НАУЧНЫЙ ФОРУМ nauchforum.ru

СХ Студенческая международная научно-практическая конференция «Молодежный научный форум»

РЕАЛИЗАЦИЯ СИММЕТРИЧНОГО ШИФРОВАНИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

Писанко Александр Вячеславович

студент, Ростовский государственный экономический университет, РФ, г. Ростов-на-Дону

Жилина Елена Викторовна

научный руководитель, канд. экон. наук, доцент, Ростовский государственный экономический университет, РФ, г. Ростов-на-Дону

Аннотация. В статье рассматривается способ реализации симметричного шифрования и дешифрования строки используя операцию XOR на языке программирования c++

Abstract. The article discusses a way to implement symmetric encryption and decryption of a string using the XOR operation in the c ++ programming language

Ключевые слова: c++, симметричное шифрования, xor

Keywords: c++, symmetric encryption, xor

Постановка проблемы

При передаче сообщения от одного пользователя к другому требуется его шифрование, когда сервер шифрует тысячи сообщений ежесекундно на первое место выводиться вопрос о производительности.

Результаты исследования

Введение. Серверные решения для программ выполняющих множество ежесекундных операций, в основном, пишутся на языке программирования c++, так как одним из его преимуществ является скорость исполнения инструкций.

Шифрование сообщений является одной из основных частей сервера если необходимо передать сообщение клиенту, так как прежде чем дойти до получателя оно пройдёт через множество узлов, некоторые из которых могут быть скомпрометированы.

Реализация симметричного шифрования сообщения.

Для корректной работы подключим библиотеку iostream

```
1.      #include <iostream>
```

Чтобы зашифровать сообщение реализуем функцию xor_operation, отвечающую за шифрование данных на основе ключа, который поступает на вход в функцию.

```
1.      char* xor_operation(char* data, char* key){
2.          int dataLenght = std::strlen(data);
3.          char* readyData = new char[dataLenght + 1];
4.
5.          std::strcpy(readyData, data);
6.
7.          for(int i = 0; i < std::strlen(data); i++)
8.              readyData[i] = data[i] ^ key[i % (sizeof(key) /
sizeof(char))];
9.
10.         return readyData;
11.     }
```

На 2 строке считываем длину поступившего на вход сообщения, это пригодиться в дальнейшем, на следующей, 3 строке, объявлена переменная readyData, отвечающая за конечное зашифрованное/дешифрованное сообщение.

Далее копируем полученные на входе данные в переменную, объявленную на строке 3. На 8 строке выполняется непосредственное шифрование при помощи операции XOR, главное преимущество которой заключается в том, что можно зашифровать и дешифровать одно и то же сообщение с помощью единственного ключа.

Ниже представлен код функции main, в котором сообщение шифруется и дешифруется с помощью ключа, объявленного на 3 строке.

```
1.      int main()
2.      {
3.          char key[] = "12345678";
4.          char message[] = "Hello world!";
5.          std::cout << "Original message: " << message << std::endl;
6.          auto encmsg = xor_operation(message, key);
7.          std::cout << "Encrypted message: " << encmsg << std::endl;
8.          auto decmsg = xor_operation(encmsg, key);
9.          std::cout << "Decrypted message: " << decmsg << std::endl;
10.       return 0;
11.     }
```

На рисунке 1 изображены данные, выведенные в консоль, сначала выводится первоначальное сообщение, затем зашифрованное ключом, а далее расшифрованное этим же ключом.

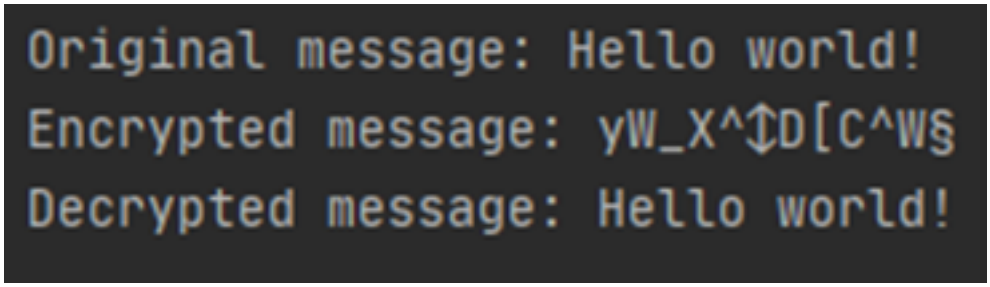

Рисунок 1. Пример работы программы

Вывод

Представленное решение позволяет реализовать симметричное шифрования сообщений на языке программирования c++ и может использоваться как отдельный модуль для сервера.

Список литературы:

1. Simple XOR Encryption/Decryption in C++ [Электронный ресурс]. – Режим доступа: https://kylewbanks.com/blog/Simple-XOR-Encryption-Decryption-in-Cpp, свободный. Дата обращения: 20.12.2020.

2. C/C++ Cryptography — XOR Encryption [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=vzsB790Mw0U&ab_channel=Zer0Mem0ry, свободный. Дата обращения: 8.11.2020.