Конвертация открытых ECDSA ключей и подписи из формата PKCS#11 в формат OpenSSL

PKCS#11 и OpenSSL имеют разный формат представления объектов. Так, они по-разному представляют ECDSA ключи и подписи на них. Поэтому, когда мы хотим проверить в OpenSSL сырую ECDSA подпись полученную через PKCS#11, ее необходимо сконвертировать в OpenSSL формат. Для проверки подписи нужно также сконвертировать и открытый ключ.

Ручной конвертации можно избежать воспользовавшись OpenSSL движком rtengine. Он предназначен для работы с объектами на токене средствами OpenSSL. Но не во всех задачах использование OpenSSL приемлемо, поэтому мы написали статью, как конвертировать объекты вручную.

Конвертация открытого ключа

Описание формата открытого ключа в PKCS#11

Библиотека PKCS#11 возвращает открытый ключ ECDSA через атрибут CKA_EC_POINT. Значение атрибута – это число представленное в виде TLV структуры – OCTET_STRING:

1. T – тег типа размером один байт. В нашем случае это 04 – OCTET_STRING.
2. L – длина значения V. Она может быть представлена в виде нескольких байт. Если длина не превышает 7F байт, то L состоит из одного байта. Если больше 7F байт, то старший бит устанавливается в 0, а младшие 7 бит определяют длину L. Например, L=7A – длина 0x7A (122). L=81 80 – длина 0x80(128). L=82 7F FF – длина 0x7FFF(32767).
3. V – значение размером L байт.

Пример CKA_EC_POINT

Снизу представлен пример TLV структуры открытого ключа. Первый байт 04 – это тег OCTET_STRING. Второй – длина открытого ключа (64 байта). Последующие 64 байта – значение ключа.

```
04 41 04 8B 92 09 CC C0 A1 B4 19 BA 80 2E 44 5D A2 16 E6 92 AA 9C BB B9 CC B0 CE 5C 76 71 C6 DF E4 CA 83 46 BB 92
2B C8 6F FC 15 20 D8 11 5F 32 4F 7F CA BB DE 9F E3 62 5E 8E 2C D2 2D C2 51 EC 3B 8C 67
```

Описание формата открытого ключа в OpenSSL

OpenSSL позволяет представлять открытый ECDSA ключ в виде ASN.1 структуры. Эта структура хранит не только значение открытого ключа, но и его параметры.

Пример открытого ключа в формате OpenSSL:

ASN.1 структуры открытого ECDSA ключа

```
30 56 -- SEQUENCE.   --   (30),  --   (56).
        30 10 -- SEQUENCE
                06 07 2A 86  48 CE 3D 02 01 -- OBJECT IDENTIFIER.   .     ecPublicKey (  . ).
                06 05 2B 81 04 00 0A -- OBJECT IDENTIFIER.   .     secp256k1 ( . )

        03 42 00 04 8B 92 09 CC C0 A1 B4 19 BA 80 2E 44 5D A2 16 E6 92 AA 9C BB B9 CC B0 CE 5C 76 71 C6 DF E4
CA 83 46 BB 92 2B C8 6F FC 15 20 D8 11 5F 32 4F 7F CA BB DE 9F E3 62 5E 8E 2C D2 2D C2 51 EC 3B 8C 67 -- BIT
STRING.
```

Описание процесса конвертации

Для того, чтобы сконцентрировать ECDSA ключ из PKCS#11 формата, нужно:

1. Создать SEQUENCE со значениями типа OBJECT IDENTIFIER. Первый элемент – OID типа открытого ключа на эллиптической кривой. Второй – OID параметра эл. кривой.
2. Сконвертировать открытый ключ из структуры OCTET_STRING в структуру BIT_STRING.
3. Положить результаты в SEQUENCE. Нужно проставить правильную длину выходной последовательности.

Конвертация из OCTET_STRING происходит следующим образом

1. Тег 04 заменяется на 03 (тег BIT STRING).
2. После байтов длины (L) ставится байт 00.
3. L увеличивается на 1.

Пример конвертации

Для открытого ключа сверху

1. На первом этапе сформируем последовательность из идентификаторов типа открытого ключа и параметра эл. кривой:
   30 10 06 07 2A 86 48 CE 3D 02 01 06 05 2B 81 04 00 0A
   a. 30 - тег
   b. 10 - размер
   c. 06 - тег
   d. 07 - размер
   e. 2A 86 48 CE 3D 02 01 - id-ecPublicKey
   f. 06 - тег
   g. 05 - размер
   h. 2B 81 04 00 0A - secp256k1
2. Сконвертируем ключ из OCTET_STRING в BIT_STRING :
   03 42 00 04 8B 92 09 ...
   a. 03 – тег
   b. 42 – размер
   c. 00 – заполнитель
   d. 04 8B 92 09... – 0x41 байт открытого ключа
3. Сложим результаты в SEQUENCE:
   30 56 30 10 06 07 2A 86 ...
   a. 30 – тег
   b. 56 – размер (0x2+0x2+0x42+0x10)

Результат:

```
30 56 30 10 06 07 2A 86 48 CE 3D 02 01 06 05 2B 81 04 00 0A 03 42 00 04 8B 92 09 CC C0 A1 B4 19 BA 80 2E 44 5D A2
16 E6 92 AA 9C BB B9 CC B0 CE 5C 76 71 C6 DF E4 CA 83 46 BB 92 2B C8 6F FC 15 20 D8 11 5F 32 4F 7F CA BB DE 9F E3
62 5E 8E 2C D2 2D C2 51 EC 3B 8C 67
```

DER и PEM форматы

Если записать в файл байты в том виде, в котором мы их получили, мы получим открытый ключ в DER формате. Если мы хотим представить его в виде печатных символов, его нужно перевести в PEM формат. Это можно сделать с помощью команды:

Конвертация в PEM

```
openssl ec -pubin -inform DER -in pubkey.der -outform PEM -out pubkey.pem
```

Проверка открытого ключа

Чтобы проверить, что ключ сконвертирован верно, выполните команду:

Проверка ключа

```
openssl ec -check -pubin -in pubkey.pem
```

Конвертация подписи

Представление подписи в PKCS#11

Для подписи данных библиотека PKCS#11 использует функцию C_Sign. Результат возвращается в виде R|S структуры. Формат у этой структуры простой:

1. 32 байта числа R
2. 32 байта числа S

Пример R|S подписи

Пример R|S

```
43 1C 62 38 11 04 91 43 76 28 A5 D8 9E 5F EE 90 B0 DC 68 39 D2 B8 11 F3 22 2C D4 DB E2 05 49 BF -- R
8A EF 68 77 8B 0D B8 E6 11 FD 55 56 37 71 62 7E B7 31 22 67 8D 61 7F B6 41 EA 7E 1F 84 C7 36 41 -- S
```

Описание формата подписи в OpenSSL

OpenSSL представляет подпись в виде ASN.1 структуры.

ASN.1 структуры открытого ECDSA ключа

```
30 45 -- SEQUENCE
        02 20 43 1C 62 38 11 04 91 43 76 28 A5 D8 9E 5F EE 90 B0 DC 68 39 D2 B8 11 F3 22 2C D4 DB E2 05 49
BF     -- R
    02 21 00 8A EF 68 77 8B 0D B8 E6 11 FD 55 56 37 71 62 7E B7 31 22 67 8D 61 7F B6 41 EA 7E 1F 84 C7 36 41
-- S
```

Описание процесса конвертации

1. Перевести R и S в TLV тип INTEGER
2. Результаты положить в SEQUENCE. Нужно проставить правильную длину выходной последовательности.

Числа R и S приводятся к TLV. типу INTEGER с помощью такого алгоритма:

1. T = 02.
2. L = кол-во байт для хранения числа. Если первый байт числа превышает 7F, то длина увеличивается на 1 и после L записывается байт 00.
3. V = значение числа.

Пример конвертации

1. Число R преобразуется к типу INTEGER:
   02 20 43 1C 62 38 11 04 ...
   a. 02 - тег
   b. 20 - длина
   c. 43 1C 62 38 11 04 ... - само число
2. Число S преобразуется к типу INTEGER:
   02 21 00 8A EF 68 77 8B
   a. 02 - тег
   b. 21 - длина. На единицу больше т.к. старший байт числа 0x8A больше 0x7F
   c. 00 - заполнитель
   d. 8A EF 68 77 8B ... - само число
3. Запишем результат в SEQUENCE:
   30 45 02 20 43 1C
   a. 30 - тег
   b. 45 - длина последоватльности (0x02 + 0x02 + 0x20 +0x21)
   c. 02 20 43 1C ... - значение последовательности (R|S)

Проверка подписи

Проверка подписи можно с помощью команды openssl

проверка подписи
`openssl dgst -sha256 -verify pubkey.pem -signature sign.der data.bin`