

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет
о практической работе №4
на тему «Использование криптографических библиотек»

Дисциплина: Методы и средства
программирования

Группа: 22ПИ2

Выполнил: Никитина М. А.

Количество баллов:

Дата сдачи:

Принял: Н. А. Сидоров

2023

1 Цель работы: освоить использование криптографических библиотек.

2 Задание на практическую работу

2.1 Создать в своем аккаунте на `github.com` репозиторий с именем `CryptoProg` для выполнения данного задания и сделать его локальную копию.

2.2 Создать в репозитории каталог `hash` для реализации программы хэширования. Подготовить в каталоге файлы `.gitignore` для управления исключениями и `Makefile` для управления сборкой.

2.3 Разработать и реализовать программу хэширования содержимого файла с использованием библиотеки `Crypto++`. Алгоритм хэширования — любой из предоставляемых библиотекой. Формат вывода результата хэш-функции — строка шестнадцатеричных цифр.

2.4 Протестировать работоспособность разработанной программы, сравнив результат ее работы с результатами работы других программ для выбранной хэш-функции.

2.5 Создать в репозитории каталог `cipher` программы зашифрования/расшифрования данных. Подготовить в каталоге файлы `.gitignore` для управления исключениями и `Makefile` для управления сборкой.

2.6 Разработать и реализовать программу зашифрования/расшифрования содержимого файлов с использованием библиотеки `Crypto++`. Тип алгоритма шифрования — блочный. Режим алгоритма — CBC. Алгоритм шифрования — любой блочный алгоритм, предоставляемый библиотекой. Ключ шифрования — вырабатывается из пароля. Интерфейс программы должен предоставлять возможность выбирать режим работы (зашифрование/расшифрование), выбирать файл с исходными данными, выбирать файл для записи результата, указывать пароль (для формирования ключа).

2.7 Протестировать работоспособность разработанной программы последовательно выполнив процедуры зашифрования и расшифрования, и сравнив содержимое исходного файла и расшифрованного файла.

2.8 Сделать отчет о выполненной работе, включив в него ссылку на свой репозиторий на сайте github.com.

3 Порядок выполнения работы

3.1 Был создан в аккаунте на github.com репозиторий с именем CryptoProg для выполнения данного задания и была сделана его локальная копия.

3.2 Был создан в репозитории каталог hash для реализации программы хэширования. Были подготовлены в каталоге файлы .gitignore для управления исключениями и Makefile для управления сборкой.

3.3 Была разработана и реализована программа хэширования содержимого файла с использованием библиотеки Crypto++. Алгоритм хэширования — SHA-256. Формат вывода результата хэш-функции — строка шестнадцатеричных цифр. Результат представлен на рисунке 1. Код программы представлен в приложении А.

```
stud@virtdeb:~/C++Projects/Pr4$ g++ -o main main.cpp -lcryptopp
stud@virtdeb:~/C++Projects/Pr4$ ./main
Введите путь к файлу:
/home/stud/Desktop/text.txt
File Hash (SHA-256): B22B009134622B6508D756F1062455D71A7026594EACB0BADF81F4F6779
29EBE
```

Рисунок 1 - Результат хэширования

3.4 Была протестирована работоспособность разработанной программы, сравнив результат ее работы с результатами работы сторонней утилиты OpenSSL для выбранной хэш-функции. Результат представлен на рисунке 2.

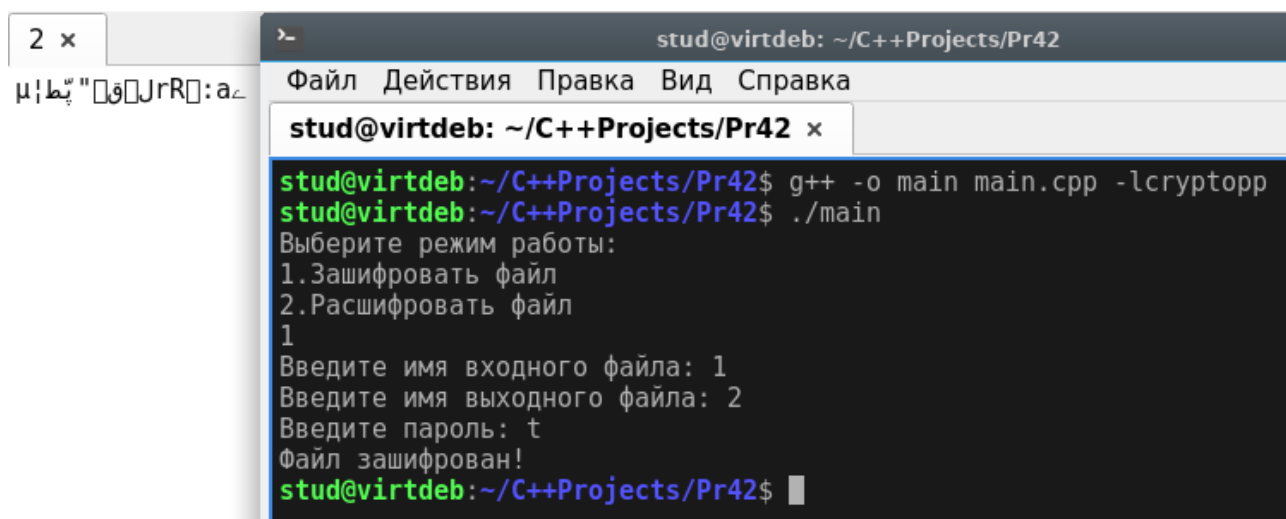
```
stud@virtdeb:~/C++Projects/Pr4$ g++ -o main main.cpp -lcryptopp
stud@virtdeb:~/C++Projects/Pr4$ ./main
Введите путь к файлу:
/home/stud/Desktop/text.txt
File Hash (SHA-256): B22B009134622B6508D756F1062455D71A7026594EACB0BADF81F4F6779
29EBE
сравнив результат ее работы с результатами работы сторонней утилиты
stud@virtdeb:~/C++Projects/Pr4$ openssl sha256 /home/stud/Desktop/text.txt
SHA256(/home/stud/Desktop/text.txt)= b22b009134622b6508d756f1062455d71a7026594ea
cb0badf81f4f677929ebe
```

Рисунок 2 - Результат сравнения результатов

3.5 Был создан в репозитории каталог cipher программы зашифрования/расшифрования данных. Были подготовлены в каталоге файлы .gitignore для управления исключениями и Makefile для управления сборкой.

3.6 Был разработан и реализован программ зашифрования/расшифрования содержимого файлов с использованием библиотеки Crypto++. Тип алгоритма шифрования — блочный. Режим алгоритма — CBC. Алгоритм шифрования — любой блочный алгоритм, предоставляемый библиотекой. Ключ шифрования — вырабатывается из пароля. Интерфейс программы предоставляет возможность выбирать режим работы (зашифрование/расшифрование), выбирать файл с исходными данными, выбирать файл для записи результата, указывать пароль (для формирования ключа). Код программы представлен в приложении Б.

3.7 Была протестирована работоспособность разработанной программы последовательно выполнив процедуры зашифрования и расшифрования, и сравнив содержимое исходного файла и расшифрованного файла. Результат представлен на рисунках 3, 4.



```
stud@virtdeb: ~/C++Projects/Pr42
Файл Действия Правка Вид Справка
stud@virtdeb: ~/C++Projects/Pr42 x
stud@virtdeb:~/C++Projects/Pr42$ g++ -o main main.cpp -lcryptopp
stud@virtdeb:~/C++Projects/Pr42$ ./main
Выберите режим работы:
1.Зашифровать файл
2.Расшифровать файл
1
Введите имя входного файла: 1
Введите имя выходного файла: 2
Введите пароль: t
Файл зашифрован!
stud@virtdeb:~/C++Projects/Pr42$
```

Рисунок 3 - Зашифрование файла «1» с содержимым «Hello»

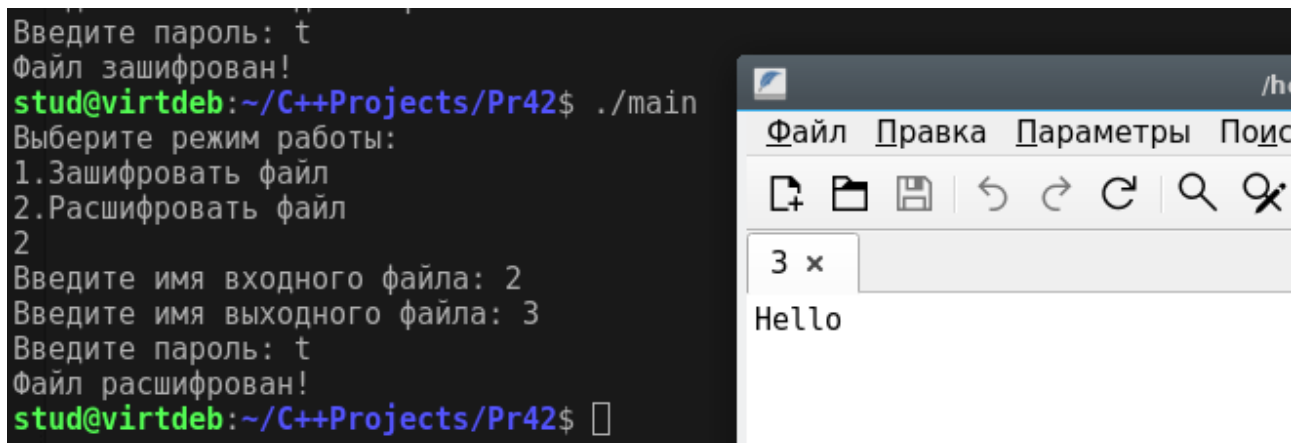


Рисунок 4 - Расшифрование файла

3.8 Был сделан отчет о выполненной работе, включив в него ссылку на свой репозиторий на сайте github.com.

4 Выводы

С помощью данной практической работы были освоены технологии сетевого взаимодействия в программах. Полный результат работы можно посмотреть по ссылке: <https://github.com/Popitka994/CryptoProg>.

Приложение А

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cryptopp/sha.h>
#include <cryptopp/hex.h>

using namespace CryptoPP;
using namespace std;

string calculateFileHash(const string& filePath) {
    ifstream file(filePath, ios::binary);
    if (!file.is_open()) {
        std::cerr << "Error: Cannot open file: " <<
filePath << std::endl;
        return "";
    }

    SHA256 hash;
    byte buffer[1024];
    while (!file.eof()) {
        file.read(reinterpret_cast<char*>(buffer),
sizeof(buffer));
        hash.Update(buffer, file.gcount());
    }

    file.close();

    size_t size = hash.DigestSize();
```

```

    byte digest[size];
    hash.Final(digest);

    std::string hexHash;
    HexEncoder encoder(new StringSink(hexHash));
    encoder.Put(digest, size);
    encoder.MessageEnd();

    return hexHash;
}

int main() {
    string filePath;
    cout<<"Введите путь к файлу:"<<endl;
    cin>>filePath;
    std::string fileHash = calculateFileHash(filePath);

    if (!fileHash.empty()) {
        std::cout << "File Hash (SHA-256): " << fileHash
        << std::endl;
        cout<<endl;
    }

    return 0;
}

```

Приложение Б

```
#include <iostream>
#include <fstream>
#include <string>
#include <cryptopp/cryptlib.h>
#include <cryptopp/aes.h>
#include <cryptopp/modes.h>
#include <cryptopp/filters.h>
#include <cryptopp/files.h>
#include <cryptopp/sha.h>
#include <cryptopp/hex.h>
#include <cryptopp/pwdbased.h>
using namespace CryptoPP;

SecByteBlock DeriveKey(const std::string& password)
{
    SecByteBlock derived(AES::DEFAULT_KEYLENGTH);
    PKCS5_PBKDF2_HMAC<SHA256> pbkdf;
    const byte* salt = (const byte*)"somesalt";
    size_t saltLen = 8;
    pbkdf.DeriveKey(derived, derived.size(), 0, (const
        byte*)password.data(), password.size(), salt,
    saltLen, 1000, 0.0);
    return derived;
}

void ProcessFile(const std::string& inputFile, const
std::string& outputFile, const std::string& password,
bool encrypt)
```



```

{
    try {
        std::ifstream in(inputFile, std::ios::binary);
        std::ofstream out(outputFile, std::ios::binary);
        if (!in.is_open() || !out.is_open()) {
            std::cout << "Ошибка открытия файла!" <<
std::endl;
            return;
        }
        SecByteBlock key = DeriveKey(password);
        byte iv[AES::BLOCKSIZE];
        memset(iv, 0x00, AES::BLOCKSIZE);
        if (encrypt) {
            CBC_Mode<AES>::Encryption enc;
            enc.SetKeyWithIV(key, key.size(), iv);
//Шифрование
            FileSource fileSrc(in, true, new
StreamTransformationFilter(enc, new FileSink(out)));
            fileSrc.PumpAll();
            fileSrc.Flush(true);
            std::cout << "Файл зашифрован!" << std::endl;
        }
        else {
            CBC_Mode<AES>::Decryption dec;
            dec.SetKeyWithIV(key, key.size(), iv);
            FileSource fileSrc(in, true, new
StreamTransformationFilter(dec, new
FileSink(out)));//РАсшифрование
            fileSrc.PumpAll();

```

```

        fileSrc.Flush(true);
        std::cout << "Файл расшифрован!" <<
std::endl;
    }
}
catch (const Exception& ex) {
    std::cerr << "Crypto++ исключение: " << ex.what()
<<
        std::endl;
}
}
int main()
{
    std::string inputFile, outputFile, password;
    int choice;
    std::cout << "Выберите режим работы:\n1.Зашифровать
файл\n2.Расшифровать файл\n";
    std::cin >> choice;
    std::cout << "Введите имя входного файла: ";
    std::cin >> inputFile;
    std::cout << "Введите имя выходного файла: ";
    std::cin >> outputFile;
    std::cout << "Введите пароль: ";
    std::cin >> password;
    bool encrypt = (choice == 1);
    ProcessFile(inputFile,  outputFile,  password,
encrypt);
    return 0;
}

```