Лабораторная работа #5 «Программирование SSL-клиента» Вариант #5	Выполнил	Ноздренков С.В.
	Группа	ЭВМ-1.Н
	Проверил	Жариков Д. Н.
	Подпись	

Цель работы

Цель работы: Изучение программного интерфейса OpenSSL. Изучение порядка действий, выполняемых клиентом SSL.

Задание

Разработать две программы: клиент и сервер, позволяющие получать список файлов указанного каталога на компьютере, где функционирует сервер, переименовывать, копировать в другой каталог или удалять указанный файл, запускать программу на выполнение, выполнять перезагрузку компьютера.

Отличия от 1/2 лабораторной работы:

Crypto.hpp

```
#include <iostream>
#include <string>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <openssl/x509.h>
using namespace std;

void dbg(const string &s) { cout << s << endl; }

/**
@brief Класс шифрования данных
*/
class crypto_t
{
    string BIO_to_string(BIO *bio)
    {
        int len = BIO_pending(bio);
        char *buf = new char[len + 1];
        BIO_read(bio, buf, len);
        string res = string(buf);
        delete[] buf;
        return res;
    }

    BIO * string_to_BIO(const string &s)
    {
        BIO *bio = NULL;

        const int len = s.size();
        char *buf = new char[len + 1];
        memset(buf, 0, len + 1);
        memcpy(buf, s.c_str(), len);
        BIO_write(bio, buf, len);
        delete[] buf;
        return bio;
    }
```

```
    string public_key;
    string private_key;

public:

    /**
    @brief Инициализация приватного и публичного ключей
    @param name - имена ключей
    */
    crypto_t()
    {
        // генерируем ключи
        RSA *rsa = RSA_generate_key(2048, RSA_F4, 0, 0);

        // сохраняем публичный ключ
        BIO *bio = BIO_new(BIO_s_mem());
        PEM_write_bio_RSAPublicKey(bio, rsa);
        public_key = BIO_to_string(bio);
        BIO_free_all(bio);

        // сохраняем приватный ключ

        bio = BIO_new(BIO_s_mem());
        PEM_write_bio_RSAPrivateKey(bio, rsa, 0, 0, 0, 0, 0);
        private_key = BIO_to_string(bio);
        BIO_free_all(bio);

        // очищаем память
        RSA_free(rsa);
    }

    /**
    @brief Возвращает публичный свой ключ
    */
    string get_public_key()
    {
        return public_key;
    }

    /**
    @brief Расшифровывает сообщение msg своим приватным ключем
    @param msg - зашифрованное сообщение
    */
    string decrypt(const string &msg)
    {
        BIO *bio = string_to_BIO(private_key);
        RSA *priv_key = PEM_read_bio_RSAPrivateKey(bio, 0, 0, 0);

        string ans;
        int len = RSA_size(priv_key);
        for (int i = 0; i < msg.size(); i += len)
        {
            string s = msg.substr(i, len);
            unsigned char *from = new unsigned char[len + 1];
            memset(from, 0, len + 1);
            memcpy(from, s.c_str(), len);

            unsigned char *to = NULL;
            int olen = RSA_private_decrypt(len, from, to, priv_key, RSA_PKCS1_PADDING);
            if (olen < 0)
            {
                delete[] from;
                delete[] to;
                return string();
            }
```

```
            ans += string(reinterpret_cast<char *>(to));


            delete[] from;
            delete[] to;
        }

        BIO_free_all(bio);
        RSA_free(priv_key);

        return ans;
    }

    /**
    @brief Шифрует сообщение msg при помощи публичного ключа public_key
    @param msg - сообщение
    @param public_key - публичный ключ
    */
    string encrypt(const string &msg, const string &public_key)
    {
        BIO *bio = string_to_BIO(public_key);
        RSA *pub_key = PEM_read_bio_RSAPublicKey(bio, 0, 0, 0);

        string ans;

        int len = RSA_size(pub_key) - 11;
        for (int i = 0; i < msg.size(); i += len - 1)
        {
            string s = msg.substr(i, len - 1);
            unsigned char *from = new unsigned char[len];
            memset(from, 0, len);
            memcpy(from, s.c_str(), s.size());

            unsigned char *to = NULL;
            int olen = RSA_public_encrypt(len, from, to, pub_key, RSA_PKCS1_PADDING);
            if (olen != RSA_size(pub_key))
            {
                delete[] from;
                delete[] to;
                return string();
            }

            ans += string(reinterpret_cast<char *>(to));


            delete[] from;
            delete[] to;
        }

        BIO_free_all(bio);
        RSA_free(pub_key);

        return ans;
    }
};
```