 г. Симеоне де Крема стал использовать таблицы омофонов для сокрытия частоты появления гласных букв в тексте при помощи более чем одной подстановки. Такие шифры позже стали называться шифрами многозначной замены или омофонами (омофоны – от греч. homos – одинаковый и phone – звук) – слова, которые звучат одинаково, но пишутся по-разному и имеют разное значение; очень много подобных слов содержит английский язык). Они получили развитие в XV веке. В книге «Трактат о шифрах» Леона Баттисты Альберти (итальянский ученый, архитектор, теоретик искусства, секретарь папы Климентия XII), опубликованной в 1466 г, приводится описание шифра замены, в котором каждой букве ставится в соответствие несколько эквивалентов, число которых пропорционально частоте встречаемости буквы в открытом тексте, Мi. В этих шифрах буквы исходного алфавита соответствуют более чем одному символу из алфавита замены. Обычно символам исходного текста с наивысшей частотой дают большее количество эквивалентов, чем более редким символам. Таким образом, распределение частоты становится более равномерным, сильно затрудняя частотный анализ.

Книжный шифр. Заметным вкладом греческого ученого Энея Тактика в криптографию является предложенный им так называемый книжный шифр. После Первой мировой войны книжный шифр приобрел иной вид. Шифрозамена для каждой буквы определялась набором цифр, которые указывали на номер страницы, строки и позиции в строке (вспомните пример использования такого шифра известными героями фильма «17 мгновений весны»). Даже с формальной стороны отсутствие полной электронной базы изданных к настоящему времени книг делает процедуру взлома шифра практически не выполнимой.

Полиалфавитные (или многоалфавитные) шифры состоят из нескольких шифров однозначной замены. Выбор варианта алфавита для зашифрования одного символа зависит от особенностей метода шифрования. Диск Альберти. В «Трактате о шифрах» Альберти приводит первое точное описание многоалфавитного шифра на основе шифровального диска. Он состоял из двух дисков – внешнего неподвижного и внутреннего подвижного дисков, на которые были нанесены буквы алфавита. Процесс шифрования заключался в нахождении буквы открытого текста на внешнем диске и замене ее на букву с внутреннего диска, стоящую под ней. После этого внутренний диск сдвигался на одну позицию и шифрование второй буквы производилось уже по-новому шифралфавиту. Ключом данного шифра являлся порядок расположения букв на дисках и начальное положение внутреннего диска относительно внешнего.

В указанной книге Трисемус впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра подстановки обычно использовались таблица для записи букв алфавита и ключевое слово (или фраза). Можно найти определенную аналогию с системой шифрования Цезаря с ключевым словом. В таблицу сначала вписывалось по стрелкам ключевое слово, причем повторяющиеся буквы также отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку.

Таким образом, ключом в таблицах Трисемуса является ключевое слово и размер таблицы. При шифровании буква открытого текста заменяется буквой, расположенной ниже нее в том же столбце. Если буква текста оказывается в нижней строке таблицы, тогда для шифртекста берут самую верхнюю букву из того же столбца.

**Ход работы**

В соответствии с заданием было необходимо реализовать алгоритмы Цезаря с ключевым словом и таблицы Трисемуса для шифрования и дешифрования текста на белорусском языке.

Алгоритм шифра Цезаря приведен в листинге 1.

class Caesar

{

private static string alpha = "абвгдеёжзiйклмнопрстуўфхцчшыьэюя";

private static char[] newAlpha = new char[32];

public static string Encrypt(string Message)

{

var map = new Dictionary<char, int>();

foreach (char c in Message)

{

if (!map.ContainsKey(c))

map.Add(c, 1);

else

map[c] += 1;

}

int len = Message.Length;

var orderkey = from i in map orderby i.Key select i;

foreach (var item in orderkey)

{

using (StreamWriter sw = new StreamWriter("info1.log", true, Encoding.Default))

{

sw.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)item.Value/len}\n");

}

}

string res = "";

foreach (char ch in Message)

{

for (int i = 0; i < alpha.Length; i++)

{

if (ch != alpha[i])

continue;

res += newAlpha[i];

break;

}

}

return res;

}

public static string Decrypt(string Message)

{

var map = new Dictionary<char, int>();

foreach (char c in Message)

{

if (!map.ContainsKey(c))

map.Add(c, 1);

else

map[c] += 1;

}

int len = Message.Length;

var orderkey = from i in map orderby i.Key select i;

foreach (var item in orderkey)

{

using (StreamWriter sw = new StreamWriter("info2.log", true, Encoding.Default))

{

sw.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)(item.Value/len)}\n");

}

}

string res = "";

foreach (char ch in Message)

{

for (int i = 0; i < newAlpha.Length; i++)

{

if (ch != newAlpha[i])

continue;

res += alpha[i];

break;

}

}

return res;

}

public static void CreateNewAlpha(string keyWord, int key)

{

bool findSame = false;

key++;

int beg = 0, current = key;

for (int i = key; i < keyWord.Length + key; i++)

{

for (int j = key; j < keyWord.Length + key; j++)

{

if (keyWord[i - key] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = keyWord[i - key];

current++;

}

findSame = false;

}

for (int i = 0; i < alpha.Length; i++)

{

for (int j = 0; j < newAlpha.Length; j++)

{

if (alpha[i] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = alpha[i];

current++;

}

findSame = false;

if (current == newAlpha.Length)

{

beg = i;

break;

}

}

current = 0;

for (int i = beg; i < alpha.Length; i++)

{

for (int j = 0; j < newAlpha.Length; j++)

{

if (alpha[i] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = alpha[i];

current++;

}

findSame = false;

}

}

public static string GetNewAlpha()

{

string strNewAlpha = new string(newAlpha);

return strNewAlpha;

}

}

Листинг1 – Реализация шифра цезаря с ключевым словом для белорусского языка

Программная реализация шифрования с использованием таблицы трисемуса приведена в листинге 2.

class Trisemus

{

private static string alphabet = "абвгдеёжзiйклмнопрстуўфхцчшыьэюя";

private static char[,] table = new char[4,8];

public static void GetNewTable (string keyWord)

{

bool findSame = false;

char[] newAlpha = new char[32];

int key = -1;

key++;

int beg = 0, current = key;

for (int i = key; i < keyWord.Length + key; i++)

{

for (int j = key; j < keyWord.Length + key; j++)

{

if (keyWord[i - key] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = keyWord[i - key];

current++;

}

findSame = false;

}

for (int i = 0; i < alphabet.Length; i++)

{

for (int j = 0; j < newAlpha.Length; j++)

{

if (alphabet[i] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = alphabet[i];

current++;

}

findSame = false;

if (current == newAlpha.Length)

{

beg = i;

break;

}

}

current = 0;

for (int i = beg; i < alphabet.Length; i++)

{

for (int j = 0; j < newAlpha.Length; j++)

{

if (alphabet[i] == newAlpha[j])

{

findSame = true;

break;

}

}

if (!findSame)

{

newAlpha[current] = alphabet[i];

current++;

}

findSame = false;

}

int index = 0;

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

{

for (int j = 0; j < 8; j++)

{

table[i, j] = newAlpha[index++];

}

}

}

internal static string Encrypt(string message)

{

var map = new Dictionary<char, int>();

foreach (char c in message)

{

if (!map.ContainsKey(c))

map.Add(c, 1);

else

map[c] += 1;

}

int len = message.Length;

var orderkey = from i in map orderby i.Key select i;

foreach (var item in orderkey)

{

using (StreamWriter sw = new StreamWriter("info3.log", true, Encoding.Default))

{

sw.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)item.Value / len}\n");

}

}

string res = "";

foreach (char ch in message)

{

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

{

for (int j = 0; j < 8; j++)

{

if (ch != table[i, j])

continue;

if (i == 3)

{

res += table[0, j];

break;

}

res += table[i+1, j];

break;

}

}

}

return res;

}

internal static string Decrypt(string message)

{

var map = new Dictionary<char, int>();

foreach (char c in message)

{

if (!map.ContainsKey(c))

map.Add(c, 1);

else

map[c] += 1;

}

int len = message.Length;

var orderkey = from i in map orderby i.Key select i;

foreach (var item in orderkey)

{

using (StreamWriter sw = new StreamWriter("info4.log", true, Encoding.Default))

{

sw.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)item.Value / len}\n");

}

}

string res = "";

foreach (char ch in message)

{

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

{

for (int j = 0; j < 8; j++)

{

if (ch != table[i, j])

continue;

if (i == 0)

{

res += table[3, j];

break;

}

res += table[i-1, j];

break;

}

}

}

return res;

}

}

Листинг 2 – реализация шифрования с использованием таблицы Трисемуса

Кроме того, было необходимо измерить время выполнения шифрования и расшифрования. Были получены следующие результаты:

* Шифрование с использованием шифра Цезаря: 585мс;
* Дешифрование с использованием шифра Цезаря: 523мс;
* Шифрование с использованием таблицы Трисемуса 547мс;
* Дешифрование с использованием таблицы Трисемуса 475мс.