УДК 004.9

ШИФРОВАНИЕ ИНФОРМАЦИИ СИММЕТРИЧНЫМ СПОСОБОМ

Аратин Д.В., студент гр. ПИб-192, 2 курс
Гилева К.В., студент гр. ПИб-192, 2 курс
Научный руководитель: Киреева К.А., ассистент
Кузбасский государственный технический университет
имени Т.Ф. Горбачева,
г. Кемерово

Аннотация: шифрование — это способ, применяемый для преобразования сведений в зашифрованную форму для того, чтобы она была доступна только определенному кругу лиц. Шифрование используется для защиты данных от других людей при хранении данных или передачи данных по каналам связи. С учетом использования ключей шифрования, среди методов шифрования выделяют симметричное шифрование и асимметричное шифрование. При симметричном шифровании необходим только один ключ для шифровки и расшифровки (идентичный для отправителя и получателя), в свою очередь данный ключ должен остаться в секрете. Также симметричное шифрование делится на разновидности шифров. Блочный алгоритм DES базируется на основе сети Фейстеля. Главной характерной чертой данного алгоритма является то, что дешифрирование и шифрование имеют одинаковые шаги для шифрования и расшифровки: части L и R меняются местами, а затем выполняется операция сложения L и F (R, Ki). Данный процесс, описанный выше, дает возможность сделать операции шифрования и расшифровки доступными для понимания.

Ключевые слова шифрование, криптография, ключ, симметричное шифрование, алгоритм DES.

В современном мире безопасность передачи информации играет важную роль. Звонок по телефону, отправка сообщений, оплата онлайн с использованием банковских карт, во всех этих действиях происходит передача информации между устройствами, но что если к этим данным может получить доступ кто-то еще? Для этого применяются различные типы алгоритмов шифрования информации. Для любого алгоритма шифрования характерной чертой является использование ключа шифрования. Он утверждает решение конкретного варианта преобразования из совокупности возможных версий для данного алгоритма. Одним из таких типов алгоритмов является симметричный тип. Алгоритмы данного типа имеют особый подход: для шифрования и расшифровки сообщения используется один и тот же ключ, которым устройства обмениваются между собой и тем самым позволяют получателю расшифровать сообщение.

Проанализируем алгоритм симметричного шифрования DES. Данный алгоритм блочного шифрования основан на базе сети Фейстеля [1], через которую пропускаются блоки исходной информации 16 раз (раундов). При шифровании исходная информация переводится в двоичный код, после чего разбивается на блоки и каждый блок отдельно зашифровывается или расшифровывается. Размер блока данного алгоритма равен 128 бита, т.к. применяется 16-ти битная кодировка Юникода (UTF-16) и получаем, что в одном блоке – 8 символов [2].

Для того чтобы зашифровать сообщения данным алгоритмом, необходимо выполнить ряд действий:

- довести исходную информацию до необходимого размера (в битах), чтобы она делилась без остатка на размер блока (128 бит);
- разделить исходную информацию на блоки;
- довести ключ до необходимой длины, а именно до половины длины блока.
- преобразовать ключ в двоичный формат, то есть в нули и единицы;
- над каждым блоком выполнять прямое преобразование сетью Фейстеля в течение 16-ти раундов. А после каждого раунда следует осуществлять циклический сдвиг ключа на установленное число битов (Рисунок 1);
- соединить все блоки после преобразований и таким образом, получаем сообщение, зашифрованное алгоритмом симметричного шифрования DES.

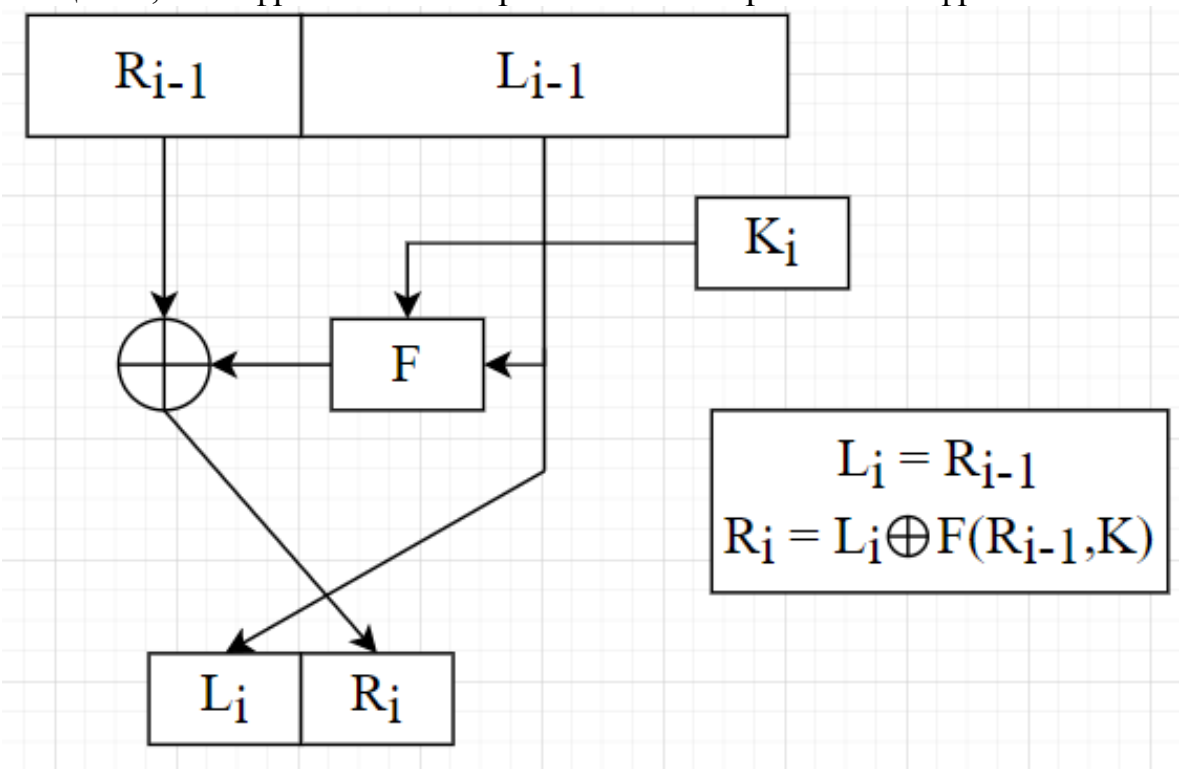

Рисунок 1 - Прямое преобразование сетью Фейстеля

Для расшифровки сообщения используется обратное преобразование сетью Фейстеля (Рисунок 2).

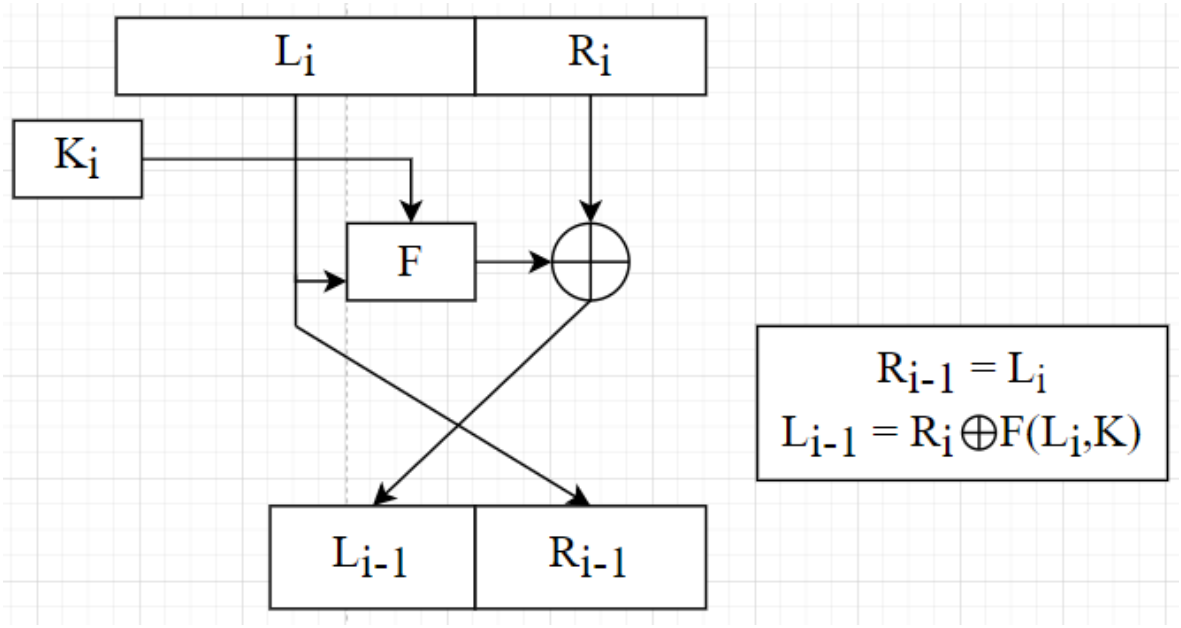

Рисунок 2- Обратное преобразование сетью Фейстеля

Рассмотрим один раунд прямого преобразования.

На i-й итерации первый блок исходного сообщения разделяется пополам – правая часть обозначается R, соответственно левая L. Над R и ключом K вычисляется логическая функция F (будет применяться XOR). Производится логическая операция «исключающее ИЛИ» над L и вычисленным ранее значением функции (L XOR F). Старое значение R переносится в левую часть блока, а в правую часть записывается значение операции L XOR F. И последняя операция раунда – осуществить циклический сдвиг ключа на фиксированное кол-во бит: iShiftKey. При шифровке: ключ сдвигается направо, при расшифровке: ключ сдвигается налево. «iShiftKey» – количество битов, на которое необходимо циклически сдвигать ключ.

Программная реализация выполнена на языке C# с использованием Visual Studio.

Объявим константы для данного алгоритма (Рисунок 3).

```
private const int iSizeBlock = 128; // размер блока DES равен 128 бита, то
                                    // есть используется 16-ти битная кодировка
Юникода (UTF-16)
        private const int iSizeChar = 16; //размер одного символа (in Unicode 16 bit)
        private const int iShiftKey = 2; //сдвиг ключа
        private const int iCountRounds = 16; //количество раундов
        string[] Blocks; //блоки в двоичном формате
```

Рисунок 3 - Объявление постоянных значений переменных

На вход поступает исходное сообщение и ключ для шифрования. Исходное сообщение должно занимать полностью блок (iSizeBlock), для этого в конец сообщения добавляются символы “|”. Ключ доводится до фиксированной длинны, а именно 4 символа, если символов меньше, то в конец ключа также добавляются символы “0” (Рисунок 44).

```
//доводим длину ключа до нужной
1 reference
private string CorrectKeyWord(string InputEncryptKey, int LengthKey)
{
    if (InputEncryptKey.Length > LengthKey)
        InputEncryptKey = InputEncryptKey.Substring(0, LengthKey);
    else
        while (InputEncryptKey.Length < LengthKey)
            InputEncryptKey = "0" + InputEncryptKey;
    return InputEncryptKey;
}
```

Рисунок 4 - Метод, доводящий ключ до необходимой длины

В дальнейшем сообщение делится на блоки, фиксированной размерности (Рисунок 5).

```
//деление строки на блоки
1 reference
private void CutStringIntoBlocks(string InputEncryptText)
{
    Blocks = new string[(InputEncryptText.Length * iSizeChar) / iSizeBlock];
    int lengthOfBlock = InputEncryptText.Length / Blocks.Length;
    for (int i = 0; i < Blocks.Length; i++)
    {
        Blocks[i] = InputEncryptText.Substring(i * lengthOfBlock, lengthOfBlock);
        Blocks[i] = StringToBinaryFormat(Blocks[i]);
    }
}
```

Рисунок 5 - Метод, делящий исходную информацию на блоки

Информация, находящаяся в этих блоках, переводится из символьного вида в бинарный вид (0 и 1) (Рисунок 66).

```
//перевод строки в двоичный формат
4 references
private string StringToBinaryFormat(string InputEncryptText)
{
    string output = "";
    for (int i = 0; i < InputEncryptText.Length; i++)
    {
        string char_binary = Convert.ToString(InputEncryptText[i], 2);
        while (char_binary.Length < iSizeChar)
            char_binary = "0" + char_binary;
        output = output + char_binary;
    }
    return output;
}
```

Рисунок 6 - Метод, переводящий строки из символьного вида в бинарный вид

Подготовленные блоки прогоняются через алгоритм шифрования. Каждый блок, содержащий в себе набор битов информации, проходит 16 раундов (Рисунок 4), один раунд шифрования представляет собой: за L принимается левая часть блока, за R правая. Над R и ключом выполняется шифрующая функция (Рисунок 5), в нашем случае это была операция “исключающего или”, после этого над результатом операции и L также выполняется операция “исключающего или” (Рисунок 6).

```
//шифрование DES один раунд
1 reference
private string EncodeDES_One_Round(string InputEncryptText, string KeyEncode)
{
    string L = InputEncryptText.Substring(0, InputEncryptText.Length / 2);
    string R = InputEncryptText.Substring(InputEncryptText.Length / 2,
    InputEncryptText.Length / 2);
    return (R + XOR(L, EncryptFunction(R, KeyEncode)));
}
```

Рисунок 4 - Метод, выполняющий один раунд шифрования информации алгоритма DES

```
//шифрующая функция EncryptFunction. в данном случае это XOR
2 references
private string EncryptFunction(string s1, string s2)
{
    return XOR(s1, s2);
}
```

Рисунок 5 - Функция для шифрования/расшифровки

```
//XOR двух строк с двоичными данными
3 references
private string XOR(string s1, string s2)
{
    string result = "";
    for (int i = 0; i < s1.Length; i++)
    {
        bool a = Convert.ToBoolean(Convert.ToInt32(s1[i].ToString()));
        bool b = Convert.ToBoolean(Convert.ToInt32(s2[i].ToString()));
        if (a ^ b)
            result += "1";
        else
            result += "0";
    }
    return result;
}
```

Рисунок 6 - Метод, реализующий «исключающее или»

В завершении раунда происходит склеивание двух строк, R и результата последней операции, далее происходит сдвиг ключа (Рисунок 7) вправо на 2 бита при шифровании или влево, при расшифровке (Рисунок 8) и после этого происходит переход к следующему раунду.

После выполнения всех 16 раундов, необходимо преобразовать сообщения из бинарного вида в символьный вид, для возможности прочтения информации, для этого был реализован вспомогательный метод (Рисунок 9). Для расшифровки сообщения используются обратные действия.

```
//вычисление ключа для следующего раунда шифрования. циклический сдвиг >> 2
2 references
private string KeyToNextRound(string Key)
{
    for (int i = 0; i < iShiftKey; i++)
    {
        Key = Key[Key.Length - 1] + Key;
        Key = Key.Remove(Key.Length - 1);
    }
    return Key;
}
```

Рисунок 7 - Метод, реализующий сдвиг ключа на 2 бита

```
//вычисление ключа для следующего раунда расшифровки. циклический сдвиг << 2
private string KeyToPrevRound(string KeyDecode)
{
    for (int i = 0; i < iShiftKey; i++)
    {
        KeyDecode = KeyDecode + KeyDecode[0];
        KeyDecode = KeyDecode.Remove(0, 1);
    }
    return KeyDecode;
}
```

Рисунок 8 - Метод, реализующий сдвиг ключа на 2 бита

```
//переводим строку с двоичными данными в символьный формат
4 references
private string StringFromBinaryToNormalFormat(string input)
{
    string output = "";
    while (input.Length > 0)
    {
        string char_binary = input.Substring(0, iSizeChar);
        input = input.Remove(0, iSizeChar);
        int a = 0;
        int degree = char_binary.Length - 1;
        foreach (char c in char_binary)
            a += Convert.ToInt32(c.ToString()) * (int)Math.Pow(2, degree--);
        output += ((char)a).ToString();
    }
    return output;
}
```

Рисунок 9 - Метод, переводящий строки из бинарного вида в символьный вид

В итоге мы получаем приложение, способное шифровать или дешифровать информацию с использованием алгоритма симметричного шифрования на основе сети Фейстеля. Интерфейс реализованного приложения представлен ниже (Рис.13).

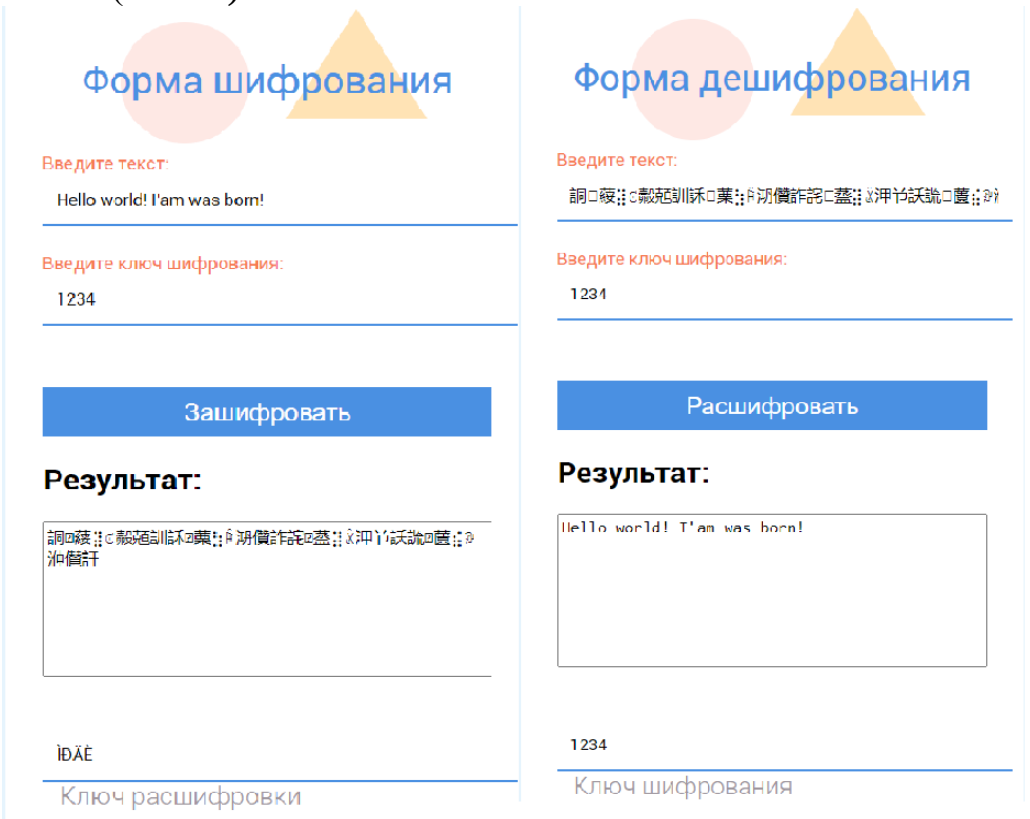

Рисунок 10 - Демонстрация программы реализующей алгоритм DES

Список литературы:

1. Почему сеть Фейстеля работает? Объяснение «на пальцах» [Электронный ресурс]: - Режим доступа: https://habr.com/ru/post/140404/ (дата обращения 12.03.2021).
2. Семенов Ю.А. Алгоритм DES [Электронный ресурс]: учебник / Семенов Ю.А. – Режим доступа: 6.4.1 Алгоритм DES (itep.ru) (дата обращения 07.03.2021).