Security in Informatics and in the Organizations (2018/2019)

Practical Class (#7):

SmartCards

Vitor Cunha (vitorcunha@ua.pt)

Recap – (Asymmetric Cryptography) Practical Use

- Confidentiality (eg. File Encryption)
  - Encrypt data with ***public key***
  - Decrypt with ***private key***
  - Source is not authenticated!

- Authenticity (eg. Digital Signatures, Authentication)
  - Encrypt challenge/identifier with ***private key***
  - Decrypt with ***public key***
  - Source is now authenticated!

*How do we keep the **private key** secure?*

*How to <u>stop</u> someone from **sharing** the **private key**?*

SmartCards: Components

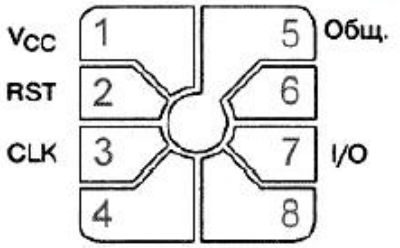
CPU
- 8/16 bit
- Crypto-coprocessor (opt.)

ROM
- Operation System
- Communication
- Cryptographic algorithms

EEPROM
- File system
  - Programs / applications
  - Keys / Passwords

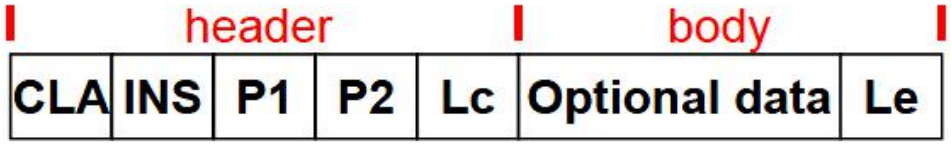
RAM
- Temporary data
  - Lost when card is disconnected

Mechanical Contacts
- ISO 7816-2
  - Power
  - Soft reset
  - Clock
  - Half duplex I/O



Physical Security
- Resistant to direct physical attacks
- Resistant to side channel attacks

Interacting with the SmartCard: APDU (ISO 7816-4)



Command APDU

- CLA (1 byte)
  - Instruction Class
- INS (1 byte)
  - Command
- P1 e P2 (2 bytes)
  - Command specific parameters
- Lc
  - Length of the optional data
- Le
  - Length of the data contained in the response
  - Zero (0) means all data available

Response APDU

- SW1 e SW2 (2 bytes)
  - State byte
  - 0x9000 means SUCCESS

Normalized communication with SmartCards (Physical Interface)

SmartCard Cryptographic Services Middleware

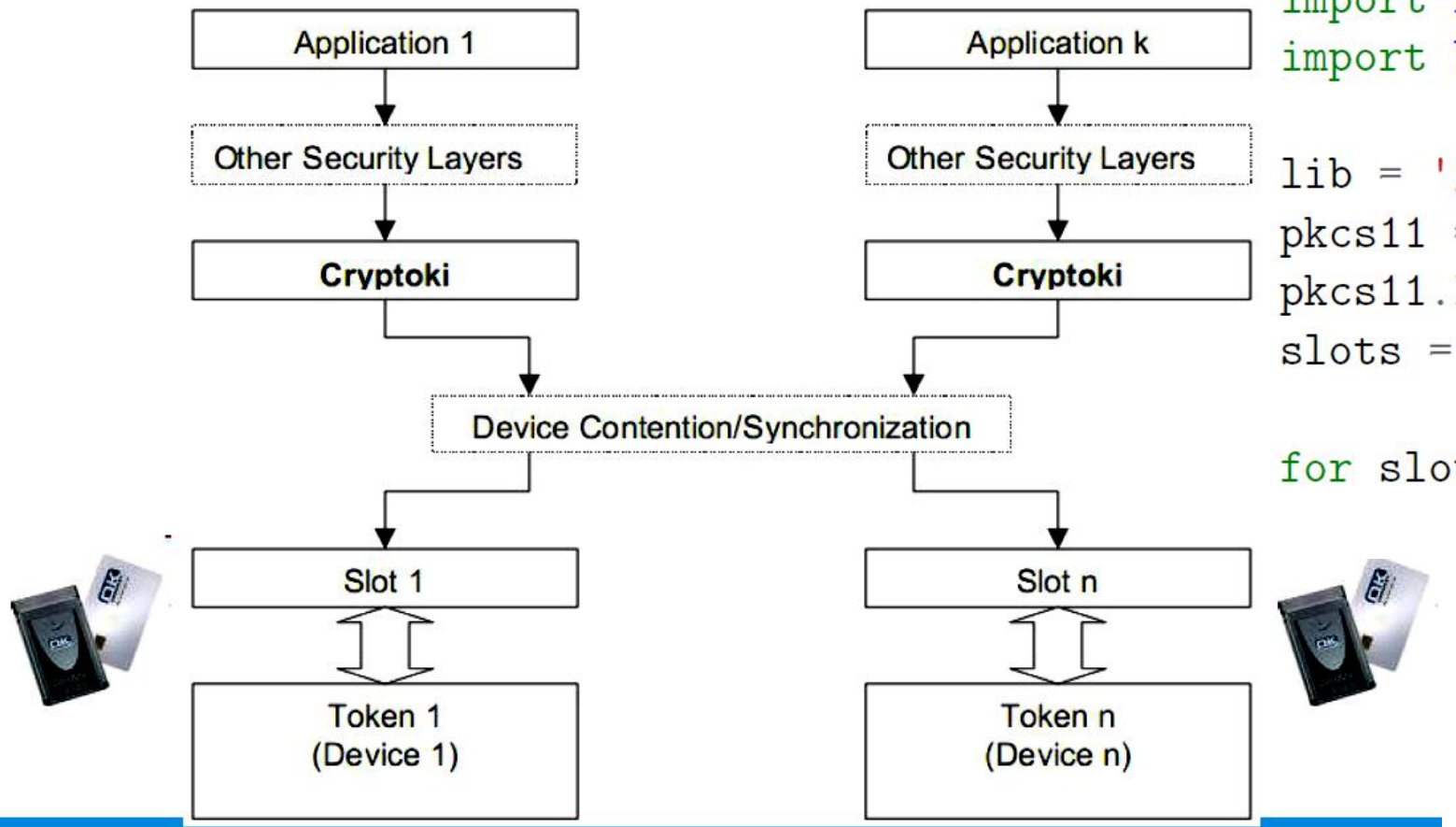
Libraries acting as bridges between the SmartCard and Higher Layer Applications

Based on standardized solutions:

- PKCS #11: Token Access Primitives
  - Cryptographic Token Interface Standard (cryptoki)
  - Defined by RSA Security Inc.
- PKCS #15: Information Structure In the Token
  - Cryptographic Token Information Format Standard
  - Defined by RSA Security Inc.
- CAPI CSP: API
  - CryptoAPI Cryptographic Service Provider
  - Defined by Microsoft for Windows applications
- PC/SC: API
  - Personal computer/Smart Card
  - Platform for access in Windows and Linux

Normalized communication with SmartCard (Applications)

PKCS #11: Interaction with applications




```
import PyKCS11
import binascii

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

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

!! Some operations require PIN !!

```
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))
```

class **pkcs11.Token**

A PKCS#11 token.

A token can be physically installed in a `Slot`, or a software token, depending on your PKCS#11 library.

Example:

*with token.open(**user_pin**='1234') as session:*

open(*rw=False, user_pin=None, so_pin=None*)

Open a session on the token and optionally log in as a user or security officer (pass one of *user_pin* or *so_pin*).

Can be used as a context manager or close with `Session.close()`.

```
with token.open() as session:
    print(session)
```

Parameters:

- **rw** – True to create a read/write session.
- **user_pin** (*bytes*) – Authenticate to this session as a user.
- **so_pin** (*bytes*) – Authenticate to this session as a security officer.

Return type: Session

You **CANNOT** recover/unlock your card with the PUK

(without going to “Loja do Cidadão”)