PV204 Security technologies LABS

Introduction to smart cards

Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CROCS
Centre for Research on
Cryptography and Security

The masterplan for this lab

- Secure channel and smartcard communication

1. Building Secure Channel protocol (together)
   - simple protocol → design attack → fix it → iterate
2. Communicate with smart card (GPPro tool)
   - ATR, basic info, CPLC
3. Communicate with card programmatically
   - Java java.smartcardio.* or C/C++ PC/SC API
   - CPLC data
   - Obtain list of supported instructions from unknown card

1. Building Secure Channel protocol

- Scenario: we like to transfer extrasupersensitive data between PC and smartcard
- Simple protocol → design attack → fix it → iterate
- Participate in discussion
  - (Steps and solution will be published after the labs in IS)

2. Communicate with smart card (GPPro)

- Contact PC/SC readers + cards
- GlobalPlatformPro tool
  - https://github.com/martinpaljak/GlobalPlatformPro/releases
  - Basic smart card commands, sending APDUs
  - Management of GlobalPlatform cards (JavaCard)
  - Type `gp --help` for all functionality
  - We will use basic functionality now, rest next week

gp --info

- Obtain information about smart card
  - gp --info
  - Obtain ATR (Answer To Reset)
  - Parse using https://smartcard-atr.appspot.com/parse?ATR=xxx

- Who is probable manufacturer of card?
- What is probable OS environment for this card?
- Is it JavaCard?
- What is card’s circuit serial number?
- When was card produced?

gp --apdu APDU_in_hexa --debug

- Send APDU command from command line
- Try gp --info --debug
  - Can you spot APDU command to obtain CPLC info?
- Send get CPLC APDU separately
  - gp --apdu 80CA9F7F --debug

- Can you relate card’s response data and gp --info?
- What is response status word?

3. Communicate with card programmatically

- SimpleAPDU project (IS, NetBeans)
  - Uses Java’s javax.smartcardio.* API
  - CardMngr.java – utility functions for card communication
- Obtain list of available readers
  - List readers = TerminalFactory.getDefault().terminals().list();
- Connect to card
  - CardTerminal.isCardPresent(), CardTerminal.connect(“*”);
- Obtain ATR: Card.getATR().getBytes()
- Send APDU:
  - ResponseAPDU resp = CardChannel.transmit(apdu)

3. Communicate with card programmatically

- Try to send get CPLC command
  - Pre-prepared in GetCPLCData() method
  - Necessary to set proper APDU
- Parse response buffer

- Can you relate card’s response data and gp --info?
- What is value of response status word?

Supported commands

- Card responds to some APDU commands
  - Generic ones (e.g., get CPLC data)
  - Custom ones (what card's owner wants)
  - Usually CLA/INS/P1 only (P2 sometimes)
- How to get list of commands supported by a card?
  - Look into documentation / standard (e.g., SIM commands)
  - Try to probe card (limited number of possible commands)
    - Be careful – many failed attempts may block your card!

Obtain list of supported commands

- Write code that will try all combination if CLA/INS
- Observe response codes
- Make list of CLA/INS which returns interesting code
- Analyse with curiosity!

SOLUTIONS – KIND OF ☺

Building SCP – steps in solution

- Scenario: we like to transfer extrasupersensitive data between PC and smartcard

1. Simple exchange in plaintext
2. Encrypted by static symmetric key
3. Integrity protection using plain hash
4. Integrity protection using MAC (CBC-MAC,HMAC)
5. Counter/Hash chain for message freshness and semantic security
6. Authentication based on static key
7. Challenge response for fresh authentication
8. Session keys derived from master key(s)
9. Forward secrecy based on RSA/DH

Building SCP – steps in solution

1. Simple exchange in plaintext
   - Many problems, attacker can eavesdrop sensitive data
2. Encrypted by static symmetric key
   - Attacker can modify sensitive data (no integrity)
3. Integrity protection using plain hash
   - Hash is not enough, attacker can modify then recompute hash
4. Integrity protection using MAC (CBC-MAC,HMAC)
   - Attacker can replay older message (no freshness)

Building SCP – steps in solution

5. Counter/Hash chain for message freshness and semantic security
   - No explicit authentication of parties
6. Authentication based on static key
   - Authentication message can be replayed from previous legit run
7. Challenge response for fresh authentication
   - Single static key can cause problems
     - Interchange of encrypted message and valid MAC
     - Large amount of data encrypted under same key (cryptoanalysis)

Building SCP – steps in solution

8. Session keys derived from master key(s)
   - If master keys are compromised, older captured communication can be decrypted
9. Forward secrecy based on RSA/DH
   - Secure?
   - Key management with multiple parties?
   - Proof of message origin? Deniability?
   - … gather your requirements!

gp --info

- Who is probable manufacturer of card?
  - Gemplus/Gemalto
- What is probable OS environment for this card?
  - MPCOS
- Is it JavaCard?
  - No
- What is card’s circuit serial number?
  - ICSerialNumber: 02006FC1 (Note: your card will be different)
  - Good to consider also other ICxxx values for uniqueness
- When was card produced?
  - ICFabricationDate: 1105
  - Probably 15th May 2011 (105th day of year ending with 1)