Smart Card Concepts

Carte à puce et Java Card

ATAC

2011-2012

Jean-Louis Lanet
Jean-louis.lanet@unilim.fr
Agenda

• Card Technology
• Standards
• Manufacturing
• Operating system
Magnetic-strip cards

- Defined by ISO 7811-2 (properties) -4 (coding) –5 (location of the magnetic stripes)
- Storage capacity 1000 bits

<table>
<thead>
<tr>
<th>Features</th>
<th>Track 1</th>
<th>Track 2</th>
<th>Track 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of Data</td>
<td>79 char</td>
<td>40 char</td>
<td>107 char</td>
</tr>
<tr>
<td>Data Coding</td>
<td>6 bit alpha</td>
<td>4 bit BCD</td>
<td>4 bit BCD</td>
</tr>
<tr>
<td>Data density</td>
<td>210 bpi</td>
<td>75 bpi</td>
<td>210 bpi</td>
</tr>
<tr>
<td>Writing</td>
<td>Not Allowed</td>
<td>Allowed</td>
<td>Allowed</td>
</tr>
</tbody>
</table>
What is a Smart Card?

A piece of silicon on a plastic body

A very secure way of storing a small amount of sensitive data
Smart Cards

- Smart card
  - Memory card
    - Simple
    - Protected
  - Micro. card
    - Dedicated
    - Proprietary
    - Open
    - Crypto Proc
Contact / Contact less
Contact

• Electrical connections between the chip and the module (wire bonding process),
• 8 contacts (C1-C8) but only 6 used (see ISO7816-2),
• C6 used as Vpp while EEPROM where not embedding charge pump,
• Supply voltage 2,7v (SIM) to 5,5v (standard TTL) and clock provided by the reader.
Contact less card (NFC)

• No electrical connection (cf. RFID technology) used of inductive coupling to supply power to the chip,
• Need : modulator, demodulator, anti-collision mechanism, voltage regulator, reset generator and an aerial.
• For data transfer all known digital modulation techniques can be used (ASK, FSK and PSK).
• Standards : close coupling ISO/IEC 10536 (3-5Mhz), proximity cards ISO/IEC 14443 (13,56Mhz) and Hand Free Cards ISO/IEC 15693,
• Used for public transportation, ski pass, access control, payment with GSM…
Form Factor

- With contact:
  - ISO 7810, 7816-1, 1816-2
  - USB
- Contactless: several standards
- Hybrids
- Buttons
  - iButton (1-wire)
  - JavaRing
- Dongle (serial, parallel, USB, mmc)
Microprocessor architecture
Microcontroller of the card
Contact card

- Microcontact
- Microchip
- Micromodule

Microprocessor

Data Bus

Address Bus

EEPROM / FLASH
ROM
RAM

Vcc
Ground
Reset
Clock
I/O

Vpp
CPU

Clock

Reset

Ground

I/O

Vcc
Different Types of Memory ...

• ROM : CPU only NO ACCESS!
  – used for embedded Operating System

• EPROM : Write once, read FOR EVER!
  – Used for initialization area (eg. Lock bytes)

• EEPROM : Write, erase, read FLEXIBLE!
  – used to store applicative data or added functionalities

• RAM : Write, erase, read TEMPORARY!
  – used during power on sessions only
New Non Volatile Memories

• Flash EEPROM Memory :
  – Advantages :
    • Same memory for Program and Data
    • Time to Market reduced for prototyping
    • Cell size (element to store 1 bit) ratio vs. E² : 1/3 smaller
  – Disadvantages :
    • Granularity Data memory : 512*32 comparing with 1-byte access.
    • Erase time more important than E² memory
    • Cell size larger than ROM
New Non Volatile Memories

• FeRAM :
  – Advantages :
    • Same memory for Program and Data and computing area (RAM).
    • Same access time for Read, Erase and Write (same as DRAM)
    • Cell size ratio vs. E² : 1/3 smaller
  – Disadvantages :
    • Technology under development (new Technology)
# Smart Card memories

<table>
<thead>
<tr>
<th></th>
<th>RAM</th>
<th>EEPROM</th>
<th>FlashRAM</th>
<th>FéRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistency</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Read acc.</td>
<td>0.1µs</td>
<td>0.15µs</td>
<td>0.15µs</td>
<td>0.15µs</td>
</tr>
<tr>
<td>Write</td>
<td>0.1µs</td>
<td>10µs</td>
<td>10µs</td>
<td>0.4µs</td>
</tr>
<tr>
<td>Erase</td>
<td>-</td>
<td>5ms</td>
<td>100ms</td>
<td>-</td>
</tr>
<tr>
<td>Granularity</td>
<td>-</td>
<td>4bytes</td>
<td>64bytes</td>
<td>-</td>
</tr>
<tr>
<td>Cycles</td>
<td>Unlimited</td>
<td>$10^6$</td>
<td>$10^5$</td>
<td>$10^{10}$</td>
</tr>
</tbody>
</table>
## Comparing Smart Card vs. PC

<table>
<thead>
<tr>
<th></th>
<th>Smart Card</th>
<th>PC</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RAM</strong></td>
<td>1kbyte</td>
<td>128Mbyte</td>
<td>130 000</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>64kbyte</td>
<td>6Gbytes</td>
<td>100 000</td>
</tr>
<tr>
<td><strong>Baud rate</strong></td>
<td>192 kbits</td>
<td>100Mbits</td>
<td>500</td>
</tr>
<tr>
<td><strong>CPU Speed</strong></td>
<td>20 Mips</td>
<td>500Mips</td>
<td>25</td>
</tr>
</tbody>
</table>
Agenda

• Card Technology
• Standards
• Manufacturing
• Operating system
ISO/IEC 7816
Integrated circuits cards with contacts

- ISO/IEC 7816-1: Physical characteristics.
- ISO/IEC 7816-4: Inter-industry commands and file system.
- ISO/IEC 7816-5: Registration system for applications in IC card.
- ISO/IEC 7816-6: Inter-industry data elements.
- ISO/IEC 7816-7: Inter-industry commands for Structured Card Query Language (SCQL).
ISO/IEC 7816-1 (7810)

- Governs the physical characteristics of a smart card:
  - Thickness: 0.76 mm
ISO/IEC 7816-2

• Governs the dimension and location of the chip contact
ISO/IEC 7816-3

• Electrical characteristics:
  – clock frequency [1 MHz, 5 MHz],
  – communication speed.

• Transmission protocols:
  – T=0, T=1, T=CL defined,
  – T=14 reserved for proprietary protocols.

• Answer to reset (ATR)

• Protocol type selection (PTS):
  – If several protocols supported or if parameters need to be adjust
  – Negotiable mode and specific mode
Smart Card Reader exchange

• The card NEVER initiates a communication with the reader

Smart Card introduction

Response to the ATR

Protocol negotiation PTS

Negotiation answer PTS

Command APDU

Answer APDU

End of session
Answer To Reset

• Starts the smart card program,
• Data elements TS-T0-Tabcd-T1…k-TCK
  – TS: Byte coding convention (3B direct, 3F inverse)
  – T0: Format characters
  – Ta,b,c,d: Interface characters,
  – T1..Tk: Historical characters to identify OS, version number of the ROM mask, can be omitted.
  – TCK: XOR checksum from T0 to the last byte before TCK.
Protocol Type Selection

- Needed only if the terminal wants to modify parameters,
- If the card agrees, it sends the PTS back to the terminal
- Otherwise the terminal execute a reset (warm => protocol change),
- Only one PTS after the ATR.

Diagram:

- Smart card
- Terminal
- ATR
- PTS necessary?
- PTS ok?
- PTS Response
- Command
- PTS request
- No
- Wait for reset
- No
Transmission protocols

- T=0 most widely used (1989), T=1 block oriented
- T=14 Japan and Germany

<table>
<thead>
<tr>
<th>Transmission protocol</th>
<th>Meaning</th>
<th>ISO</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=0</td>
<td>Asynchronous, half duplex, byte oriented</td>
<td>7816-3</td>
</tr>
<tr>
<td>T=1</td>
<td>Asynchronous, half duplex, block oriented</td>
<td>7816-3</td>
</tr>
<tr>
<td>T=2</td>
<td>Asynchronous, full duplex, block oriented, tbs</td>
<td>10536-4</td>
</tr>
<tr>
<td>T=14</td>
<td>National functions</td>
<td>No ISO</td>
</tr>
</tbody>
</table>
Transport protocols

• T=0
  – Byte oriented, Serial transmission (1 start bit, 8 bits data, 1 parity bit, 2 stop bits)
  – Transmission error (parity only) 2 etu mute (“0”)

• T=1
  – Block oriented, Header : NAD, PCB, LEN; data : INF, CRC.
  – NAD 3 bits destination address, 3 bits source address
  – PCB define the kind of block
    • I (#block, more) numbered mod 2, more = 1, another block follow
    • R(#block, error) numbered mod 2, next expected bloc,
    • S specific command (RESYNC, IFS, ABORT, WTX)
The Application Protocol Data Unit

- Independence of application versus low layers
- An APDU contains either:
  - a command message,
  - a response message.
APDU syntax

<table>
<thead>
<tr>
<th>Header fields</th>
<th>Body fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>Lc</td>
</tr>
<tr>
<td>INS</td>
<td>Data Field</td>
</tr>
<tr>
<td>P1</td>
<td>Le</td>
</tr>
<tr>
<td>P2</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

CLA : class
INS : instruction
P1 : parameter 1
P2 : parameter 2
Lc : length of command data
Le : expected length of the response
Response syntax

Le : length of the expected response
SW1: Status Word 1
SW2: Status Word 2
## CLA Class byte

<table>
<thead>
<tr>
<th>b7 to b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Logical channel number</td>
</tr>
<tr>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No secure messaging</td>
</tr>
<tr>
<td>1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Secure messaging header not authentic</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Secure messaging header authentic</td>
</tr>
<tr>
<td>‘0’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Structure and coding compliant with 7816-4</td>
</tr>
<tr>
<td>‘8’, ‘9’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>User specific codes</td>
</tr>
<tr>
<td>‘A’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Structure and code defined in additional document GSM11.11</td>
</tr>
</tbody>
</table>

## Class

<table>
<thead>
<tr>
<th>Class</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>’80’</td>
<td>Electronic purse compliant with EN 1546-3</td>
</tr>
<tr>
<td>‘8x’</td>
<td>Credit card compliant with EMV-2</td>
</tr>
<tr>
<td>‘A0’</td>
<td>GSM compliant with prETS 300 608/GSM 11.11</td>
</tr>
</tbody>
</table>
Four possibilities

Case 1: \( L_c = 0 \) and \( L_e = 0 \)

Case 2: \( L_c = 0 \) and \( L_e \neq 0 \)

Warning \( L_e = 0 \) means 256 bytes expected

The difference between case one and two is made with the command not at the protocol level!
Four possibilities

Case 3: $L_c \neq 0$ and $L_e = 0$

Case 4: $L_c \neq 0$ and $L_e \neq 0$

Commande Get Response
Return Codes

- SW1, SW2 = '90 00' command successful, '63xx' or '65xx' means EEprom has been modified,
- More than 50 different return codes defined by standard,
- Often not respected…

Process completed

Return code

Process aborted

Normal processing

’90 00’
’61 xx’

Warning processing

’62 xx’
’63 xx’

Execution error

’64 xx’
’65 xx’

Checking error

’67 xx’ to ’6F xx’
ISO/IEC 7816-4

• There are no user programs, no memory management and no parallelism.

• It just defines the file system
  – Specifies contents of messages (commands, responses).
  – Structure of files and data.

• and the security architecture
  – Access methods to files and data.
  – Methods for secure messaging.

• But also the filter mechanism.
Agenda

• Card Technology
• Standards
• Manufacturing
• Operating system
Manufacturing cycle (1/3)

- Wafer production
- Testing
- Sawing
- Bonding

Motorola, Atmel, Texas Instruments, STMicroelectronics, Siemens, Hitachi
Manufacturing cycle (2/3)
Manufacturing Cycle (3/3)

Test et personalization (soft mask, card key, Embedded number, card batch)

Personalization by issuer - Fixed and derived data - Private key - Applets

Distribution + Mailing

CB Carte bleue

142635 Lanet
Cutting silicon wafer into individual chips. During the previous step, electrical test, defective chips are marked with an ink drop.
Manufacturing : Testing
Die Bonding

Gluing the chip into the cavity located on the film, ensuring proper physical and electrical connection.
Manufacturing : Bonding

Electrically connecting the chip's bonding pads and the contacts on the micro module using gold wires.
Bonding
Manufacturing: potting

Protecting the chip and wires with a drop of epoxy resin, ensuring the physical durability of the micro module.
Grinding the micro module to the proper thickness prior to embedding.
Electrical Testing
Manufacturing: finished modules
Card Moulding

Injection of ABS plastic material to form the body of the card
Offset Printing
Grinding

The micro module cavity is machined to specific dimensions.
Manufacturing : Embedding & Test
Plug-In

Cutting to plug-in format for mobile phones.
Personalization
Packing
Agenda

- Card Technology
- Standards
- Manufacturing
- Operating system
Fundamentals

- Functions: manage the resources
- Program written in ROM code (no self modifying techniques allowed),
- No change are possible once the chip is manufactured
  - quick and dirty programming IS NOT AN OPTION !!!
  - Smart Card OS is reliable and robust,
- Design consideration:
  - Persistence…
  - Closely coupled with the hardware
Introduction

• At the beginning no real OS only stand alone applications
• Mask your own code
  – Pros: small footprint, complete control
  – Cons: development in C and target assembly language, use emulators, Mask lead time 2 months, bug fixes.
Development 7816-4

• Use proprietary cards
  – What you get
    • File system
    • Fixed set of APDU commands: read/write files, cryptographic primitives
  – Pros: off the shelf product, cheaper
  – Cons not extensible, bug fixes.
Example

- P1=Offset High,
- P2=Offset low.

Syntax:

```
CLA  INS  P1  P2  Le
A0   B0   xx  yy  Le
```

P1, P2 : specify the data to be retrieved
Le : length of data to retrieve
7816-4 based OS

- Data are stored in files structures in Eeprom,
- A file must be selected before any action,
- Made of a header and a body,
  - The header stores the access conditions and the structure of the file.
  - For security reasons header and body are stored on different eeprom pages
Soft Masks

• It is an extension of the hard mask
• Often written in C, compiled and linked to the libraries,
• Can be download in Eeprom if the card is not blocked,
• Need ?
  – Bug fixes,
  – Adding new functionalities,
  – A customer needs a rewriting of a command…
OS based on 7816-4

• **Pro**
  – Cheap, easy to use
  – Possible to insert new commands

• **Cons**
  – Unable to execute code
  – Frozen after personalisation phase,
  – Data oriented.
Time to market

• Time between decision and product launch
• Could take as one year if mask need to be redevelop
• Not really adapted to current market :
  – Mobile phone is a highly competitive market,
  – Interoperability is needed,
  – Development cost are too important,
• Smart card manufacturers developed generic smart card: open cards
  – With real operating system
  – Able to download application during their life cycle
Applet life cycle

- Card not issued
- Card is deployed

- SIM card
- Adding an e-purse
- Adding a loyalty
- Removing the loyalty
Open cards

• From a developer point of view:
  – Until now, writing an application required a specific knowledge,
  – No need of smart card specialists,
  – Solution : use general purpose programming language (C, Java, Visual Basic…)
  – Much more easier to integrate applications,
  – More tools to test applications,

• From an end-user point of view
  – Several application on a single card,
  – Possibility to load/unload application when needed.
Smartcards of the present days

• Java Card
  – Embedded virtual machine,
  – Open standard (Java Card 2.2),
  – Wide support of the industry
    • IBM, Visa,…
  – Reduction of development time.
Applet development

• Write code in Java
• Compile it
• Debug it (simulator)
• Verify and Convert it (specific byte code)
• Load it
  Personalization center
  Point of sale
  Over the Internet
MULTOS

- Based on the MEL (Multos Executable Language) interpreter.
  - Operating system and memory firewalls
  - Virtual Machine layer to provide abstraction
  - Application Programming Interface (API)
  - Application management including secure loading and deleting methods.

- See http://www.multos.com
Basic Card

- Based on the Basic language
  - DOS like file system,
  - P-Code byte code interpreter
- PRO
  - Fit well for a small amount of cards
- CONS
  - Not supported by major smart card manufacturers
  - Proprietary code (http://www.zeitcontrol.de)
Next step?

- Smart Card Web Server: portable web server,
- Access to the secret stored in the card through your browser,
- Use the USB port, TCP/IP protocol, data security through SSL, multi-threading, full garbage collection… *id est* JC3.0
Any question?