МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

Отчет о курсовой работе

**Программная реализация**

**криптографических протоколов**

Москва-2017

**Цель работы**

Закрепление и углубление знаний по методам криптографической защиты информации и выработка практических навыков деятельности специалиста по защите информации по разработке средств криптографической защиты информационных систем на современном научно-техническом уровне в соответствии с требованиями действующих международных и национальных стандартов.

**Задание**

Реализация криптографического протокола аутентификации участников 5.1.1[1] с использованием схемы цифровой подписи ECDSA и криптографической хэш-функции SHA-3 (Keccak) (1 вариант).

# Описание алгоритмов

## Однопроходной односторонний протокол аутентификации 5.1.1[1]

Согласно протоколу, сторона B аутентифицирует сторону A.

Алгоритм:

1. A отправляет (описан ниже) и, опционально, свой сертификат B;
2. При получении сообщения, содержащего , B производит следующие шаги:
   1. Убеждается в том, что у него есть валидный публичный ключ A путём проверки сертификата или иным путём;
   2. Проверяет посредством проверки подписи A, содержащейся в токене, проверки метки времени или порядкового номера, а также проверки соответствия указанного в токене идентификатора действительному идентификатору B.

В случае успешного прохождения проверок считается, что сторона A успешно прошла аутентификацию.

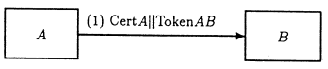


Рисунок 1. Схема протокола

Структура токена:

, где

– метка времени или порядковый номер ,

– идентификатор B,

– опциональное поле для данных приложения,

– цифровая подпись закрытым ключом A,

– опциональное поле для данных приложения (если его значение не известно B и его нельзя извлечь из самой подписи, то оно также должно содержаться в ).

В данной работе используется метка времени – UNIX-время, представленное 4-х байтовым знаковым целым; идентификатор – символ ‘b’, в кодировке ASCII; – размер в байтах цифровой подписи , представленный 4-х байтовым знаковым целым; цифровая подпись – цифровая подпись по схеме ECDSA с использованием кривой secp521r1 и хэш функции SHA3-512, в DER кодировке; и не используются.

# Описание используемых интерфейсов криптографических библиотек

В данной работе используются библиотеки libcrypto (openssl) для генерации ключей, вычисления и проверки подписей, и librhash для вычисления хэш-функции.

## Интерфейс библиотеки libcrypto

### Используемые структуры данных

#### Заголовочный файл <openssl/ossl\_typ.h>

EVP\_PKEY\_CTX – контекст алгоритма публичного ключа.

EVP\_PKEY – приватный/публичный ключ. Также содержит параметры алгоритма.

EC\_KEY – приватный/публичный ключ для эллиптических кривых.

#### Заголовочный файл <openssl/ec.h>

ECDSA\_SIG – цифровая подпись по алгоритму ECDSA.

### Используемые подпрограммы

#### Заголовочный файл <openssl/crypto.h>

ERR\_load\_crypto\_strings() – инициализация: загрузка человекочитаемых строк ошибок.

CRYPTO\_cleanup\_all\_ex\_data() – очистка дополнительной памяти.

#### Заголовочный файл <openssl/err.h>

ERR\_free\_strings() – выгрузка строк ошибок.

#### Заголовочный файл <openssl/evp.h>

OpenSSL\_add\_all\_algorithms() – инициализация: загрузка алгоритмов и конфигурации.

EVP\_PKEY\_CTX\_new\_id(int id, ENGINE \*e) – создание нового контекста, где

id – идентификатор алгоритма,

e – криптографический движок.

EVP\_PKEY\_paramgen\_init(EVP\_PKEY\_CTX \*ctx) – инициализация генерации параметров алгоритма, где

ctx – контекст алгоритма.

EVP\_PKEY\_paramgen(EVP\_PKEY\_CTX \*ctx, EVP\_PKEY \*\*ppkey) – генерация параметров алгоритма, где

ctx – контекст алгоритма,

ppkey – ключ.

EVP\_PKEY\_CTX\_new(EVP\_PKEY \*pkey, ENGINE \*e) – создание нового контекста, где

pkey – ключ,

e – криптографический движок.

EVP\_PKEY\_keygen\_init(EVP\_PKEY\_CTX \*ctx) – инициализация генерации ключа, где

ctx – контекст алгоритма.

EVP\_PKEY\_keygen(EVP\_PKEY\_CTX \*ctx, EVP\_PKEY \*\*ppkey) – генерация ключа, где

ctx – контекст алгоритма,

ppkey – ключ.

EVP\_PKEY\_get1\_EC\_KEY(EVP\_PKEY \*pkey) – получение ключа для алгоритма эллиптических кривых из универсальной структуры ключа, где

pkey – ключ.

EVP\_PKEY\_CTX\_free(EVP\_PKEY\_CTX \*ctx) – освобождение памяти контекста, где

ctx – контекст алгоритма.

EVP\_PKEY\_free(EVP\_PKEY \*pkey) – освобождение памяти ключа, где

pkey – ключ.

EVP\_cleanup() – выгрузка алгоритмов.

#### Заголовочный файл <openssl/pem.h>

PEM\_write\_PrivateKey(FILE \*fp, EVP\_PKEY \*x, const EVP\_CIPHER \*enc, unsigned char \*kstr, int klen, pem\_password\_cb \*cb, void \*u) – запись приватного ключа в файл в PEM кодировке, где

fp – файловый дескриптор,

x – ключ,

enc – алгоритм шифрования ключа,

kstr – парольная фраза,

klen – длина парольной фразы,

cb – функция обратного вызова для запроса парольной фразы,

u – параметр для функции обратного вызова.

PEM\_write\_PUBKEY(FILE \*fp, EVP\_PKEY \*x) – запись публичного ключа в файл в PEM кодировке, где

fp – файловый дексриптор,

x – ключ.

PEM\_read\_PrivateKey(FILE \*fp, EVP\_PKEY \*\*x, pem\_password\_cb \*cb, void \*u) – чтение приватного ключа в PEM кодировке из файла, где

fp – файловый дескриптор,

x – ключ,

cb – функция обратного вызова для запроса парольной фразы,

u – параметр для функции обратного вызова.

PEM\_read\_PUBKEY(FILE \*fp, EVP\_PKEY \*\*x, pem\_password\_cb \*cb, void \*u) – чтение публичного ключа в PEM кодировке из файла, где

fp – файловый дескриптор,

x – ключ,

cb – функция обратного вызова для запроса парольной фразы,

u – параметр для функции обратного вызова.

#### Заголовочный файл <openssl/ec.h>

EVP\_PKEY\_CTX\_set\_ec\_paramgen\_curve\_nid(EVP\_PKEY\_CTX \*ctx, int nid) – установка эллиптической кривой для контекста, где

ctx – контекст алгоритма,

nid – идентификатор кривой.

EC\_KEY\_check\_key(const EC\_KEY \*key) – валидация ключа алгоритма эллиптической кривой, где

key – ключ алгоритма эллиптической кривой.

ECDSA\_do\_sign(const unsigned char \*dgst, int dgst\_len, EC\_KEY \*eckey) – вычисление подписи по алгоритму ECDSA, где

dgst – хэш-сумма сообщения,

dgst\_len – длина хэш-суммы,

eckey – ключ алгоритма эллиптической кривой.

ECDSA\_do\_verify(const unsigned char \*dgst, int dgst\_len, const ECDSA\_SIG \*sig, EC\_KEY \*eckey) – проверка подписи по алгоритму ECDSA, где

dgst – хэш-сумма сообщения,

dgst\_len – длина хэш-суммы,

sig – подпись по алгоритму ECDSA,

eckey – ключ алгоритма эллиптической кривой.

i2d\_ECDSA\_SIG(const ECDSA\_SIG \*sig, unsigned char \*\*pp) – DER кодировка подписи по алгоритму ECDSA, где

sig – подпись по алгоритму ECDSA,

pp – буфер для записи DER-кодированной подписи.

d2i\_ECDSA\_SIG(ECDSA\_SIG \*\*sig, const unsigned char \*\*pp, long len) – декодирование DER-кодированной подписи по алгоритму ECDSA, где

sig – подпись по алгоритму ECDSA,

pp – подпись по алгоритму ECDSA в DER кодировке,

len – длина подписи в DER кодировке.

ECDSA\_SIG\_free(ECDSA\_SIG \*sig) – освобождение памяти подписи по алгоритму ECDSA, где

sig – подпись по алгоритму ECDSA.

EC\_KEY\_free(EC\_KEY \*key) – освобождение памяти ключа алгоритма эллиптической кривой, где

key – ключ алгоритма эллиптической кривой.

## Интерфейс библиотеки librhash

### Используемые подпрограммы

#### Заголовочный файл <rhash.h>

rhash\_library\_init() – инициализация библиотеки.

rhash\_msg(unsigned hash\_id, const void\* message, size\_t length, unsigned char\* result) – вычисление хэш-суммы сообщения, где

hash\_id – идентификатор алгоритма хэш,

message – сообщение,

length – длина сообщения,

result – хэш-сумма сообщения.

# Описание разработанных приложений

## Приложение генерации ключей (gen.c)

Приложение для генерации ключевой пары для алгоритмов на эллиптической кривой secp521r1. Сгенерированные ключи записываются в файлы в PEM кодировке, по путям, указанным в аргументах командной строки.

Блок-схема приложения приведена в приложении 1.

Аргументы командной строки:

1. Путь к файлу, в который будет записан приватный ключ;
2. (Опционально) Путь к файлу, в который будет записан публичный ключ.

При успешном завершении приложение не выводит информацию в стандартный поток вывода, возвращает значение 0. В указанных в аргументах командной строки файлах будут записаны необходимые ключи.

В случае ошибки на любом этапе работы, в стандартный поток вывода ошибок выводится примерное описание ошибки, приложение завершается с отрицательным кодом возврата. При этом по путям, указанным в аргументах командной строки, могут быть созданы файлы с неопределёнными данными.

## Приложение реализующее сторону A протокола аутентификации (a.c)

Приложение пытается подключиться по протоколу TCP по адресу, указанному в аргументах командной строки, к стороне B, и осуществляет взаимодействие по протоколу аутентификации.

Блок-схема приложения приведена в приложении 2.

Аргументы командной строки:

1. Путь к файлу с приватным ключом стороны A в PEM кодировке;
2. IP-адрес стороны B протокола аутентификации;
3. Порт стороны B протокола аутентификации.

При успешном завершении приложение не выводит информацию в стандартный поток вывода, возвращает значение 0.

В случае ошибки на любом этапе работы, в стандартный поток вывода ошибок выводится примерное описание ошибки, приложение завершается с отрицательным кодом возврата.

## Приложение реализующее сторону B протокола аутентификации (b.c)

Приложение ожидает подключения по протоколу TCP по адресу, указанному в аргументах командной строки, стороны A, и осуществляет взаимодействие по протоколу аутентификации.

Блок-схема приложения приведена в приложении 3.

Аргументы командной строки:

1. Путь к файлу с публичным ключом стороны A в PEM кодировке;
2. Порт ожидания подключения стороны A протокола аутентификации.
3. Максимальная разница в секундах между меткой времени в токене и временем получения токена стороной B, при которой аутентификация будет считаться успешно пройденной.
4. (Опционально) При наличии любого значения в этом параметре, приложении обработает протокол только один раз и завершиться. При отсутствии параметра, приложение работает до остановки пользователем, постоянно принимая новые подключения и отрабатывая протокол для каждого из них.

В процессе работы, для каждого подключения после завершения протокола аутентификации в стандартный поток вывода приложение выводит информацию об адресе стороны A, результате проверки подписи, разности времени между меткой времени в токене и временем приема сообщения, а также итоговое решение о прохождении аутентификации.

При успешном завершении приложение возвращает значение 0. Отрицательный результат аутентификации ошибкой не считается.

При наличии 4 параметра командной строки, при возникновении любой ошибки на любом этапе работы программы, в стандартный поток вывода ошибок выводится примерное описание ошибки (за исключением неверной подписи, в случае которой выводится описанная раньше информация в стандартный поток вывода), приложение завершается с отрицательным кодом возврата.

При отсутствии 4 параметра командной строки, в случае критической ошибки на любом этапе работы, в стандартный поток вывода ошибок выводится примерное описание ошибки, приложение завершается с отрицательным кодом возврата. В случае возникновения некритической ошибки (например, ошибка сетевого взаимодействия с конкретным клиентом, неверный идентификатор в токене, неверная подпись), в стандартный поток вывода ошибок выводится примерное описание ошибки (за исключением неверной подписи, в случае которой выводится описанная раньше информация в стандартный поток вывода), приложение ожидает подключение следующего клиента.

## Приложении проверки реализации протокола (main.c)

Вспомогательное приложение для тестирования. Приложении генерирует ключи использую приложение gen, запускает приложение b с указанием 4 аргумента командной строки, после чего запускает приложение a, указав информацию для подключения к b.

Приложение не принимает аргументов командной строки.

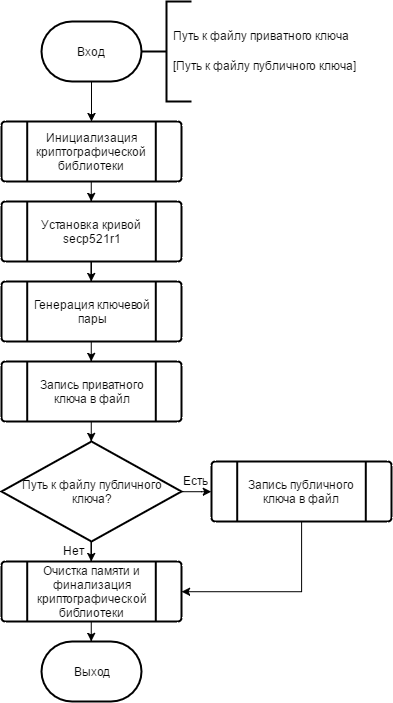
При успешном завершении приложений gen, a и b, приложение не выводит информацию в стандартный поток вывода, возвращает значение 0.

При любой ошибке на любом этапе работы, а также при возвращении запущенными приложениями отрицательного кода возврата, в стандартный поток вывода ошибок выводится примерное описание ошибки, приложение завершается с отрицательным кодом возврата.

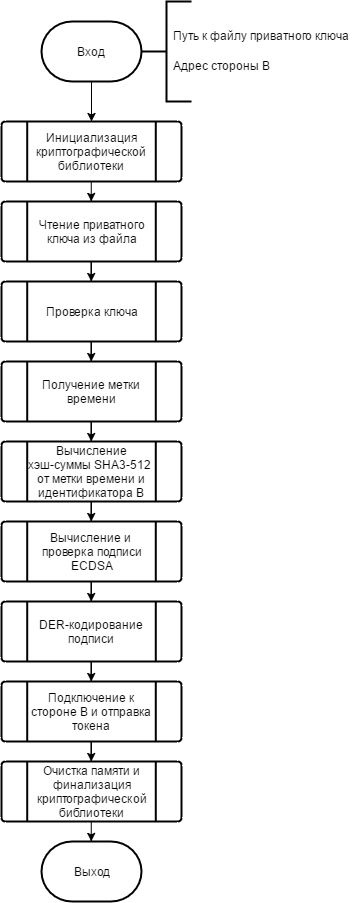
**Вывод**

В результате выполнения курсовой работы приобретены практические навыки реализации протокола аутентификации и работы с криптографическими библиотеками. Разработаны программы генерации ключей алгоритмов эллиптической кривой и реализующие стороны протокола однопроходной односторонней аутентификации.

**Приложение 1. Блок-схема приложения gen.c**



**Приложение 2. Блок-схема приложения a.c**



**Приложение 3. Блок-схема приложения b.c**

