Activating GemPC USB card reader using myEID smartcard with FreeBSD to use with OpenDNSSEC

On FreeBSD, install packages:

```
    devel/libccid
      devel/pcsc-lite
    security/pcsc-tools
    security/opensc
```

You will have to fix

/usr/local/share/opensc/myeid.profile

as decribed here:

https://github.com/OpenSC/OpenSC/wiki/Aventra-MyEID-PKI-card

```
204c204
<                       file-id   = 4501;
---
>                       file-id   = 4601;
```

You will then need to restart pcscd (service pcscd restart)

Plug the reader, and insert a card

```
# opensc-tool --list-readers
```

should list the card reader, asumming that pcscd started and libccid is installed:

```
# Detected readers (pcsc)
Nr.  Card  Features  Name
0    Yes             Gemalto GemPC Twin 00 00
```

Run this to initialize the smartcard:

```
pkcs15-init -C --so-pin 1111 --so-puk 1111 --pin 1111 --puk 1111
```

If everything goes well, the card should be ready for initialization.

Initiatilization:

```
pkcs15-init -C --so-pin 1111 --so-puk 1111 --pin 1111 --puk 1111

pkcs15-init -P -a -1 -l "Basic PIN"  --pin nsec3 --puk nsec4

 User PIN : nsec3
 User PUK : nsec4

pkcs11-tool --module /usr/lib/opensc-pkcs11.so -L
```

Once initialized, the output will look like:

```
Available slots:
Slot 0 (0xffffffffffffffff): Virtual hotplug slot
  (empty)
```

```
Slot 1 (0x1): Gemalto GemPC Twin 00 00
  token label:   MyEID (Basic PIN)
  token manuf:   Aventra Ltd.
  token model:   PKCS#15
  token flags:   rng, login required, PIN initialized, token initialized
  serial num  :  0093019074952092
```

Note the "token label:" field above. It will be used in the Repository definition in the OpenDNSSEC conf.xml

Now install the opendnssec tools to test the access to the HSM

```
    aptitude install libhsm-bin
```

Edit /etc/opendnssec/conf.xml to define the smartcard as a repository

```
        <Repository name="token">
            <Module>/usr/local/lib/opensc-pkcs11.so</Module>
            <!-- TokenLabel must match what's reported by pkcs11-tool -->
            <TokenLabel>MyEID (Basic PIN)</TokenLabel>
            <!-- User PIN when initialized -->
            <PIN>nsec3</PIN>
        </Repository>
```

Test access to the smartcard

```
    ods-hsmutil list token
    Listing keys in repository: token
    0 keys found.
```

Test generation of a key

```
    ods-hsmutil generate token rsa 1024
    Generating 1024 bit RSA key in repository: token
    Key generation successful: d15e0018de6c0d17c71b41e746498d73
```

The smartcard is ready to be used with OpenDNSSEC

Let's assume you want to keep the KSK in the smartcard, and the ZSK on a different HSM (will use softHSM for the example).

Setting up the softHSM:

```
apt-get install softhsm

softhsm --init-token --slot 0 --label "ZSK repo"
SO PIN: SO_must_prevail
User PIN: 1234
```

and then add this softHSM repository to the configuration.

In conf.xml:

```
        <Repository name="SoftHSM">
            <Module>/usr/local/lib/softhsm/libsofthsm.so</Module>
            <TokenLabel>ZSK repo</TokenLabel>
            <PIN>1234</PIN>
            <SkipPublicKey/>
        </Repository>
```

The relevant section to in kasp.xml is (note the Repository names):

```
        <KSK>
            <Algorithm length="2048">8</Algorithm>
            <Lifetime>P1Y</Lifetime>
            <Repository>token</Repository>
        </KSK>

        <!-- Parameters for ZSK only -->
        <ZSK>
            <Algorithm length="1024">8</Algorithm>
            <Lifetime>P30D</Lifetime>
            <Repository>ZSK repo</Repository>
        </ZSK>
```