Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 1:

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

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

Минск 2023

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

**Задачи:**

* Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости подстановочных шифров.
* Ознакомиться с особенностями реализации и свойствами различных подстановочных шифров на основе готового программного средства (L\_LUX).
* Разработать приложение для реализации указанных преподавателем методов подстановочного зашифрования/расшифрования.
* Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.
* Оценить скорость зашифрования/расшифрования реализованных способов шифров.
* Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения эксперимента с использованием приложения и результатов эксперимента.

1 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 (Сi ∈ С). 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). В данных шифрах операция замены производится раздельно над каждым одиночным символом сообщения Мi.
* Полиграммных. В таких шифрах одна подстановка соответствует сразу нескольким символам исходного текста.
* Омофонических (однозвучные шифры или шифры многозначной замены). Создавались с целью увеличить сложность частотного анализа шифртекстов путем маскировки реальных частот появления символов текста с помощью омофонии.
* Полиалфавитных. Состоят из нескольких шифров однозначной замены. Выбор варианта алфавита для зашифрования одного символа зависит от особенностей метода шифрования.

2 Практическая часть

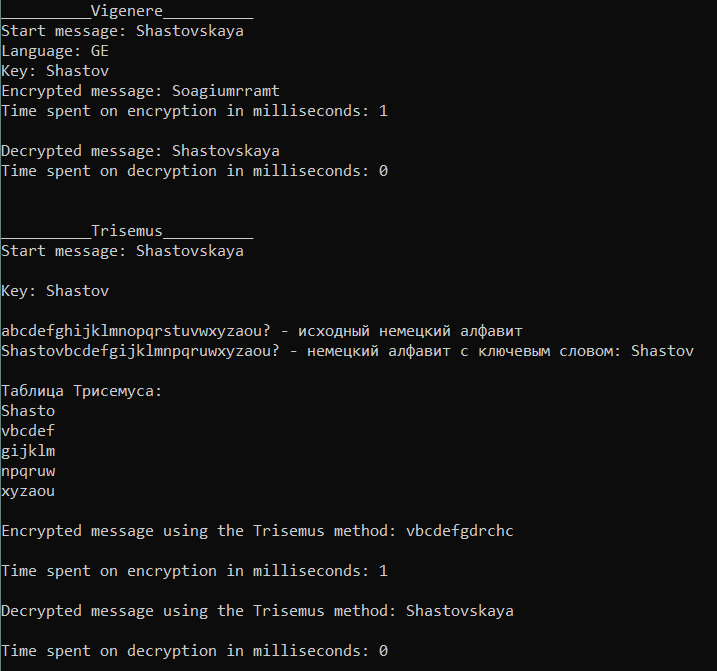


Рисунок 2.1 – Результат шифрования/дешифрования исходного сообщения методами Виженера и Трисемуса с ключевым словом

На данном скриншоте отображен процесс работы программы с шифрованием и дешифрованием исходного сообщения методами Виженера и Трисемуса с ключевым словом. Так же выведены вспомогательные средства: алфавиты, таблицы и прочее.

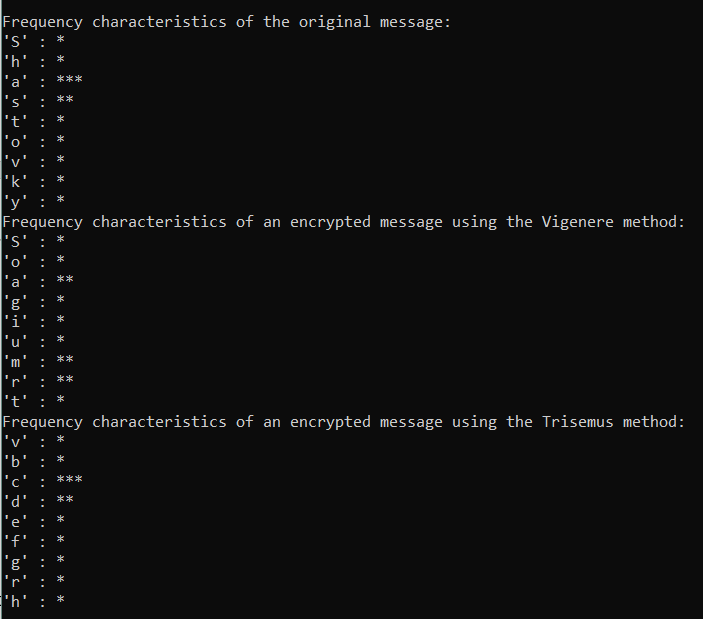


Рис. 2.2 – Частоты появления символов в исходном и зашифрованных сообщениях

На данном скриншоте отображены гистограммы частот появления символов для исходного и зашифрованного сообщений.

Основные функции программы представлены на рисунке 2.3.

public static string TrisemusEncryptionOfGerman(string key, string message)

{

string alphabet = "abcdefghijklmnopqrstuvwxyzäöüß";

string alphabetWithKey = "";

string temp = alphabet;

int rows = 5;

string[] tableOfTrisemus = new string[rows];

string encryptedMessage = "";

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

{

if (alphabetWithKey.Length > 0)

{

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

{

if (alphabetWithKey[j] != key[i])

{

alphabetWithKey += key[i];

break;

}

}

}

else

{

alphabetWithKey += key[i];

}

}

int counter = 0;

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

{

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

{

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

{

temp = temp.Remove(i - counter, 1);

counter++;

break;

}

}

}

//Console.WriteLine(temp);

alphabetWithKey += temp;

Console.WriteLine(alphabet + " - исходный немецкий алфавит");

Console.WriteLine(alphabetWithKey + " - немецкий алфавит с ключевым словом: " + key + "\n");

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

{

int count = 0;

for (int j = i \* 6; j < alphabetWithKey.Length; j++)

{

if (count < 6)

{

tableOfTrisemus[i] += alphabetWithKey[j];

count++;

}

else

{

break;

}

}

}

Console.WriteLine("Таблица Трисемуса:");

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

{

Console.WriteLine(tableOfTrisemus[i]);

}

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

{

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

{

for (int k = 0; k < tableOfTrisemus[j].Length; k++)

{

if (message[i] == tableOfTrisemus[j][k])

{

if (j == 4)

{

encryptedMessage += tableOfTrisemus[0][k];

}

else

{

encryptedMessage += tableOfTrisemus[j + 1][k];

}

}

}

}

}

return encryptedMessage;

}

public static string TrisemusEncryptionOfGerman(string key, string message)

{

string alphabet = "abcdefghijklmnopqrstuvwxyzäöüß";

string alphabetWithKey = "";

string temp = alphabet;

int rows = 5;

string[] tableOfTrisemus = new string[rows];

string encryptedMessage = "";

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

{

if (alphabetWithKey.Length > 0)

{

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

{

if (alphabetWithKey[j] != key[i])

{

alphabetWithKey += key[i];

break;

}

}

}

else

{

alphabetWithKey += key[i];

}

}

int counter = 0;

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

{

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

{

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

{

temp = temp.Remove(i - counter, 1);

counter++;

break;

}

}

}

//Console.WriteLine(temp);

alphabetWithKey += temp;

Console.WriteLine(alphabet + " - исходный немецкий алфавит");

Console.WriteLine(alphabetWithKey + " - немецкий алфавит с ключевым словом: " + key + "\n");

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

{

int count = 0;

for (int j = i \* 6; j < alphabetWithKey.Length; j++)

{

if (count < 6)

{

tableOfTrisemus[i] += alphabetWithKey[j];

count++;

}

else

{

break;

}

}

}

Console.WriteLine("Таблица Трисемуса:");

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

{

Console.WriteLine(tableOfTrisemus[i]);

}

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

{

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

{

for (int k = 0; k < tableOfTrisemus[j].Length; k++)

{

if (message[i] == tableOfTrisemus[j][k])

{

if (j == 4)

{

encryptedMessage += tableOfTrisemus[0][k];

}

else

{

encryptedMessage += tableOfTrisemus[j + 1][k];

}

}

}

}

}

return encryptedMessage;

}

Рисунок 2.3 – Основные функции программы

Вывод

В ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации подстановочных шифров.