SMART KARTY V LINUXU

A PROČ BY VÁS MĚLY ZAJÍMAT

JAKUB JELEN, RED HAT

AGENDA

- Co je to a proč?
- Anatomie Smart karet a software využívající je
- Projekt OpenSC
- Praktické ukázky

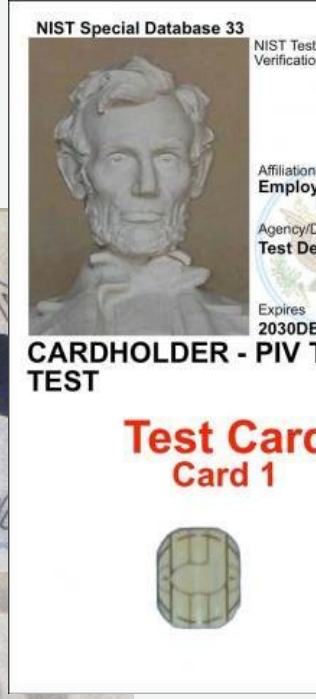
CO JE TO A PROČ?

SMART KARTA

- Plastová karta
- Standardní zlatý konektor
- Integrovaný obvod
  - Bezpečná paměť: citlivá data
    - Privátní klíče
    - Certifikáty
    - Datové položky
  - Čip:
    - kryptografické operace
- Kontrola PINu před kritickými operacemi
  - Čtení citlivých informací
  - Kryptografická operace

TYPY KARET

- SIM karty
- Kreditní a debetní čipové karty
- Občanky
  - Estonsko
  - Belgie
  - Španělsko
- Identifikační karty
  - vláda (PIV)
  - armáda US (CAC)



TYPY KARET

- USB tokeny
  - Yubikey
  - Nitrokey
  - GoldKey
  - ...
- Kombinace s jinou funkcionalitou
  - U2F, OTP, OAUTH, PGP
  - Šifrovaná USB flash pamět

ŽIVOTNÍ CYKLUS KARET

- Prvotní konfigurace
  - Generování klíčů, nahrávání certifikátů, nastavení PINu
- **Běžné použití**
  - Použití klíčů, certifikátů, změna PINu
- Administrativní úkony
  - resetování zapomenutého PINu, mazání obsahu karty

VYUŽITÍ V LINUXU

- Nahrazení hesel, nebo privátních klíčů/certifikátů uložených jinde
  - Autentizace k webovým službám z prohlížeče (e-government, banky)
  - Přihlášení k lokálnímu počítači
  - Odemčení šifrovaného disku (LUKS, dm-crypt)
  - Autentizace k SSH serveru
  - SSO/pkinit - mechanismus s získání kerberos ticketu
- Podpisy/šifrování emailů

ANATOMIE

Vrstva	Specifikace	Nástroj/knihovna v Linuxu
Aplikace		Firefox, OpenSSH, ...
Knihovny		NSS, engine_pkcs11+OpenSSL
	PKCS#11, PKCS#15, PIV, GSC-IS	OpenSC
	PC/SC	pcsc-lite
	CCID	pcsc-lite-ccid
USB		libusb
Fyzická	ISO/IEC 7816	

FYZICKÁ VRSTVA

- ISO/IEC 7816
  - Fyzické parametry kontaktu
  - Elektrické vlastnosti
  - Popis příkazů a výměny dat (APDU)
  - Přístupová práva

Command APDU						
Header (required)				Body (optional)		
CLA	INS	P1	P2	Lc	Data Field	Le

- ISO standard -- uzavřené
- Pristupne: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_6_basic_interindustry_commands.aspx
- Popisuje také ostatní vrstvy
  - ne vše se dnes používá:
  - Structured Card Query Language (SCQL)

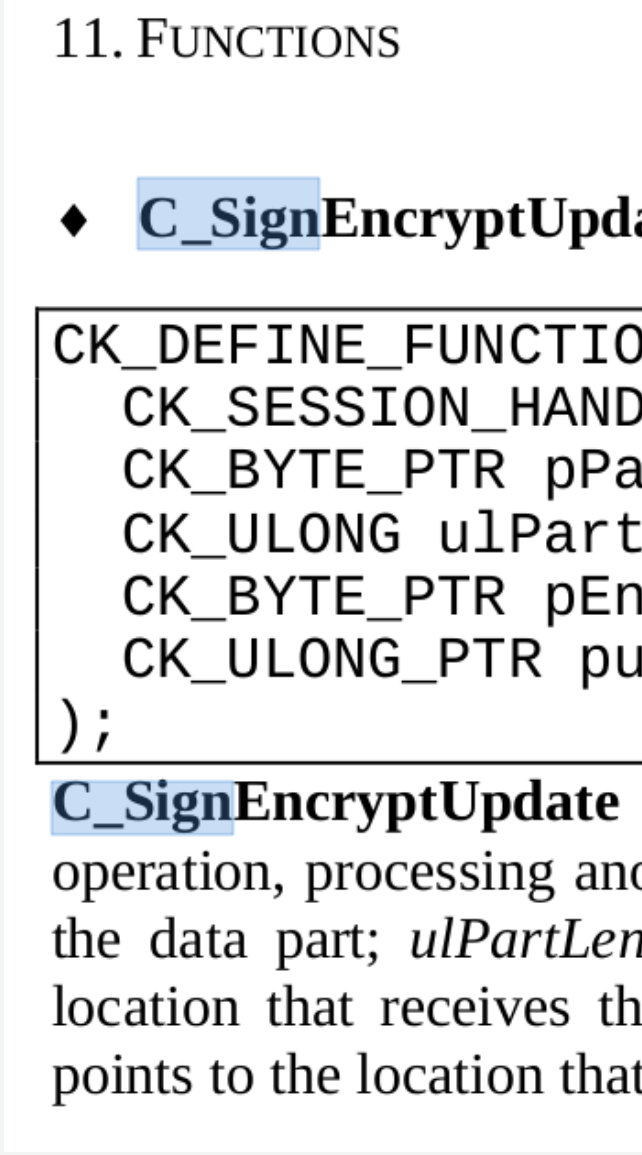
KNIHOVNY

- pcsc-lite
  - Open source implementace PC/SC protokolu
  - pcscd systemovy daemon
  - Přímo používáno OpenSC
- CCID
  - Chip card interface device
  - Součást USB protokolu pro komunikaci se smart kartami
- OpenSC
  - Ovladače pro specifické karty (desítky různých)
  - Vystavuje PKCS#11 rozhraní pro použití v aplikacích
- engine_pkcs11
  - OpenSSL nepodporuje PKCS#11

PKCS#11

cryptographic token interface

- Vysokoúrovňové API pro HSM a Smart karty
- Modul je sdílený objekt (*.so) poskytující C API
- Definuje
  - Kryptografické objekty (RSA klíče, ...)
  - Operace pro použití, generování
- Vývoj:
  - RSA Security (1994)
  - OASIS (od 2013), otevřené standardy


11. FUNCTIONS

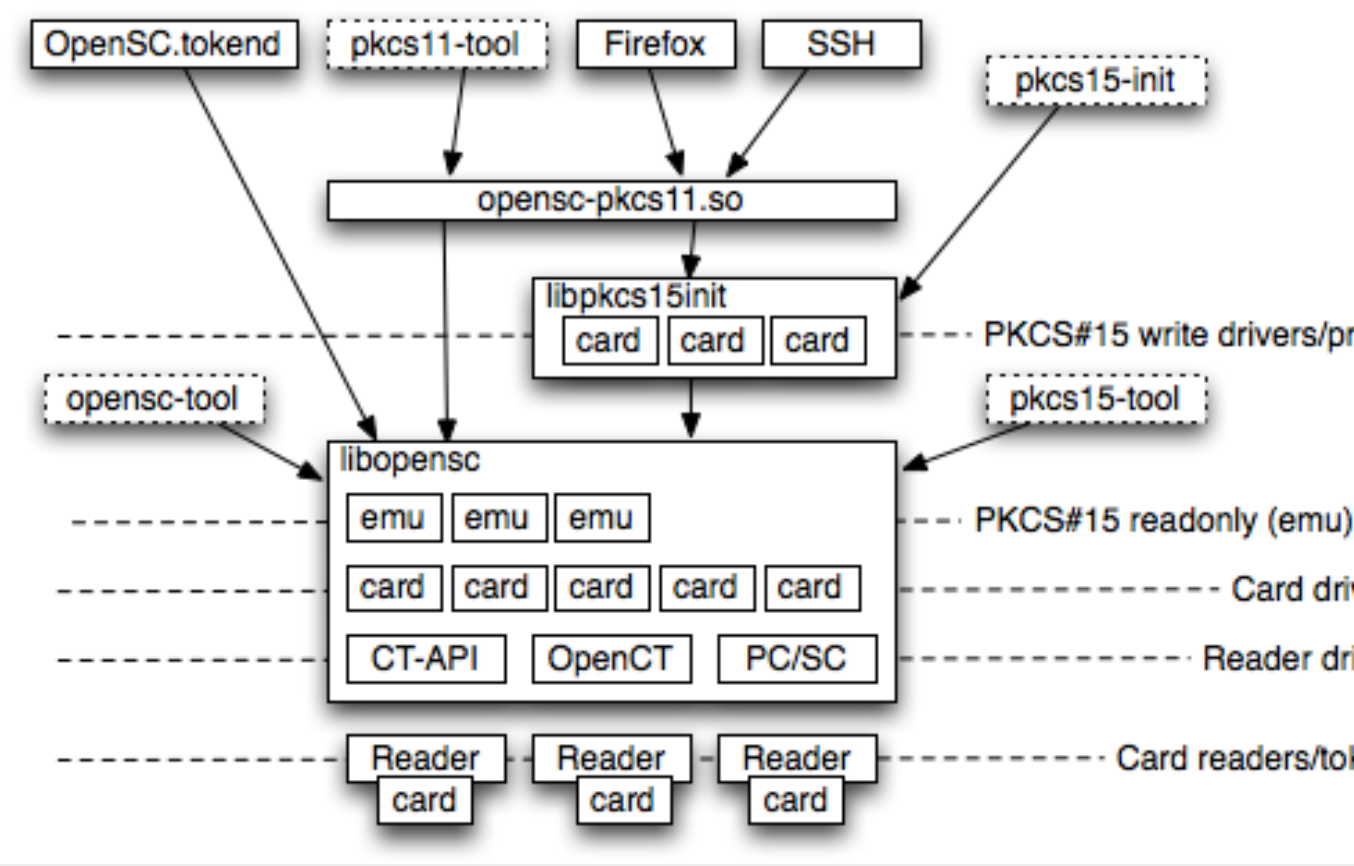
♦ C_SignEncryptUpda

```
CK_DEFINE_FUNCTIO
  CK_SESSION_HAND
  CK_BYTE_PTR pPa
  CK_ULONG ulPart
  CK_BYTE_PTR pEn
  CK_ULONG_PTR pu
);
```

C_SignEncryptUpdate

operation, processing and
the data part; *ulPartLen*
location that receives th
points to the location that

OPENSC



OpenSC

PROJEKT OPENSC

- vznik 2001, Finsko
- Podpora většiny dnešních karet:
  - PIV - US vláda
  - CAC - US armáda
  - Atos CardOS (PKCS#15)
  - myEID
  - Coolkey (RHCS)
  - ...
- Multiplatformní Linux, Windows, macOS
- implementace PKCS#11 API pro ostatní aplikace
- PKCS#15 jako nižší vrstva (kde chybí, emulováno)

MOJE ROLE

- Implementace, podpora a testování karet pomocí CI:
  - CAC - US armáda
  - Coolkey (RHCS)
- Opravy a vylepšení pro ostatní karty (CardOS 5, PIV), dokumentace
- Funkčnost C_WaitForSlot pro interaktivní aplikace
- RSA-PSS podpisy

PRAKTICKÉ UKÁZKY

CERTIFIKÁT Z KARTY

```
# Get certificate from card in DER format
$ pkcs11-tool -r --id 11 --type cert \
    --module /usr/lib64/opensc-pkcs11.so > cert.der

# Convert to more-used PEM
$ openssl x509 -inform DER -in cert.der > cert.pem

# List the content of the certificate
$ openssl x509 -in cert.pem -text
```

PODEPSAT NĚJAKÁ DATA

```
# Sign some data using key ID=1
$ cat data | pkcs11-tool --id 1 -m RSA-PKCS -p 123456 \
    -s --module /usr/lib64/opensc-pkcs11.so > data.sig

# Verify the signature
$ openssl rsautl -verify -certin -inkey cert.pem -in data.sig
```

OPENSSH KLIENT

```
# List public keys on the smart card in OpenSSH format
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so

# Attempt to connect to server example
$ ssh -I /usr/lib64/pkcs11/opensc-pkcs11.so example
Enter PIN for 'PIV_II (PIV Card Holder pin)':

# Store permanent configuration in client config
$ cat ~/.ssh/config
Host shost
  PKCS11Provider /usr/lib64/pkcs11/opensc-pkcs11.so
```

OPENSSH AGENT

```
# Make sure the ssh-agent is running
# (does not work with gnome-keyring!)
$ test -e "$SSH_AUTH_SOCK" || eval $(ssh-agent)

# Add card to the ssh-agent
$ ssh-add -s /usr/lib64/pkcs11/opensc-pkcs11.so

# Connect to server example
$ ssh example
```

- Jaký klíč bude použit?

OPENSSH SERVER

```
# Start ssh-agent for server and add a card
$ eval $(ssh-agent -a /root/agent.sock)
$ ssh-add -s /usr/lib64/pkcs11/opensc-pkcs11.so

# Configure the server with the agent connection
$ sudo cat /etc/ssh/sshd_config | grep -i HostKeyAgent
HostKeyAgent /root/agent.sock

# Start the ssh server
$ sudo systemctl start sshd
```

SUDO (PAM_SSH_AGENT_AUTH)

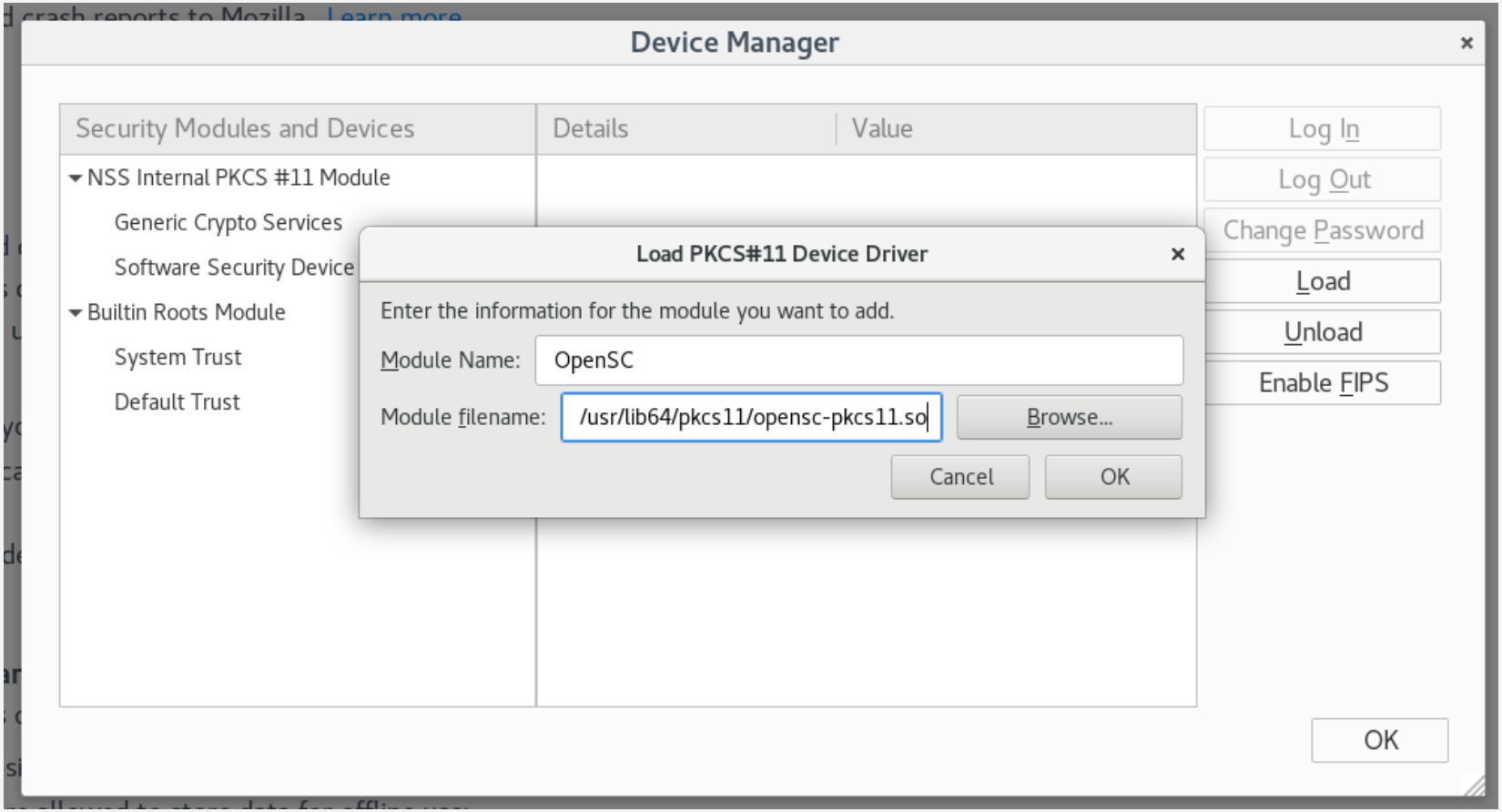
```
# Start ssh-agent for server and add a card
$ eval $(ssh-agent -a /root/agent.sock)
$ ssh-add -s /usr/lib64/pkcs11/opensc-pkcs11.so

# Configure sudo pam stack to accept ssh-agent authenticaitor
$ cat /etc/pam.d/sudo
...
auth sufficient pam_ssh_agent_auth.so file=/etc/security/auth

# Store the public key from the card to
# /etc/security/authorized_keys
```

FIREFOX/THUNDERBIRD

- Preferences -> Privacy&Security -> Security -> Security Devices ...



ODEMYKANÍ ŠIFROVANÉHO DISKU

- Vygenerujeme náhodný symetrický klíč
  - RANDOM_KEY=`openssl rand -base64 32`
- Vytvoříme LUKS key-slot s tímto klíčem
  - cryptsetup luksAddKey /dev/sda2 <(echo "$RANDOM_KEY")
- Tento klíč neuložíme, ale zašifrujeme pomocí veřejného klíče z karty. Výsledek uložíme na disk
- Vytvoříme vlastní LUKS key-script, který dešifruje předchozí soubor pomocí privátního klíče na kartě
  - https://gist.github.com/randomoracle/5922f0f8dc6dfe17b672
- Nastavíme LUKS aby tento key-script spouštěl, když má být oddíl odemčen (/etc/crypttab):

```
home /dev/sdb1 /crypto/fde_ciphertext.bin luks,keyscript=/crypto/unlock.sh
```

SMART KARTY

- Projekt OpenSC
- Bezpečné uložení a použití kryptografického materiálu
- Může nahradit většinu použití hesel
- PKCS#11 rozhraní pro vývojáře

Děkuji za pozornost