

(In)security of smart-cards

Željko Vrba

March 16, 2008

Abstract

Smart-cards are used to implement 2-factor authentication to different systems. Such authentication, compared to ordinary password-based authentication, can increase security level of systems. This lecture will give an overview of smart-card technologies, related standards, and problems that arise in large-scale deployment. The last part of the lecture exposes on concrete examples some of the types of security problems which arise when smart-cards are used with insecure software and operating systems.

- 1 Introduction
- 2 Smart-cards
- 3 Security issues
- 4 Conclusion

- Obtained master-level education at FER, Zagreb.
- Now: PhD student at ND group at IFI; research area: parallel and distributed programming.
- Previous employment: one of chief administrators of Croatian national CA.
- Background / interests: Low-level (close to hardware) programming, mathematics, formal methods, cryptography, operating systems, machine learning.

Back-end

- NDA, but: public tenders!
- High-security environment (physical + SW)
- Root CA + some sub-CAs
- Mix of proprietary SW on different platforms: integration.

Users

- MS Windows: only supported platform (by CA).
- USB reader + smart-card + PKCS#11 and CAPI middleware.
- Possibly have some scd from before.
- FIPS level 2 certification (law).
- Closed APDU specification.

Advantages

- Two-factor authentication.
- Security:
 - Tamper-proof.
 - Non-exportable keys.
 - Certifications (required by some governments).
- Form-factor.
- Possibly multi-application.

Disadvantages

- Need for additional HW (card, reader) and SW (middleware) (= extra cost).
- Non-trivial installation, hard to use for average user.
- As secure as the weakest link: usually SW using it or the OS (think worms, viruses).

Technical

- Static data storage.
- 8–64kB EEPROM capacity.
- May have also a CPU and cryptographic coprocessor (RSA).
- Java interpreter, *cardlets*, GlobalPlatform.
- Serial, USB or contact-less connection to host.
- Embedded on an USB token.

Security

- EAL, 7 levels: quality assurance of development methodology. (higher level is *not* more secure – functionality).
- FIPS, 4 levels, *security policy*
 - ① Limited requirements
 - ② + Tamper-evidence, role-based authentication.
 - ③ + Tamper-resistance, identity-based authentication.
 - ④ + Environmental attacks.

ISO 7816

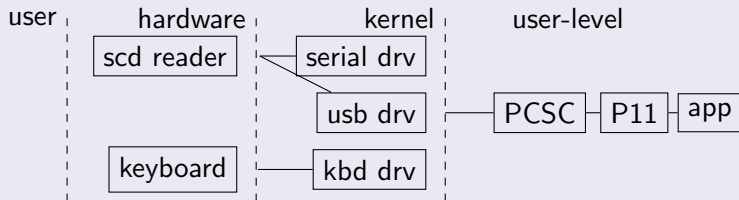
- http://www.cardwerk.com/smartcards/smartcard_standards.aspx
 - ① Physical characteristics.
 - ② Contact dimensions and locations.
 - ③ Transmission protocol (wire).
 - ④ Command APDU format.
- (in total 15 standards today; 8th: security operations).
- Sufficient for full life-cycle management.
- Not widely deployed; mostly proprietary APDUs.

High-level programming interfaces

- RSA's PKCS#11 (multi-platform) and PKCS#15
- Microsoft's CryptoAPI (only Windows)

Running environment

Layered software architecture...



... and its weaknesses

- Weak or nonexistent protection between layers.
- Rich and powerful debugging primitives.
- Dynamic linking (function hijacking)
- Generic capabilities over a process: attach, list loaded libraries, get function's address, attach to process, read/write its memory and registers, set breakpoints

Exploitation techniques (1)

Library hooking to PKCS#11 or CAPI

- Intercept calls to C_Login, C_Sign, C_Decrypt functions.
- PIN theft, signature and decryption of arbitrary data.

Intercept PCSC or USB traffic

- Intercept APDUs: harder because of lack of documentation.
- Intercept traffic at USB level:
 - Windows filter drivers; useful for protocol analysis.
 - <http://www.hhdsoftware.com/Family/usb-monitor.html>
 - <http://usbsnoop.sourceforge.net>

Exploitation techniques (2)

DoS attacks

- Deliberately present wrong PIN to the card.
- May also try PUK, or other “master” authentication mechanism.
- Can end up in absolutely unusable card.
- Non-extractable key: what if used for encryption?

Other attacks

- Memory, swap file and core-dump scanning.
- Software and hardware keyloggers.
- Firewire bus.
- Cameras and microphones.
- Exploit “by design” bugs in middleware.
- Everything is easier than breaking cryptographic primitives!

- Pinpad (prevents only PIN theft).
- Static linking (*harder* to intercept library calls).
- Encrypted communication (*somewhat* effective against traffic interception: keys still in RAM)
- Proper coding (locking and zeroing sensitive data in RAM).
- Proper OS configuration (disable core dumps, module loading and debugging capabilities).

- EU directive on ES: non-repudiation, qualified certificates and qualified ES.
- QES: legally valid as a hand-written signature.
- *But*: signature is a conscious act of acceptance and/or taking responsibility.
- What if:
 - A person prepares a virus and waits for it to spread.
 - Willfully commits a transaction and claims it wasn't done by him, but by some virus.
 - Closer inspection will indeed uncover the virus.
- Questions: Who takes liability in the court? Are users required to be computer security experts to be able to use smart-cards?

Banking tokens

- No setup, almost trivial usage.
- Lower “mathematical” security, higher “factual” security (no way to subvert it).
- Even better if combined with signed challenge (*not* e.g. Nordea, but *yes* e.g. Zagrebačka Banka).

High-end cryptographic modules (nCipher, Thales)

- Large memory, multiple keys.
- Cryptographic accelerators.
- Trusted execution engines (CA in a crypto-module).
- Expensive (list price: ~ few kEUR/piece).

Situation today...

- Hard to properly secure OS-es.
- Security flaws:
 - Programmability of the host system.
 - Layered architecture without any protection between layers.
- Not very user-friendly (actually, expert-only when problems arise).

...and tomorrow

- AMD's secure virtual machine (Pacifica)
- Intel trusted execution mode (former LaGrande)
- Hardware-supported absolute isolation of programs.