Отчет по лабораторной работе №7

по дисциплине: Информационная безопасность

Ким Михаил Алексеевич

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования.

# 2 Задание

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

# 3 Теоретическое введение

* Терминал (или «Bash», сокращение от «Bourne-Again shell») — это программа, которая используется для взаимодействия с командной оболочкой. Терминал применяется для выполнения административных задач, например: установку пакетов, действия с файлами и управление пользователями. [1]
* Гамми́рование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле. Например, в поле Галуа суммирование принимает вид операции «исключающее ИЛИ (XOR)» [2].

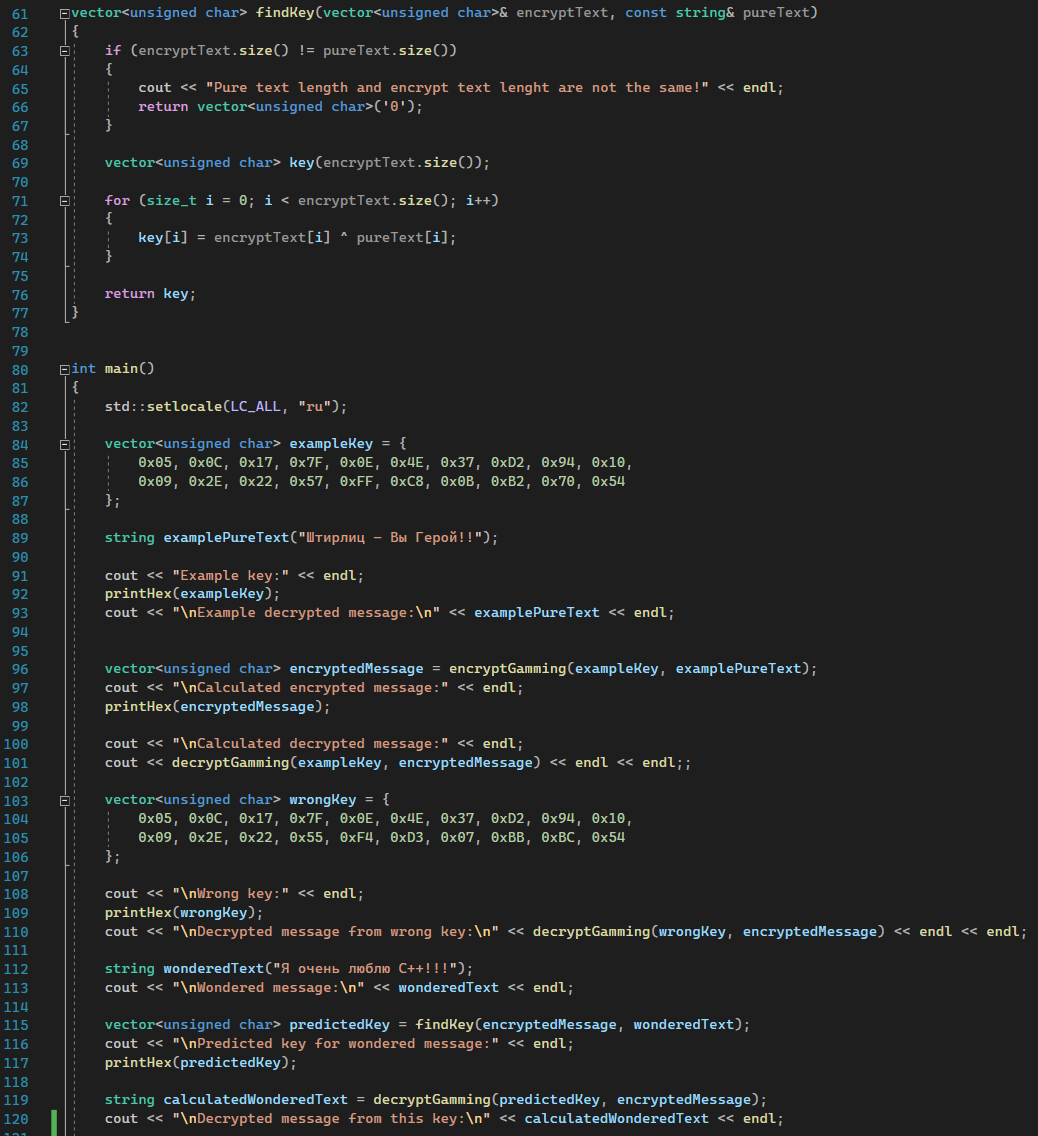
# 4 Выполнение лабораторной работы

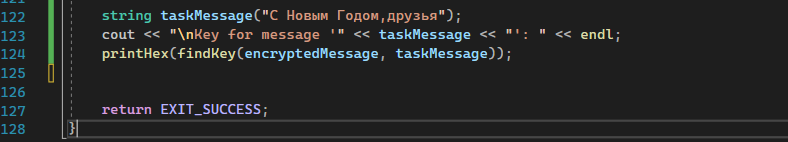
## 4.1 Создание программы

1. Напишем программу на C++, удовлетворяющую всем условиям задания. Программа будет содержать четыре вспомогательные функции: вывод информации в 16-ричной системе счисления, кодирование и декодирование информации, нахождение ключа по исходному тексту и шифротексту. (рис. [1](#fig:01), [2](#fig:02), [3](#fig:03)).

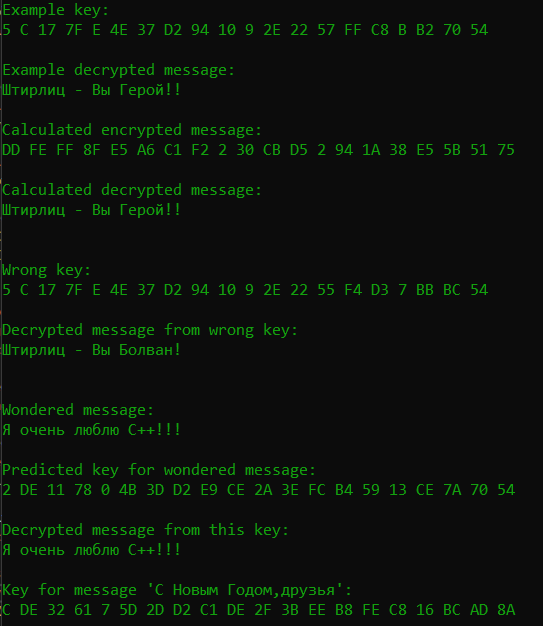
* #include <cstdlib>  
  #include <iostream>  
  #include <vector>  
  #include <string>  
    
  using std::cin;  
  using std::cout;  
  using std::endl;  
  using std::string;  
  using std::vector;  
    
    
  void printHex(vector<unsigned char> vec)  
  {  
   cout << std::hex << std::uppercase;  
    
   for (size\_t i = 0; i < vec.size(); i++)  
   {  
   cout << static\_cast<short>(vec[i]) << ' ';  
   }  
    
   cout << std::dec << std::nouppercase << endl;  
  }  
    
  vector<unsigned char> encryptGamming(const vector<unsigned char>& key, const string& pureText)  
  {  
   if (key.size() != pureText.size())  
   {  
   cout << "Key length and text lenght are not the same!" << endl;  
   return vector<unsigned char>('0');  
   }  
    
   vector<unsigned char> encryptedText(key.size());  
    
   for (size\_t i = 0; i < key.size(); i++)  
   {  
   encryptedText[i] = pureText[i] ^ key[i];  
   }  
    
   return encryptedText;  
  }  
    
  string decryptGamming(const vector<unsigned char>& key, vector<unsigned char>& encryptText)  
  {  
   if (key.size() != encryptText.size())  
   {  
   cout << "Key length and encrypt text lenght are not the same!" << endl;  
   return "0";  
   }  
    
   string decryptedText(key.size(), '0');  
    
   for (size\_t i = 0; i < key.size(); i++)  
   {  
   decryptedText[i] = encryptText[i] ^ key[i];  
   }  
    
   return decryptedText;  
  }  
    
  vector<unsigned char> findKey(vector<unsigned char>& encryptText, const string& pureText)  
  {  
   if (encryptText.size() != pureText.size())  
   {  
   cout << "Pure text length and encrypt text lenght are not the same!" << endl;  
   return vector<unsigned char>('0');  
   }  
    
   vector<unsigned char> key(encryptText.size());  
    
   for (size\_t i = 0; i < encryptText.size(); i++)  
   {  
   key[i] = encryptText[i] ^ pureText[i];  
   }  
    
   return key;  
  }  
    
    
  int main()  
  {  
   std::setlocale(LC\_ALL, "ru");  
    
   vector<unsigned char> exampleKey = {  
   0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,  
   0x09, 0x2E, 0x22, 0x57, 0xFF, 0xC8, 0x0B, 0xB2, 0x70, 0x54  
   };  
    
   string examplePureText("Штирлиц – Вы Герой!!");  
    
   cout << "Example key:" << endl;  
   printHex(exampleKey);  
   cout << "\nExample decrypted message:\n" << examplePureText << endl;  
    
    
   vector<unsigned char> encryptedMessage = encryptGamming(exampleKey, examplePureText);  
   cout << "\nCalculated encrypted message:" << endl;  
   printHex(encryptedMessage);  
    
   cout << "\nCalculated decrypted message:" << endl;  
   cout << decryptGamming(exampleKey, encryptedMessage) << endl << endl;;  
    
   vector<unsigned char> wrongKey = {  
   0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,  
   0x09, 0x2E, 0x22, 0x55, 0xF4, 0xD3, 0x07, 0xBB, 0xBC, 0x54  
   };  
    
   cout << "\nWrong key:" << endl;  
   printHex(wrongKey);  
   cout << "\nDecrypted message from wrong key:\n" << decryptGamming(wrongKey, encryptedMessage) << endl << endl;  
    
   string wonderedText("Я очень люблю C++!!!");  
   cout << "\nWondered message:\n" << wonderedText << endl;  
    
   vector<unsigned char> predictedKey = findKey(encryptedMessage, wonderedText);  
   cout << "\nPredicted key for wondered message:" << endl;  
   printHex(predictedKey);  
    
   string calculatedWonderedText = decryptGamming(predictedKey, encryptedMessage);  
   cout << "\nDecrypted message from this key:\n" << calculatedWonderedText << endl;  
    
   string taskMessage("С Новым Годом,друзья");  
   cout << "\nKey for message '" << taskMessage << "': " << endl;  
   printHex(findKey(encryptedMessage, taskMessage));  
    
    
   return EXIT\_SUCCESS;  
  }

* 
* Figure 1: Однократное гаммирование на C++. Листинг. 1

* 
* Figure 2: Однократное гаммирование на C++. Листинг. 2

* 
* Figure 3: Однократное гаммирование на C++. Листинг. 3

1. Результат работы программы после компиляции и выполнения (рис. [4](#fig:04)).

* 
* Figure 4: Компиляция и выполнение программы

# 5 Анализ результатов

Работа выполнена без каких-либо серьезных нареканий. Единственным моментом, которому было уделено больше внимания, чем он того заслуживает, стал выбор типа данных для хранения текста, шифротекста и ключа.

# 6 Выводы

Освоено на практике применение режима однократного гаммирования.

# Список литературы

1. Терминал Linux [Электронный ресурс]. URL: <{https://www.reg.ru/blog/linux-shpargalka-komandy-terminala-dlya-novichkov/}>.

2. Гаммирование [Электронный ресурс]. Wikipedia Inc. URL: <https://en.wikipedia.org/wiki/XOR_cipher>.