DI-FCT-UNL
Segurança de Redes e Sistemas de Computadores
Network and Computer Systems Security

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
1º Sem., 2020/2021

Key Management and Cryptographic Devices

Use of Public Key Crypto requires Secure and Trusted Key-Management

- **Generation control of keypairs**
- **Careful confinement, management and use (processing) in secure environments**
  - **Management of Private Keys** (in private-key rings)
  - **Public keys**: can be distributed, disseminated and publicly disclosed
    - Management as "public-key rings"
    - Trusted association to the correct UIDs of principals
    - Validation requires a trusted verification of such associations, as "verifiable" and "certified" associations
- Another issue: management of keys and certificates require the use of **standard and interoperable representation formats**
  - **Private and public keys or related parameters**
  - **Public key certificates / trusted management of public keys**

Protection of Private Keys

- **Private Keys**: must be protected from exposition risks, avoiding:
  - **Storage exposition**
    - Use of secure storage (encrypted)
      - Encrypted in disks or other storage devices
      - But where are the protection encryption keys ?
      - What if Protection Keys are "lost" ? Recovery-Mechanism
    - Ex., Keystores, protected by PBE and/or Symmetric Encryption
  - **Memory exposition** (when transferred to, managed and processed in memory) must be in memory w/ minimal exposure - only when required !
    - **Better:** stored and processed in locked "devices" or "appliances" where it may be impossible (or unlikely) the access by no-authorized parties (w/ cryptographic operations possibly performed in those devices)
      - Never exposed outside these devices !
      - Require crypto operations supported and processed "inside"
      - Access-control via authentication and cryptographic APIs

Management of Key Rings by Principals

Usually in Files

Trusted Associations
<subjectIDs, PublicKeys>

- As files, different formats
- As public keystores managing <$subjectID_i$, $PublicKey_i$> associations
  Ex: java keystores, PEM files, etc
- As trusted stores containing public key certificate stores and formats (ex., X509v3, PEM, DER. PKCS#12, etc.)

Usually in Protected (encrypted) Files

- As protected files w/ different formats
- As private keystores
  - java keystores wi/ different representations, ex: PEM, DER, PKCS#8

Management of Key Rings by Principals

Usually in Files

Trusted Associations
<subjectIDs, PublicKeys>

- As files, different formats
- As public keystores managing <$subjectID_i$, $PublicKey_i$> associations
  Ex: java keystores, PEM files, etc
- As trusted stores containing public key certificate stores and formats (ex., X509v3, PEM, DER. PKCS#12, etc.)

Usually in Protected (encrypted) Files

Master Keys
- Generated from secret seeds or passphrases
- Symmetric Encryption
- PWD-based Encryption

- As protected files w/ different formats
- As private keystores
  - java keystores wi/ different representations, ex: PEM, DER, PKCS#8

HSMs (Hardware Security Modules): Ex. of manufacturers, IBM, Safenet, nShield, …

https://www.ibm.com/security/cryptocards/hsms

IBM Crypto Express Modules

IBM PCIe
Crypto Coprocessor

Safenet
LAN-Based HSMs

Safenet
USB-Based HSM

Safenet
PCIs-Based HSM

Safenet
HSM Backup Appliance

https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/

HSMs (Hardware Security Modules): Ex., nShield, …

Ex.
nShield Connect
(Net Appliance)

Ex.
nShield Solo
(PCIe enabled)

Ex.
nShield Edge
(USB enabled)

https://www.ncipher.com/products/general-purpose-hsms

HSM Typical Features

- High performance cryptographic operations
- Compliance:
  - Security: FIPS 140.2 Levels 2 and 3, USGv6, Com. Criteria EAL4
  - Ex., Safety and environmental standards
- Supported cryptographic APIs (CAPIs): (the external surface)
  - PKCS#11
  - OpenSSL
  - Java JCE
  - Microsoft CAPI
  - CNG API
- OS and Virtualization compliance
- Reliability MTBF Metrics (~100000 hours)
- Security/Robustness:
  - Products w/ broad acceptance and evaluation
  - But …. `https://cryptosense.com/blog/how-ledger-hacked-an-hsm`

HSMs can improve considerably the performance of cryptographic operations

Ex., Compare w/ openssl performance in your computer ;-): openssl speed rsa ecc

nShield Connect Models	500+	XC Base	1500+	6000+	XC Mid	XC High
RSA Signing Performance (tps) for NIST Recommended Key Lengths						
2048 bit	150	430	450	3000	3500	8600
4096 bit	80	100	190	500	850	2025
ECC Prime Curve Signing Performance (tps) for NIST Recommended Key Lengths						
256 bit	540	680	1260	2400	5500	14,400

Devices for personal use

https://www.yubico.com/products/yubihsm/

Ex., YubyKey Series
USB-A, USB-C
Lightening, NFC

https://www.schneier.com/blog/archives/2019/07/yubico_security.html

Smartcards, Smartcard Readers

Cardomatic Smartcard-HSM
USB Stick

https://www.cardomatic.de/SmartCard-HSM-USB-Stick/

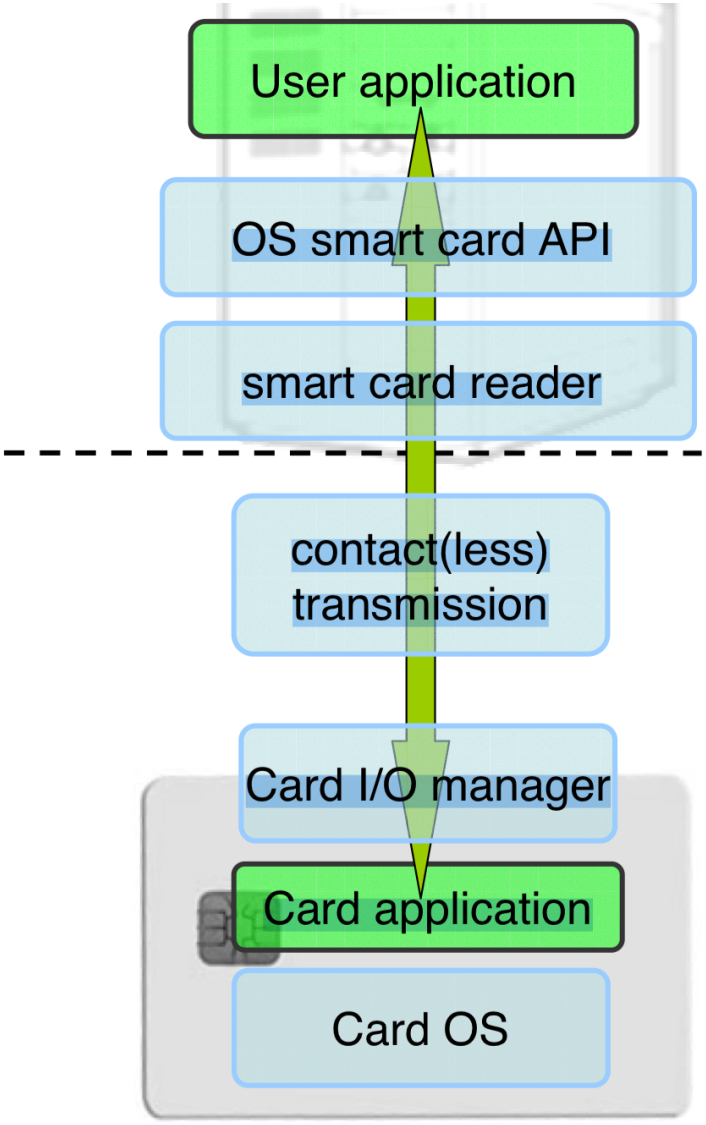
USB

USB +
Local Auth.
And Access Control
Pin/Pwd

USB +
Local Auth
and Access Control
Biometry

Main (some) Smart Card Standards …

- ISO 7816
  - Card physical properties
  - Physical layer communication protocol
  - Packet format (APDU)
- ☐ PC/SC, PKCS#11 https://en.wikipedia.org/wiki/PK
  - Standardized interface on host side
  - Card can be proprietary
- ☐ MultOS
  - Multi-languages programming, native compilation
  - High security certifications, often bank cards
- ☐ Java Card
  - Open programming platform from Sun
  - Applets portable between cards
- ☐ Microsoft .NET for smartcards
  - Similar to Java Card, relatively new
  - Applications portable between cards
- ☐ GlobalPlatform
  - Remote card management interface
  - Secure installation of applications



Interaction w/ Smartcards and other cryptographic devices

- Interface (via reader) by sending commands / receiving results: **APDUs or App. Protocol Data Units**)
  - APDUs are standardized messages (msg in / msg out)
- Note: APDUs are standardized structures but the content may be different depending on specific implementations
  - Many Smartcard manufacturers, variety of implementations and programming support
  - Applications (and programmers) don't use directly (in general) APDUs (considered a low level abstraction)
- Use of **more high-level abstractions or programming interfaces**
  - **Crypto APIs**
  - Provide standard generic primitives allowing the manipulation of objects in the smartcard, cryptographic and key-management operations
  - Examples:
    - **PKCS#11 (Crypto API defined by the RSA Labs)**
    - **Microsoft CryptoAPI (Cryptographic Application Programming Interface)**

PKCS#11 (aka, Cryptoki)

- Cryptoki: Cryptographic Token Interface
  - Provides an "uniform logic view" of a physical device (such as a smartcard) regarded as a "cryptographic token"
  - Implements an Object-Oriented Interface, through Middleware (libraries) provided by manufacturers
    - Also the case of the Portuguese Citizen Card and compatible Readers
    - In general a PKCS#11 middleware can be adopted by generic applications designed to support smartcards
      - Ex., Email User Agents, Browsers, etc.
      - Ex., Firefox (see Privacy and Security)

See https://en.wikipedia.org/wiki/PKCS_11 for more details

PKCS#11 in Java

- There is a Sun PKCS#11 Provider for Java JCA/JCE: can be used since the Java 5 (J2SE 5.0)
- In contrast to most other providers, it does not implement cryptographic algorithms itself.
  - It acts as a bridge between the Java JCA and JCE APIs and the native PKCS#11 cryptographic API, translating the calls and conventions between the two.
- This means that Java applications calling standard JCA/JCE APIs **can, without modification, take advantage of algorithms offered by underlying PKCS#11 implementations, such as, for example:**
  - Cryptographic Smartcards,
  - HSMs or Hardware cryptographic accelerators
  - High performance software implementations.

PKCS#11 in Java

- A Java PKCS#11 Crypto Provider can be installed or used as any other crypto provider: use the device as a "crypto-provider"

```
…
# configuration for security providers 1-9 omitted
security.provider10=sun.security.pkcs11.SunPKCS11 /opt/bar/cfg/pkcs11.cfg
```

See more in:
https://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html

Microsoft CryptoAPI (aka CAPI)

- High-Level Middleware Integration, including Smartcard interoperability for Microsoft Windows OS

- Architecture based on a generic module (providing an external API) and specific CSP (*Cryptographic Service Providers*), each one provided for specific physical devices
  - One CSP can or cannot use the PKCS#11 definition for specific smartcards: CSP as a "external API"

See more in:

See https://en.wikipedia.org/wiki/Microsoft_CryptoAPI for details

Microsoft CryptoAPI System Architecture CryptoNG API (aka CNG) and CAPICOM

https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography--cryptoapi--and-c