Information and Organisational Security

Guides for Practical Classes

João Paulo Barraca and Vitor Cunha

Department of Electronics, Telecommunications and Informatics
University of Aveiro

2019–2020

5

Digital Signatures with the PTEID

Topics:

- Digital signatures.
- PKCS#11 devices.
- Producing digital signatures with the Portuguese Citizen Card.

5.1 Introduction

In this guide you will develop Python programs using the Python PKCS11 module[1] that interacts with PKCS#11 tokens to produce and validate digital signatures.

For that, you need to have installed the following software:

- pcscd: `apt install pcscd`
- Python3 PIP: `apt install python3-pip`
- PyKCS11: `pip3 install --user pykcs11`
- PTEID Software: available at `https://www.autenticacao.gov.pt/cc-aplicacao` and pre-installed in the Virtual Machine Image.
- (Optional) Opensc: `apt install opensc-pkcs11`

[1] `https://github.com/danni/python-pkcs11`

You can verify the correct operation of the software by executing `eidguiV2` and `pkcs11-tool -L`.

5.2 PKCS#11 Standard

The PKCS#11 standard (Cryptographic Token Interface Standard) is produced by RSA Security and defines native programming interfaces to cryptographic tokens, such as hardware cryptographic accelerators and smartcards, as is the case of Portuguese Citizen Card.

Python PyKCS11 includes a provider that, in contrast to most other providers, does not implement cryptographic algorithms itself. Instead, it exposes functions that are executed in the hardware tokens, and allows to obtain the objects stored in a smart card. When using the Python Cryptography module, this is the role of the Backends (providing access to hardware tokens). Unfortunately, no public (and stable) backend supports PKCS#11, making it required to use a standalone module (PyKCS#11).

5.2.1 Loading the card interface module

Interating with the Portuguese Citizen Card requires the use of a module that translated PKCS#11 calls into some internal format adequate to the smart card hardware. This is independent of the programming language used, and it is why we need to install the Portuguese Citizen Card software: besides a visualization tool, it also contains libraries that allow accessing the Portuguese Citizen Card (`libpteidpkcs11.so`)

Using the PyKCS11 module, the following snippet will try to list all slots, and present information about the tokens contained:

```python
import PyKCS11
import binascii

lib = '/usr/local/lib/libpteidpkcs11.so'
pkcs11 = PyKCS11.PyKCS11Lib()
pkcs11.load(lib)
slots = pkcs11.getSlotList()

for slot in slots:
        print(pkcs11.getTokenInfo(slot))
```

If you obtain an error with this version, please use the library available at

(`/usr/lib/x86_64-linux-gnu/opensc-pkcs11.so`) or, in alternative, the SDK v1.6 available in the *Elearning* web page.

Exercise: List all tokens and print the detais of each token (revision, manufacturer, model, serial, etc...)

5.3 Contents of the Portuguese Citizen Card

After the token is available, it is possible to list the objects it contains. Some objects have public visibility and are accessed through a public session, while others are private, and the user must explicitelly create a private session. The different between sessions lies in the existence of omission of a pin code when opening the session.

You can list the contents of the Portuguese Citizen Card (i.e., its PKCS#11 objects), with code based on the following snippet:

```python
all_attr = list(PyKCS11.CKA.keys())
#Filter attributes
all_attr = [e for e in all_attr if isinstance(e, int)]

session = pkcs11.openSession(slot)
for obj in session.findObjects():
    # Get object attributes
    attr = session.getAttributeValue(obj, all_attr)
    # Create dictionary with attributes
    attr = dict(zip(map(PyKCS11.CKA.get, all_attr), attr))

    print('Label: ', attr['CKA_LABEL'])
```

The information you get in the `CKA_LABEL` attribute is the identification of the object included in the Portuguese Citizen Card. Through these identifiers we can select which certificate, or private key, we intend to use for each cryptographic operation. The attribute `CKA_CLASS` identifies the class of the object and the attribute `CKA_CERTIFICATE_TYPE` can identify the type of a certificate.

You can individually inspect all attributes of the objects. If you require access to the certificate content, convert the tuple contained in the `CKA_VALUE` to bytes (`bytes(attributes['CKA_VALUE'])`) and load it as a DER certificate (`x509.load_der_x509_certificate`).

Exercise: Print the label of all objects, as well as the issuer and subject of all certificates.

Hint: Once loaded as a X509 certificate, you can reuse the code from the last class to print the issuer and subject.

5.4 Digital signature

Create a program capable of generating a digital signature of a document, using the Portuguese Citizen Card, and storing it on a given file. Furthermore, complement it for writing on a file the public key certificate corresponding to the private key used for signing the document.

For this purpose consider the `CITIZEN AUTHENTICATION CERTIFICATE` and the corresponding `CITIZEN AUTHENTICATION KEY` (the private key), as many citizens to not have the `CITIZEN SIGNATURE CERTIFICATE` activated.

Because the private key is not extractable, you cannot load this key as a RSA key. Instead, use the `session` object to sign the text:

```
private_key = session.findObjects([
                  (PyKCS11.CKA_CLASS, PyKCS11.CKO_PRIVATE_KEY),
                  (PyKCS11.CKA_LABEL, 'CITIZEN AUTHENTICATION KEY')
                  ])[0]

mechanism = PyKCS11.Mechanism(PyKCS11.CKM_SHA1_RSA_PKCS, None)

text = b'text to sign'
signature = bytes(session.sign(private_key, text, mechanism))
```

5.5 Signature validation

Create a program capable of checking a digital signature on a document. The program should use as inputs a file with the digital signature, a file with the signer's public key certificate and a file with the signed contents. You should also display the identity of the entity that produced the signature (the subject). This is important as the user should know the identity of the user that created the signature.

Hint: Validating the key doesn't use the Portuguese Citizen Card and reuses the validation processes from the last laboratory guide. To verify the signature, use `padding.PKCS1v15()` as padding and `hashes.SHA1()` as the hash mechanism. Do not forget to validate the certification chain, dates and purposes.

5.6 Bibliography

`https://cryptography.io/en/latest/`

`https://github.com/LudovicRousseau/PyKCS11`

`https://pkcs11wrap.sourceforge.io/api/`