Задание № 4 ч.2 по практикуму на ЭВМ.

Реализация ЭЦП ГОСТ Р 34.10-2012

Материалы

ГОСТ Р 34.10-2012	–	стандарт ЭЦП
ГОСТ Р 34.11-2012	–	стандарт хэш-функции
МР 26.2.002-2018	–	параметры эллиптических кривых и групп точек на них
`rand.c`	–	генератор последовательностей, см. Треб. к функциональности.

Описание

В данном задании требуется реализовать 4 программы для работы с ЭЦП.

Создание ключей

Данный раздел описывает программы для работы с ключами:

- `genpkey` – генерация приватного ключа и запись его в файл.
- `convpkey` – конвертирование приватного ключа в открытый.

Использование:

```
genpkey [-s] file
convpkey [-s] file [file2]
```

- `-s` – использовать набор параметров:
  `id-tc26-gost-3410-2012-256-paramSetA` (МР 26.2.002-2018 Б.1).
  По умолчанию используется набор:
  `id-tc26-gost-3410-2012-512-paramSetC` (МР 26.2.002-2018 Б.2).
- `file` – название файла (путь к файлу) с приватным ключ.
- `file2` – название файла (путь к файлу), в который будет записан открытй ключ. По умолчанию `file.pub`.

Формат выходных данных:

В файле приватного ключа находится число `d`, записанное от старшего байта к младшему. В файле открытого ключа записаны два числа подряд: `u` и `v` – координаты точки `Q` на скрученной кривой Эдвардса в аналогичном `d` формате.

В случае невозможности открыть файл, надо напечатать сообщение об ошибке в `stderr`.

Требования к функциональности:

- Реализация должна содержать функции или класс для генерации ключей.
- Функция генерации приватного ключа `d`:
  1. В качестве входных данных должна принимать набор параметров кривой (например, структуру со значениями).
  2. Генерация должна зависеть от функции получения (псевдо)случайных последовательностей. Например ее можно передать как параметр шаблона (в C++) или просто ее вызывать. Ее прототип:
     `int rand_bytes(uint8_t *buf, int num)`, где результат 1 обозначает успех, иначе 0.
- Функция получения публичного ключа по набору параметров и приватному ключу.

Далее представлен пример, демонстрирующий возможную реализацию требований (в данном случае это структура `struct paramset`, функции `gen_priv_key`, `get_pub_key`, `rand_bytes`) на языке C.

Имплементация `rand_bytes` идет вместе с заданием.

```
/* пример использования следующих объектов */
const struct paramset SetA;
void gen_priv_key(uint8_t *d,
                  const struct paramset *pset);
void get_pub_key(uint8_t *Q,
                 uint8_t *d,
                 const struct paramset *pset);
int rand_bytes(uint8_t *buf, int num);

/* буферы для результатов */
uint8_t d[Z_256_BIT_SIZE];
uint8_t Q[2][Z_256_BIT_SIZE];

/* получение приватного ключа */
gen_priv_key(d, &SetA, rand_bytes);
/* конвертирование приватного ключа в открытый */
get_pub_key(Q, d, &SetA);

/* буферы для генерации случайного числа
 * t[0] - содержит младшие разряды t */
uint8_t t[Z_256_BIT_SIZE];

/* алгоритм генерации приватного ключа */
do {
    rand_bytes(t, sizeof t);
} while (Z_256_iszero(t) or Z_256_l(t, pset->q));
/* функция iszero: проверка на 0, функция l(a,b) проверяет, что a < b. */
memcpy(d, t, sizeof d);
```

Создание и проверка ЭЦП

Раздел описывает программы для вычисления и проверки электронной цифровой подписи.

- `sign` – создание ЭЦП.
- `verify` – проверка ЭЦП.

Использование:

```
sign [-s] priv-key file [crt]
verify [-s] pub-key file [crt]
```

- `-s` – использовать набор параметров:
  `id-tc26-gost-3410-2012-256-paramSetA` (МР 26.2.002-2018 Б.1).
  По умолчанию используется набор:
  `id-tc26-gost-3410-2012-512-paramSetC` (МР 26.2.002-2018 Б.2).
- `priv-key` – название файла (путь к файлу) с приватным ключом.
- `pub-key` – название файла (путь к файлу) с открытым ключом.
- `file` – название файла (путь к файлу), для которого вычисляется ЭЦП.
- `crt` – название файла (путь к файлу) с подписью. По умолчанию `file.crt`.

Требования к функциональности:

- Реализация должна использовать контекст – структуру или класс для хранения текущего состояния вычисления или проверки подписи.
- Должны присутствовать следующие функции:
  1. инициализация структуры/класса контекста для последующей обработки данных, принимающая на вход набор параметров ЭЦП.
  2. функция обработки порции данных, передаваемых по указателю, с указанием размера данных.
  3. функция вычисления подписи по приватному ключу, которая завершает обработку данных, сохраняя подпись в переданный буфер.
  4. функция проверки подписи по открытому ключу, которая завершает обработку данных, возвращая реузльтат проверки.

Далее представлен пример, демонстрирующий возможную реализацию требований (в данном случае это структуры `struct context`, `struct paramset`, функции `rand_bytes`, `init`, `update`, `sign` и `verify`) на языке C.

```
/* пример использования следующих объектов */
struct context ctx_s, ctx_v;
const struct paramset SetA;
int rand_bytes(uint8_t *buf, int num);
void init(struct context *ctx,
          const struct paramset *pset);
void update(struct context *ctx,
            const uint8_t *data,
            size_t data_size);
void sign(struct context *ctx,
          const uint8_t *d,
          uint8_t *sig);
bool verify(struct context *ctx,
            const uint8_t *Q,
            const uint8_t *sig);

/* некоторая пара ключей Q=dP */
const uint8_t d[Z_256_BIT_SIZE];
const uint8_t Q[2][Z_256_BIT_SIZE];
/* буфер для получения подписи */
uint8_t signature[SIGN_256_BYTE_SIZE];

/* буфер для чтения данных, размер произвольный */
uint8_t data[DATA_BUF_SIZE];
size_t size;

/* открытие какого-то файла :) */
FILE *file = fopen(argv[0], "rb");

/* посчитаем и проверим подпись для данных из file */
init(&ctx_s, &SetA);
init(&ctx_v, &SetA);
while ((size = fread(data, 1, sizeod data, file)) {
    update(&ctx_s, data, size);
    update(&ctx_v, data, size);
}
/* получим подпись */
sign(&ctx_s, d, signature);
/* проверим подпись */
assert(verify(&ctx_v, Q, signature));
```

Можно реализовать такие `sign` и `verify`, которые позволят вызывать их на одном и том же контексте, т.е. получать и проверять различные подписи для любых `d` и `Q=tP` для некоторого `t`, т.е. принадлежащих кривой, параметры которой были переданы в `init`. Требования к `update` аналогичны требованиям, предъявляемым в задании 4.ч1.

Требования к коду:

1. Программа должна быть написана на языке C/C++.
2. Код не должен быть скопирован у другого студента.
3. Стремиться к тому, чтобы каждая функция в коде не превышала 25 строк.
4. Стремиться к тому, чтобы каждая строка в коде не превышала 80 символов.
5. Функции и переменные должны иметь осмысленные имена.
6. Компиляция производится gcc версии 4.9+ (лучше 8.1+), с флагами -Wall -O2.
7. Реализация должна быть кроссплатформенной.
8. В архиве с программой должен быть Makefile.

Формат приема заданий:

1. Задания отсылаются на почту is.cmc.2018@yandex.ru.
2. Тема письма в формате “Задание_|4.2|_Ф_И_О”.
3. В случае наличия замечаний/ошибок проверяющий отправляет комментарий. Процесс повторяется до тех пор, пока аспирант не сообщит, что замечаний больше нет.

Задать вопросы, получить актуальную и оперативную информацию можно в Telegram-чате: `https://t.me/joinchat/DpzKQxJXQ_xZAYlYyaYoPQ`.