Отчет по лабораторной работе №8

по дисциплине: Информационная безопасность

Ким Михаил Алексеевич

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 2 Задание

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочитать оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P1 и P2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C1 и C2 обоих текстов P1 и P2 при известном ключе. Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

# 3 Теоретическое введение

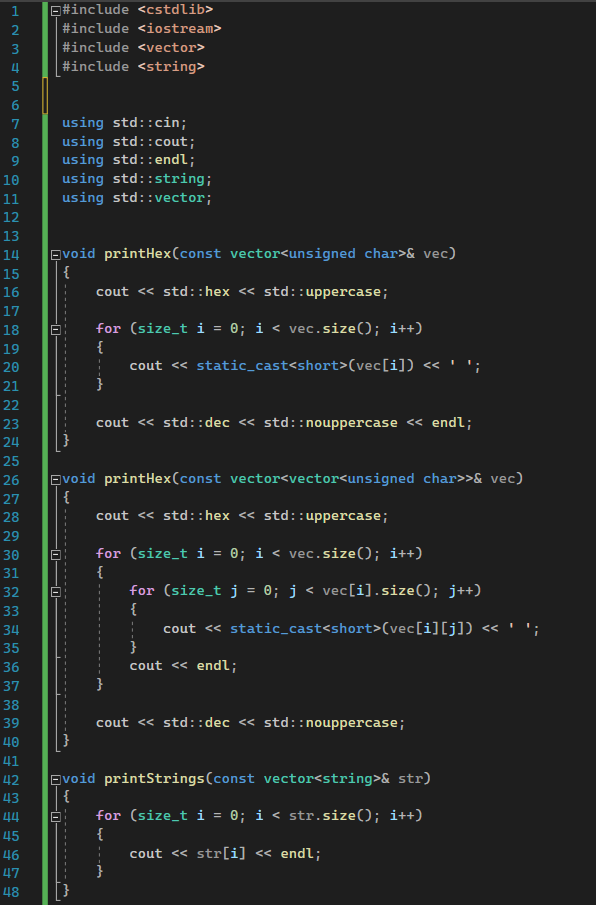
* Терминал (или «Bash», сокращение от «Bourne-Again shell») — это программа, которая используется для взаимодействия с командной оболочкой. Терминал применяется для выполнения административных задач, например: установку пакетов, действия с файлами и управление пользователями. [1]
* Гамми́рование, или 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

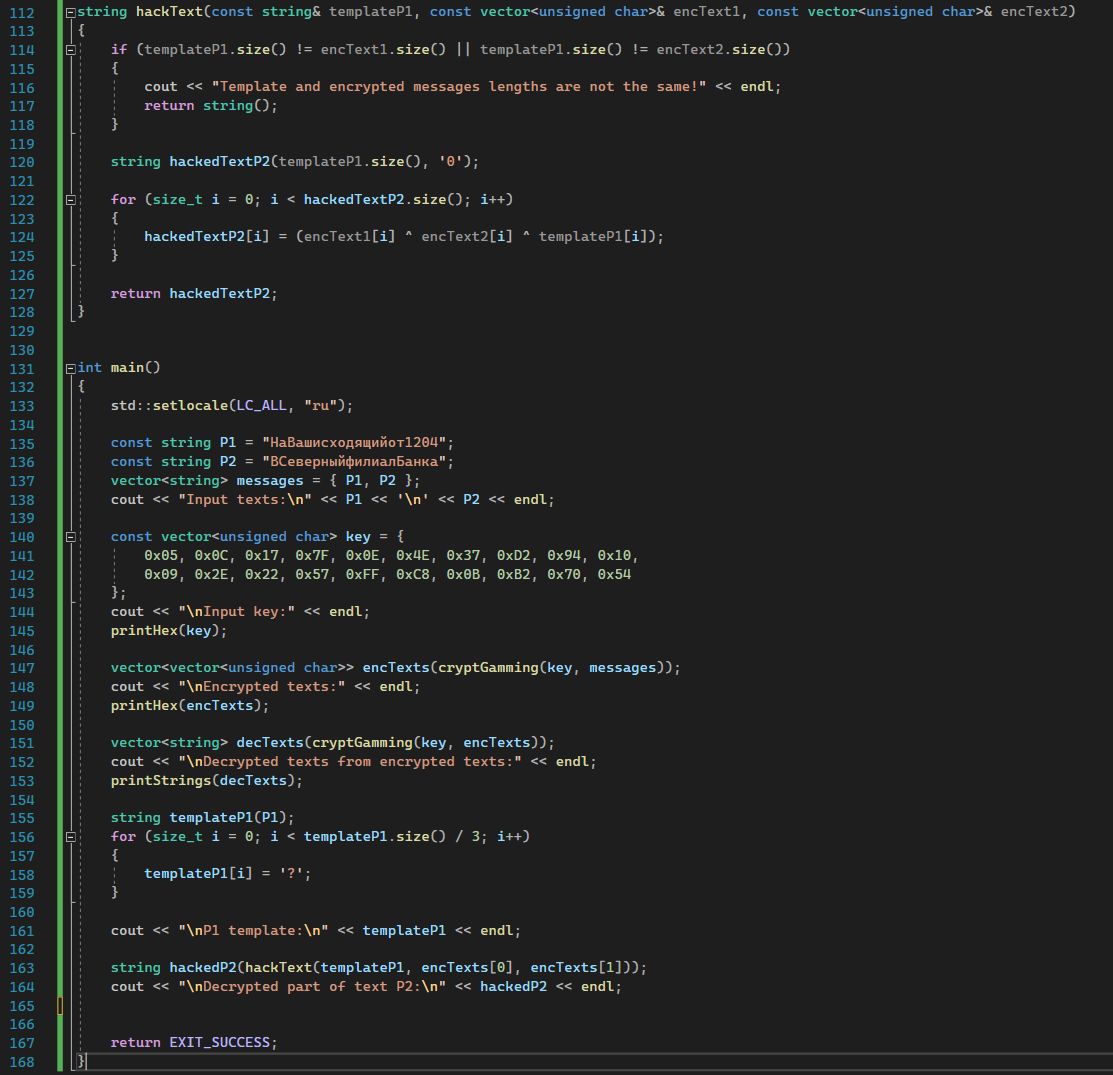
## 4.1 Создание программы

1. Напишем программу на C++, удовлетворяющую всем условиям задания. Программа будет содержать четыре вспомогательные функции и их перегрузки: вывод в консоль информации в 16-ричной системе счисления, вывод в консоль вектора строк, кодирование и декодирование информации, определение фрагмента текста по шаблону и шифротекстам. (рис. [1](#fig:01), [2](#fig:02), [3](#fig:03)).

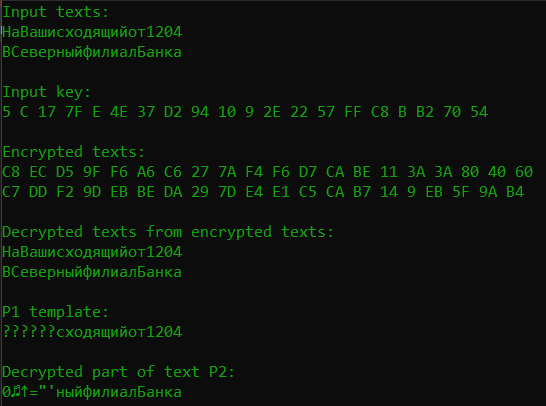
* #include <cstdlib>  
  #include <iostream>  
  #include <vector>  
  #include <string>  
    
    
  using std::cin;  
  using std::cout;  
  using std::endl;  
  using std::string;  
  using std::vector;  
    
    
  void printHex(const vector<unsigned char>& vec)  
  {  
   cout << std::hex << std::uppercase;  
    
   for (size\_t i = 0; i < vec.size(); i++)  
   {  
   cout << static\_cast<short>(vec[i]) << ' ';  
   }  
    
   cout << std::dec << std::nouppercase << endl;  
  }  
    
  void printHex(const vector<vector<unsigned char>>& vec)  
  {  
   cout << std::hex << std::uppercase;  
    
   for (size\_t i = 0; i < vec.size(); i++)  
   {  
   for (size\_t j = 0; j < vec[i].size(); j++)  
   {  
   cout << static\_cast<short>(vec[i][j]) << ' ';  
   }  
   cout << endl;  
   }  
    
   cout << std::dec << std::nouppercase;  
  }  
    
  void printStrings(const vector<string>& str)  
  {  
   for (size\_t i = 0; i < str.size(); i++)  
   {  
   cout << str[i] << endl;  
   }  
  }  
    
  vector<vector<unsigned char>> cryptGamming(const vector<unsigned char>& key, const vector<string>& inpTextVector)  
  {  
   size\_t elemSize = inpTextVector[0].size();  
   for (size\_t i = 0; i < inpTextVector.size(); i++)  
   {  
   if (inpTextVector[i].size() != elemSize)  
   {  
   cout << "Input texts lengths are not the same!" << endl;  
   return vector<vector<unsigned char>>();  
   }  
   }  
    
   if (key.size() != elemSize)  
   {  
   cout << "Key length and input texts length are not the same!" << endl;  
   return vector<vector<unsigned char>>();  
   }  
    
    
   vector<vector<unsigned char>> outTextVector(inpTextVector.size(), vector<unsigned char>(elemSize));  
   for (size\_t i = 0; i < outTextVector.size(); i++)  
   {  
   for (size\_t j = 0; j < elemSize; j++)  
   {  
   outTextVector[i][j] = inpTextVector[i][j] ^ key[j];  
   }  
   }  
    
   return outTextVector;  
  }  
    
  vector<string> cryptGamming(const vector<unsigned char>& key, vector<vector<unsigned char>>& inpTextVector)  
  {  
   size\_t elemSize = inpTextVector[0].size();  
   for (size\_t i = 0; i < inpTextVector.size(); i++)  
   {  
   if (inpTextVector[i].size() != elemSize)  
   {  
   cout << "Input texts lengths are not the same!" << endl;  
   return vector<string>();  
   }  
   }  
    
   if (key.size() != elemSize)  
   {  
   cout << "Key length and input texts length are not the same!" << endl;  
   return vector<string>();  
   }  
    
    
   vector<string> outTextVector(inpTextVector.size(), string(elemSize, '0'));  
   for (size\_t i = 0; i < outTextVector.size(); i++)  
   {  
   for (size\_t j = 0; j < elemSize; j++)  
   {  
   outTextVector[i][j] = inpTextVector[i][j] ^ key[j];  
   }  
   }  
    
   return outTextVector;  
  }  
    
  string hackText(const string& templateP1, const vector<unsigned char>& encText1, const vector<unsigned char>& encText2)  
  {  
   if (templateP1.size() != encText1.size() || templateP1.size() != encText2.size())  
   {  
   cout << "Template and encrypted messages lengths are not the same!" << endl;  
   return string();  
   }  
    
   string hackedTextP2(templateP1.size(), '0');  
    
   for (size\_t i = 0; i < hackedTextP2.size(); i++)  
   {  
   hackedTextP2[i] = (encText1[i] ^ encText2[i] ^ templateP1[i]);  
   }  
    
   return hackedTextP2;  
  }  
    
    
  int main()  
  {  
   std::setlocale(LC\_ALL, "ru");  
    
   const string P1 = "НаВашисходящийот1204";  
   const string P2 = "ВСеверныйфилиалБанка";  
   vector<string> messages = { P1, P2 };  
   cout << "Input texts:\n" << P1 << '\n' << P2 << endl;  
    
   const vector<unsigned char> key = {  
   0x05, 0x0C, 0x17, 0x7F, 0x0E, 0x4E, 0x37, 0xD2, 0x94, 0x10,  
   0x09, 0x2E, 0x22, 0x57, 0xFF, 0xC8, 0x0B, 0xB2, 0x70, 0x54  
   };  
   cout << "\nInput key:" << endl;  
   printHex(key);  
    
   vector<vector<unsigned char>> encTexts(cryptGamming(key, messages));  
   cout << "\nEncrypted texts:" << endl;  
   printHex(encTexts);  
    
   vector<string> decTexts(cryptGamming(key, encTexts));  
   cout << "\nDecrypted texts from encrypted texts:" << endl;  
   printStrings(decTexts);  
    
   string templateP1(P1);  
   for (size\_t i = 0; i < templateP1.size() / 3; i++)  
   {  
   templateP1[i] = '?';  
   }  
    
   cout << "\nP1 template:\n" << templateP1 << endl;  
    
   string hackedP2(hackText(templateP1, encTexts[0], encTexts[1]));  
   cout << "\nDecrypted part of text P2:\n" << hackedP2 << endl;  
    
    
   return EXIT\_SUCCESS;  
  }

* 
* Figure 1: Однократное гаммирование различных текстов одним ключом на C++. Листинг. 1

* 
* Figure 2: Однократное гаммирование различных текстов одним ключом на C++. Листинг. 2

* 
* Figure 3: Однократное гаммирование различных текстов одним ключом 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. Обожаю C++.

# 6 Выводы

Освоено на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 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>.