

Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

**Кафедра вычислительных технологий**

**Лабораторная работа №3**

**по курсу** «**Защита информации**»

**Тема: «Изучение алгоритмов хеширования и ЭЦП с использованием криптобиблиотек»**

|  |  |
| --- | --- |
|  | **Выполнил** |
| **Студент** | Швец Григорий Владиславович |
| **Группа** | А-06-19 |
| **Дата** | 29.04.2023 |

|  |  |
| --- | --- |
|  | **Принял** |
| **Преподаватель** | к.т.н. Андреева Ирина Николаевна |
| **Оценка** |  |
| **Дата** |  |
| **Подпись** |  |

Москва 2023

Содержание

[**Цель работы** 3](#_Toc133776084)

[**Задание** 3](#_Toc133776085)

[**Исходный код программ** 3](#_Toc133776086)

# **Цель работы**

Целью данной лабораторной работы является изучение алгоритмов хеширования и ЭЦП с использованием криптобиблиотек.

# **Задание**

1. Изучить материал по алгоритмам хеширования и формирования ЭЦП.
2. Подготовить текстовый файл, содержимое которого будет хешироваться и подписываться ЭЦП в соответствии с пунктами задания.

3. Используя алгоритмы хеширования или ЭЦП (см.табл.), сформировать хеш **или** ЭЦП и записать их в конец файла, **имя которого вводится с клавиатуры**.

4. Внести изменения в исходный текстовый файл и обеспечить возможность поверки нового хеша **или** ЭЦП с записанными в файле, **имя которого вводится с клавиатуры. Предусмотреть** вывод на экран сообщения об отсутствии или наличии изменений в файле, и вывести его актуальное на момент проверки содержимое на экран.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Библиотека** | **ХЕШ-алгоритм** | **ЭЦП-алгоритм** |
| 17 | Libgcrypt | Whirlpool | ГОСТ 34.10-2001 |

# **Исходный код программ**

|  |
| --- |
| #include <iostream>  #include <gcrypt.h>  #include <fstream>  #include <cstring>  #include <string>  #include <sstream>  #include <iomanip>  using namespace std;  int main() {  gcry\_error\_t gcryError;  gcry\_md\_hd\_t gcryMdHd;  string InputFile, OutputFile;  string PrevHashValue;  bool first\_iteration = true;  while (true) {  cout << "Please, enter the name of input file or write 'q' to exit: ";  cin >> InputFile;  cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n";  if (InputFile == "q") {  break;  }  ifstream file(InputFile);  if (!file) {  cout << "File open error!\n";  continue;  }  else {  cout << "All OK! File is open!\n";  cout << "====================================================================================|\n";  }  ostringstream oss;  oss << file.rdbuf();  string str = oss.str();  const char\* plaintext = str.c\_str();  cout << "Input file content: " << str.c\_str();  cout << "\n====================================================================================|\n";  gcryError = gcry\_md\_open(&gcryMdHd, GCRY\_MD\_WHIRLPOOL, 0);  if (gcryError) {  cout << "gcry\_md\_open error: " << gcry\_strsource(gcryError) << endl;  return 1;  }  gcry\_md\_write(gcryMdHd, plaintext, strlen(plaintext));  size\_t digest\_size = gcry\_md\_get\_algo\_dlen(GCRY\_MD\_WHIRLPOOL);  cout << "Current digest size is: " << digest\_size;  cout << "\n====================================================================================|";  unsigned char\* digest = new unsigned char[digest\_size];  gcry\_md\_final(gcryMdHd);  memcpy(digest, gcry\_md\_read(gcryMdHd, GCRY\_MD\_WHIRLPOOL), digest\_size);  gcry\_md\_close(gcryMdHd);  cout << "\nWhirlpool hash of the plaintext is: ";  ostringstream output\_stream;  for (int i = 0; i < digest\_size; i++) {  printf("%02x", digest[i]);  output\_stream << hex << setw(2) << setfill('0') << static\_cast<int>(digest[i]); // Store hex string  }  cout << endl;  cout << "====================================================================================|\n" << endl;  if (!first\_iteration && PrevHashValue != output\_stream.str()) {  cout << "---ATTENTION!---\nFile content is changed.\n";  }  else {  cout << "File content is not changed.\n";  }  cout << "Enter the output file name: ";  cin >> OutputFile;  ofstream output(OutputFile);  output << output\_stream.str();  delete[] digest;  PrevHashValue = output\_stream.str();  first\_iteration = false;  }  return 0;  }  //// Генерация ключевой пары  //gcry\_sexp\_t keypair, public\_key, private\_key;  //gcry\_error\_t error = gcry\_sexp\_build(&keypair, NULL, "%S");  //if (error) {  // cerr << "Ошибка при генерации ключевой пары: " << gcry\_strerror(error) << endl;  // return 1;  //}  //public\_key = gcry\_sexp\_find\_token(keypair, "public-key", 0);  //private\_key = gcry\_sexp\_find\_token(keypair, "private-key", 0);  //// Хэширование данных  //gcry\_md\_hd\_t hash;  //const char\* data = "gadhgaepojgnqerlkugnqeripgnqpero!";  //size\_t data\_length = strlen(data);  //error = gcry\_md\_open(&hash, GCRY\_MD\_GOSTR3411\_94, GCRY\_MD\_FLAG\_SECURE);  //if (error) {  // cerr << "Ошибка при открытии контекста хэширования: " << gcry\_strerror(error) << endl;  // return 1;  //}  //gcry\_md\_write(hash, data, data\_length);  //unsigned char\* digest = gcry\_md\_read(hash, GCRY\_MD\_GOSTR3411\_94);  //size\_t digest\_length = gcry\_md\_get\_algo\_dlen(GCRY\_MD\_GOSTR3411\_94);  //// Подписание данных  //gcry\_sexp\_t signature;  ////error = gcry\_pk\_sign(&signature, hash, private\_key);  //gcry\_sexp\_t hash\_sexp;  //gcry\_sexp\_build(&hash\_sexp, NULL, "(data (flags pkcs1) (hash GOSTR3411\_94 %b))", digest, digest\_length);  //error = gcry\_pk\_sign(&signature, hash\_sexp, private\_key);  ////error = gcry\_pk\_sign(&signature, digest, digest\_length, private\_key, NULL, NULL);  ////error = gcry\_pk\_sign(&signature, gcry\_md\_read(hash, GCRY\_MD\_GOSTR3411\_94), private\_key);  //if (error) {  // cerr << "Ошибка при подписании данных: " << gcry\_strerror(error) << endl;  // return 1;  //}  //gcry\_sexp\_t r = gcry\_sexp\_find\_token(signature, "r", 0);  //gcry\_sexp\_t s = gcry\_sexp\_find\_token(signature, "s", 0);  //const char\* r\_value = gcry\_sexp\_nth\_data(r, 1, &digest\_length);  //const char\* s\_value = gcry\_sexp\_nth\_data(s, 1, &digest\_length);  //cout << "Подпись: " << endl;  //for (size\_t i = 0; i < digest\_length; i++) {  // printf("%02X", r\_value[i]);  //}  //for (size\_t i = 0; i < digest\_length; i++) {  // printf("%02X", s\_value[i]);  //}  //cout << endl;  //// Проверка подписи  //gcry\_sexp\_t data\_hash;  //error = gcry\_sexp\_build(&data\_hash, NULL, "(data (hash GOSTR3411\_94 %b))", digest, digest\_length);  //if (error) {  // cerr << "Ошибка при построении хэша данных: " << gcry\_strerror(error) << endl;  // return 1;  //}  //error = gcry\_pk\_verify(signature, data\_hash, public\_key); |