EMBEDDED SECURITY SYSTEMS 2015

Mirosław Kutyłowski

&grades: 40% lecture, 60% lab

exam, no tests during the course, exam in English unless...

short problems, skills examined not knowledge

lower bound: 40% 3, 50% 3.5, 60% 4, 70 % 4.5 80% 5.0

Objectives

presentation of architecture, limitations and fonalities of embedded systems used in security area C2 developing programming skills concerning cryptographic smart cards and FPGA

_____

1. smart cards    ≈8 hours

2. telecommunication systems ≈2 hours

2. HSM, TPM, remote attestation ≈4 hours

3. FPGA ≈4

4. sensor systems ≈2 hours

5. RFID tags ≈4 hours

6. CUDA and parallel programming ≈4 hours

_____

## 1. SMART CARDS

**cards of no-smart solutions:**

- **embossed** - credit cards: reading does not require electricity, manipulation more difficult then with magnetic strip

- **magnetic:** ≈1000 bits, 3 tracks, track 1: 79 6bit chars, track 2: 40 4bit chars, track 3: 107 4-bit chars, limited density (movement in reader against the head), standard data, track 3 for read-write, no physical protection, cheap readers, accidental erasure by a nearby magnet,  horror as ATM cards

**smart cards: classification**

- memory cards (with security logic and without)

  usually memory: non-volatile EEPROM, serial communication, control logic: where you can write. Cheap. E.g. prepayed telephone cards

- processor: with coprocessor  or without (as bad as: RSA in 20 minutes)

- contact or wireless

- multimegabyte

**Contactless cards:**

- energy: inductive (low!)

- small range (typicaly 10 cm)

- a reader may activate it from distance

- response with low energy, recognizable from a short distance only

- memory: kilobytes

- well sealed against corrosion

- main parts:

    - antenna (most area of the card)

    - electronic part: modulation, demodulation, clock generator, voltage regulation, reset generation

    - interface between RF interface and memory chip

    - access logic

    - application data: EEPROM, ROM

**processor cards:**

I/O ——— CPU —-- flash memory

RAM

ROM

|

NPU (numerical processing unit)
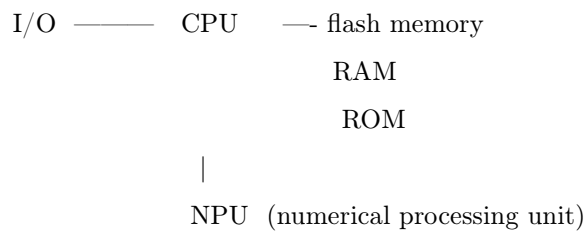
**contacts**:

- 8 fields, normally 6 used (2 for future applications), places for contacts strictly determined in standard

- ground (GND), voltage (Vcc) , I/O, clock (CLK), CTRL, sometimes: Single Wire Protocol (SWP), USB

- easy to destroy

- corrosion, mechanical scratches, not for intensive use

**security tokens:**

- type 1: USB tokens – contact interface like in USB, insert into a port after breaking out of a card

- type 2: small display (eg. 4 digits). input also possible: e.g. a card with 2 buttons (each one side of the card), battery inside

**optical:**

- writing technique - like CD (linear and not circular)

- area designated field according to standards, may leave place for contact interface of a chip and  magnetic strip, less place for graphical part on the card

- megabytes  ($\approx 6$MB storage)

- redundancy, therefore not easy to destroy information

- usage: e.g. border control cards (Mexico-USA)

**Physical properties:**

standard format: 85.6x54 mm (ID-1), other formats for SIM cards (in larger ID-1 cards with stamping),

parameters:

- mechanical robustness (card and contacts)

- temperature resistance

- surfice

- electrostatic discharge

- electromagnetic susceptibility

- ultraviolet radiation

- X-ray radiation


**Material**: trade-off with different properties

PVC: polivinyl chloride, credit cards, cheap, problems with low and high temperatures, injection molding impossible, lifetime 2 years, cost factor: 1

ABS: mobile, termally stable up to 100 C, laser engraving poor, lifetime 3 years, cost factor: 2

PC: polycarbonate, ID cards, durable, 160 C, problems with hot stamping, lifetime 5 years, cost factor: 7, low scratch resistance,

PET: health cards, mechanical: very good resistance, lifetime 3 years, cost factor: 2.5


**Graphical security means:**

- Guilloche patterns - fine lines on the surface under the outer transparent foil, in case of any manipulation the pattern destoryed. Technique used on bank notes

- colored signature field - printed paper strip glued to the surface

- microtext - look like simple lines but something printed – used on bank notes, defence against photocopying, readable only under a loupe

- ultraviolet ink

- barcodes (one and two dimensional), two dimensional PDF 417 can encode up to 1000 bytes, error correction codes so that up to 25% of the surface can be damaged (dirty)

- hologram - few companies in the world, cheap, holograms are embossed holograms, holograms reflected in diffuse daylight (some holograms require laser light), permanently bonded to the surface microstructure,

- kinegram - as holograms, show different image from different angles.

- MLI: (multiple laser image) - small lenses, some are blackend by the laser. Looks like a hologram but can contain personalized information (holograms are always the same)

- embossing - like in credit cards (the characters are pressed with a considerable force). Rather old style...

- laser engraving (surface or inside, uder the coat) – equipment fairly expensive, used to personalize cards. However, it is slow (major slow down for production of ID cards). However - a professional forger can make corrections ...

- scratch field – nice for card delivery. The character printed under the coat are not readable even with ultraviolet, infrared light etc

- thermochrome (TC) display: not a real display, but can be reprinted with a special reader. heating a point to $120^oC$ makes a black dot. Heating the whole strip makes it almost transparent again

- MM (modulated feature) – hidden MM box, invisible, contains control digits for the contents of the magnetic strip. Used by POS and ATM terminals. Control digits computed with MM algorithm

**Chip modules:**

- the chip too fragile and too thick to be laminated on the surface. It is inserted inside

- electrical connections are the problem, automatic bonding of the gold wires to the back of contacts with ultrasonic welding

- Chip-on flex modules, stages of production:

  - tape with empty modules

  - gluing the dice into modules

  - bonding the dice

  - encapsulating the dice

- lead-frame: chip produced together with contacts and the simply inserted by a robot into the card body and glued

**Electrical properties:**

- 8 connections, 2 auxiliary and can be omitted or used e.g. for USB

  connections:

  - C1: Vcc voltage supply

  - C2: RST reset

- C3 CLK clock

- C4 AUX1

- C5 GDN ground

- C6 SPU standard or proprietary use (SWP)

- C7 I/O

- C8 AUX2

```
−−−−−−
|C1   C5 |
|C2   C6|
|C3   C7|
|C4   C8|

−−−−−−
```

- max 60 mA for 5V, max ambient temperature 50 degrees, 350 $\mu A$ per megaherz, power consumption too low to cause overheating, power reduction e.g. for SIM in different phases of activity (low if the phone is not transmitting and using cryptoprocesor)

- contact C6 was for EEPROM erasing but not needed anymore, used for Single Wire Protocol

- voltage is a problem: 3V for SIM cards (batteries for smarphones weight optimization), 5V needed for EEPROM erasure. charge pumps applied

- no internal clock supply (potential risk: adversary may increase the clock frequency to create faults, fault cryptanalysis)

- problems with collisions on I/O line (too high currents would destroy interface components)

- protection against out of range voltages, electrostatic charges, precisely defined activationa and deactivation sequences: first ground, then voltage, then clock, warm reset when voltage increases on the reset line

**Microcontrollers:**

- area: manufacturing costs and durability (bending, torsion), typically $10\text{mm}^2$, square shape

- must be integrated, "standard components" are not well suited due to size of the resulting circuit,

- native designs are proprietary, even a crime to check the layout

- semiconductor technology -> density increases -> chip area drops . But some problems: error probability, necessity to decrease voltage, ...

- extremely high reliability needed. So behind the "state-of-the-art" which is frequently instable

- memory small (e.g. 100KB), a 8-bit processor ok for less than 64KB, then extensions, usually CISC (complex instruction set computer) - instruction over  a number of steps, some based on RISC (reduced instruction set computer), also 32 bit processors that needed also for interpreter based architectures (Java Card)

―――――――――――――――――――――――――――――――――――――――――――――――――――――-

# MEMORY

**Memory** types:

**non volatile:**

**EPROM** - UV erasure, not suited for  smart cards,

**EEPROM** - electrical erasure, cell capacitors, discharged state=0, charged state=1, erase state -> non-erased (single bit), non-erased-> erased: page or sector, both slow, size 1.14 $\mu$m, 100.000-1.000.000 erasures, 2-10ms, tunneling efect - if there are electrons on the floating gate then they prevent flow in the substrate

**flash** - a different technique for writing: hot electron injection, write time „flash", erase like EEPROM, size 0.47, 10.000-100.000 erasures, very fast writing, lower voltage (12V) than EEPROM (17V), NOR flash: free read of individual cells, but complicated circuits, or NAND flash: dense but reading full blocks

**ROM** - connections broken – memory via a circuit, irreversible process - one disconnected never can be reconnected, lack of connection = 0, small: $0.54\mu$m area size

**volatile:**

RAM - transistors, flip-flop, size 1.87 $\mu$m area size, erasures - unlimited, write: 70ns

―――――――――――――――――――――――――――――――――――――――――――――――――――――-

# AUXILIARY UNITs:

**UART** - universal asynchronous receiver transmitter, software solutions would be too slow

**USB** – USB connection has rigid timing requirements, they cannot be guaranteed by the regular chip, 12 MB/s (Full Speed), CRC and buffers on the endpoints

**SWP** single wire protocol - communication between SIM and NFC controller concurrently with the regular I/O, data sent with voltage modulation and returned with current modulation- full duplex,

**timer** - a 16 bit counter (or 16 bit), used for timeout detection, watchdog for security reasons

**CRC** cyclic redundancy clock, Reed Solomon codes,

- $x = x_1...x_k$ is the sequence to be encoded

- $p_x(a) = \sum_{i=1}^{k} x_i a^{i-1}$ polynomial over some finite field

- $c(x) = p_x(a_1)p_x(a_2)...p_x(a_n)$ is the code of $x$, where $a_i$ is the $i$th power of the root of degree $n$.

- $C(x) = x \cdot A$, Vandermode matrix, row 1: $1....1$, row 2: $a_1, .... a_n$, row 3: $a_1^2, ..., a_n^2$, and so on

- properties: distance between the codewords: $n - k + 1$ (this is optimal), since two polynomials of degree $k$ may have only $k - 1$ equal values

- it can correct half of it bits

**RNG** temperature etc. hard to implement,

**pragmatic solutions:** PRNG (sometimes poor),

be aware that the algorithm implemented is not original one (e.g. DSA but DSA+LFSR+...),

**PRNG:** the next value derived with the key from the previous values.

- Round Robin- eg 12 values in a buffer

- testing - NIST tests, good for excluding biased/faulty generators, no security guarantees

- hardware Trojans: faults in the circuitry that are not changing the layout-wires, but e.g. the number of electrons in the substrate (invisible during the audit, but may be used to "break randomness" if the manufacturer knows what are the faulty places

**Clock multiplication:** external clock cannot have frequency over 5MHz. Internally we can increase it a few times with a multiplication circuit. Potentially: one could adjust the speed to adjust energy usage (problems with intereference of oscillators with the GSM, UMTS communication)

**MMU**: memory management unit for monitoring boundaries between the application programs (strict separation). must be tailored to the opearting system of the chip

**JAVA accelerator:** approaches 1) dedicated hardware component, high speed but takes place, 2) native instructions for java

**Symmetric crypto coprocessor:** 75 microseconds per DES, 150 per 3DES

**asymmetric coprocessor:**

RSA up to 2048, problematic key generation, probabilistic time

EC 160-256 bits: create DSA, random numbers problematic

hash functions: SHA, Keccak, SHA1 in PL

memory for keys:

- masterkey, derived keys, dynamic keys (session)

- PIN: master or deriving from master key, PIN updates in different memory, problems of nonuniformity of PIN (no leading zeroes, etc), subclasses where strategy gives higher chances

# DATA ENCODING

Abstract Syntax notation, ASN.1: primitive types (boolean, integer, octet string, bitstring), constructed data types,

encoded via TLV structures: (Tag, Length,Value), tags for frequently used data types are in a standard,

BER -Basic Encoding Rules, Distinguished Encoding Rules – subset of BER

Some details:

- tag:1-2 bytes, the first byte: b8, b7 define the class: universal, application, context-specific, private class, b6: data object primitive or constructed, b5-b1: tag code, if all ones then the second byte specifies the tag code

- Length: 1-4 bytes:

    - 1 byte: 00 to 7F: encode length 0-127

    - 2 bytes: 1st byte 81, 2nd byte encodes length 0-255

    - 3 bytes: 1st byte 82, 2nd and 3rd bytes encode length 0-65535

    - ...

properties: not too flexible, but not too high overhead, much better than XML

**Data Compression (useful for images):**

only very simple methods used as compression/decompression procedures might require too much space and it does not pay off to compress. Two main methods:

- run-length encoding: blocks of zeroes and ones, each longer block encoded by its length after an escape symbol, short blocks encoded directly,

- encodings like Huffman encoding: variable length encoding

---

**Internal logical organization - state machines**

- after activation and reset always in the same initial state

- in case of any trouble - reset

- state machine: transitions describing the events

- standard  SDL notation:
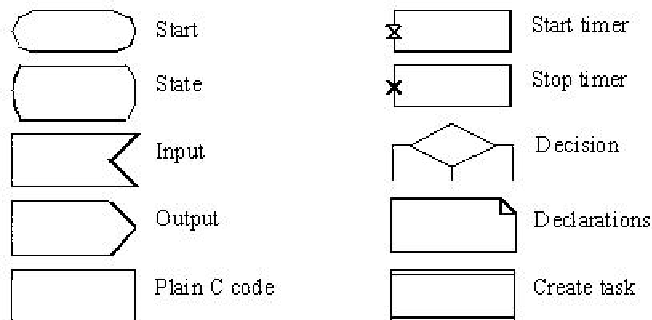
and circle as a label (instead of C code from picture simple conditions)

- try to write automaton with flow without cycles (except for reset) with a few short paths

- input consisting of (standard) instructions. Their execution automatically delivers response to the reader. Later more about instructions.

- methods to design automata with compound states

- example

---

# FILE MANAGEMENT

**general:**

- previously direct physical addressing

- still no man-machine interfaces, hexadecimal adressing

- file itself contains all information about itself in a **header**, the rest is the **body**

- header not changed frequently (problems with EEPROM, flash) , so placed in a separate page

- files allocated to memory place for security reasons, shared files sometimes via shared headers

- separation of applications via separation of directories

**lifecycle**:

1. CREATE (might be with initial data or not)

2. ACTIVATE/DEACTIVATE – activation necessary before use (only a limited number of files might be active at a time!)

3. TERMINATE - permanently blocked, its memory is permanently inaccessible (secure option)

4. or: DELETE - memory recovered (insecure)

**Types:**

MF- Master file, main directory, implicitly selected after activation

DF - Dedicated Files, directories, hierarchy, always not deep (1-2 levels)

ADF - application dedicated files - not below MF

EF - elementary file, data necessary for application

Internal EF - hidden files used by operating system. They cannot be selected for applications

**File names**

logical names due to transferability of programs

- File Identifier (FID), 2 bytes, some reserved, e.g. 3F00 is MF

- MF: FID

- DF: FID, DF name (16 bytes, normally bytes 5-16 are AID - application ID: 5 bytes RID (registered identifier: A-international registrations, D-national registration, country code, application provider number)

- EF: FID, SFI (short FID) - to be used in instructions

**File selection**

one opened for I/O

SELECT (explicitely) or implicit (by READ ..., UPDATE, ...)

**File structures**

**transparent**: a single sequence of characters, no internal structure, reading file contents by specifying offset, instructions: READ BINARY, WRITE BINARY, UPDATE BINARY

**linear fixed:** equal length records, reading by specyfing record number, instructions: READ RECORD, WRITE RECORD, UPDATE RECORD

**linear variable:** frequently not supported, should be avoided, the same instructions as for linear fixed, but execution more complicated

**cyclic:** a fixed number of records of a fixed size, access to first last, previous or next record

**PUT DATA, GET DATA:** for TLV objects

**Access conditions:**

- defined at file creation and then almost always unchanged

- state oriented (current state is compared with the file access information) or command oriented (which commands have to be executed first)

- global security state (card, state of MF) and local security state (state of a file)

---

# STANDARD OPERATIONS

- each card implements some set, but unnecessary commands are removed

- different sets: general, payment cards, telecommunication cards, over 20 standards, important is Global Platform Specification, Common Electronic Purse Specification (CEPS)

- arguments, return values

**File commands**:

CREATE FILE

ACTIVATE FILE

DEACTIVATE FILE

TERMINATE EF

TERMINATE DF

DELETE FILE

TERMINATE CARD USAGE

SELECT - selects a file, or an upper DF

STATUS - shows selected file and it properties in return, otherwise does not do anything

READ BINARY (number of bytes, offset) return: data, return code

WRITE BINARY ... this is not a write instruction but logical AND

UPDATE BINARY - this is the write operation in the standard meaning

ERASE BINARY (not necessarily physical erase)

READ RECORD, WRITE RECORD (like WRITE BINARY), UPDATE RECORD, APPEND RECORD (adding a new record at the end)

PUT DATA, GET DATA - direct access to TLV objects

SEEK (length of data, pattern, offset, mode (forward from the start, backwards from the start, forward from the next record, backwards from the previous record) returns record number

SEARCH - another standard, file can be checked

SEARCH BINARY- the same for transparent files

INCREASE, DECREASE – for cyclic files, operations on a counter (changing the current position)

EXECUTE – starts executable EF (some operating systems enable this)

**PIN commands:**

VERIFY PIN, VERIFY CHV – PIN verification, (CHV = Card holder Verification) , switch: global PIN or application specific (each application may have its own PIN)

CHANGE CHV – reset PIN

RESET RETRY COUNTER – with PUK, counter like in ATM cards enable only a limited number of PIN trials

UNBLOCK CHV (using PUK)

ENABLE VERIFICATION REQUIREMENT – e.g. GSM for PIN with SIM cards

DISABLE VERIFICATION REQUIREMENT – switch off the PIN verification

**Security operations:**

GET CHALLENGE - returns a random number

INTERNAL AUTHENTICATE

- sends random number, key number

- return: $X_{\mathrm{ICC}} := E(\mathrm{key}, \mathrm{random})$

- terminal tests the result (showing that the terminal holds the symmetric key)

EXTERNAL AUTHENTICATE

- random number $R$ sent to the terminal

- $E_K(R)$ sent to card, the card checks whether the result is correct (the key $K$ is shared with the terminal)

MUTUAL AUTHENTICATE

- GET DATA (data about chip number)

- GET CHALLENGE – $RND_{ICC}$ transfered to terminal

- authentication

a) terminal generates $RND_{IFD}$, computes $X_{IFD} = E_K(RND_{IFD}, RND_{ICC}, \text{chip number})$

b) card decrypts, checks if $RND_{ICC}$ is present there. If no then aborts.

c) the card sends $X_{IFD} = E_K(RND_{ICC}, RND_{IFD}, \text{chip number})$

d) terminal decrypts and checks the plaintext $RND_{ICC}, RND_{IFD}, \text{chip number}$

GENNERAL AUTHENTICATE (for the use e.g. with PACE)

PERFORM SECURITY OPERATION

option COMPUTE CRYPTOGRAPHIC CHECKSUM - computes a cryptographic MAC of a file

option VERIFY CRYPTOGRAPHIC CHECKSUM - checks cryptographic MAC of a file

option ENCIPHER – encrypts the data, algorithm and mode determined first by MANAGE SECURITY ENVIRONMENT

option DECIPHER - returns decrypted data

option HASH  - returns hash of data, a switch available for performing only the last part of hash computation (for efficiency reasons)

option COMPUTE DIGITAL SIGANTURE

option VERIFY DIGITAL SIGNATURE

option VERIFY CERTIFICATE

option GENERATE

**Global Platform:**

LOAD – load data

INSTALL – specifies among others the size of volatile and nonvolatile memory reserved

**Hardware test**

TEST RAM

TEST NVM  – testing non volatile memory, given area rewriten many times with a pattern

COMPARE NVM – reading NVM and checking if the given pattern really retained there

DELETE NVM – clears a given area of NVM

**Electronic Purse commands**

 INITIALIZE IEP for Load– options:

 load amount, currency code , PPSAM descriptor, random number, user defined data

response: provider id (PPIEP), IEP identifier, cryptoalbo used, expiry date, purse balance, IEP transaction number, key information, signature, return code

CREDIT IEP

load: information on key to be used, signature,

response: signature, response code

INITIALIZE IEP for Purchase – the card sends data to the terminal: purse provider identifier, IEP identifier, crypto algorithm used, expiry data, purse balance, currency code, authentication mode, IEP transaction number, key information, signature

DEBIT IEP – charging the balance, parameters: PSAM identifier, PSAM transaction number amount to be debited, currency code, key info, signature, the card returns a signature of the transaction

**Credit card commands**

GET PROCESSING OPTIONS

GENERATE APPLICATION CRYPTOGRAM

command: desired application cryptogram, transaction related data

response: cryptogram information data, application transaction counter, application cryptogram, return code

**Processing times examples:**

(below data may differ for different implementations, especially for cryptographic operations)

READ BINARY 100 bytes: processing time 2ms, transfer 146 ms

UPDATE BINARY 100 bytes with erasing: processing 35ms, transfer 162 ms

EXTERNAL AUTHENTICATE: 235ms processing, 270ms data transfer

INTERNAL AUTHENTICATE: 135ms processing, 201ms data transfer

MUTUAL AUTHENTICATE: 135ms processing time, 163 ms data transfer

VERIFY PIN: 27ms processing time, 56 ms transfer time

DEBIT IEP: 235 ms processing time, 270 ms transfer

CREDIT IEP: 175 ms processing time, 222ms transfer

corollary: processing and time complexity is crucial!

─────────────────────────────────────────────────────────

# Communication

**sequence of steps:**

1. power up on card

2. card sends ATR, smart card in a sleep mode

3. smart cards obtains APDU and changes to active mode

4. response

points 3 and 4 executed in a cycle

**ATR** "answer to reset", sent on I/O line, max 33 bytes, usually a few bytes, transmission "divisor rate" the same for all cards, start must occur in a time window (400-40000 clock cycles - eg. up to 8.14 ms for 4.9153MHz frequency of the signal), if not then tries 2 more times

TS - initial character, German: „00111011", French: „00111111" , start with 1, measures time of elementary time unit: the time between two first falling edges of TS and divides by 3

T0 format character - which interface characters transmitted afterwards in ATR (a few options possible - see below)

interface characters: TA1, TB1, TC1, TA2 ... interface charcters defining basic parameters of transmission, like guard time, divisor, etc.

- TA1: initial $etu = \frac{372}{f} \cdot sec$, where $f$ is the frequency used

  work $etu = \frac{F}{D} \cdot \frac{1}{f}$ sec, where $F$ is the rate conversion factor, D= bit rate adjustment factor (used due to variations of conditions during operation), the standard describes an encoding for a few combinations of values

- TAi for $i > 1$ define supply voltage and parameters of the clock

T1, ... Historical characters – chaos, many informations on smart cards, OS, ..

TCK check character (error detection code):

- protocol T=0: not sent as there is bytewise error detection

- protocol T=1: XOR checksum for all bytes starting from T0 up to the TCK

**PPS (protocol Parameter Selection)**

Some cards allow changing parameters of the transmissions – important for SIM cards, USIM cards.

- there is a fixed collection of possibilities

- some of the configurations are not specified ("for future use")

**APDU application protocol data unit**

APDU is the only way of communication with the card.

- fixed format, quite compressed

- initiated by command APDU - sent from the terminal to the card, the answer is response APDU

**command APDU**

- header:

  - CLA (class byte, e.g. GSM denote as 'A0') it may denote credit card, electronic purse, private use, ...

  - INS instruction byte

  - P1, P2: two parameters, (no more possible, the meaning defined by the standard)

- body:

  - Lc field (length of data transmitted)

  - data field

  - Le field (length of expected response)

**response APDU**

- data comes first (length is defined by Le from Command APDU)

- afterwards the status bytes SW1 and SW1. The following options for the status: process completed (normal, warning), process aborted (exec. error, checking error)

**Secure messaging**

the goal: securing the communication with the card: authentication, or even confidentiality

this is a problem even for contact interface as I/O contact can be traced

- **authentic mode procedure:**

  - the CCS (cryptographic checksum) computed, e.g. with AES,

  - the input is the original APDU+some padding (necessary to have a full number of blocks),

  - the out is the old APDU in plaintext encoded as TLV sturcture plus a TLV encoded CCS

  - Warning: there is no confidentiality!

- **combined mode procedure:**

  - CCS computed similarly as for authentic mode

  - then the resulting APDU encrypted: but only all TLV objects,

  - the result is APDu with CLA, INS, P1,P2 unencrypted, then TLV encoded cyphertext in the data field

  - Problem: the command and parameters are still unencrypted! Information leaked

- A solution: using a command ENVELOPE which says the card to decode the ciphertext and find there the real command to execute

- the response APDU also contains encrypted data

**send sequence counter**

also a security mechanism: as APDU might be undelivered e.g. by jamming the radio channel

- started with a random value during a communication session

- then appended at each round

- there are two methods of encoding the counter:

  - as a separate data field

  - the counter is XOR with some portion of APDU before computing CCS – advantage: if the recipient knows the expected value of the counter, then he can easily recompute. Advantage is that no extra communication space required.

**Logical channels**

application may run in parallel on the card, no interleaving between request-response, APDU specifies the logical channel - to which application the communication belongs

---

# Data transmission with contacts

**Physical layer**

communication only digital, 0V as reference level, the other level is +5V, or +3V, or +1.8V

conventions: 0V represents 0 (direct convention), or 1 (inverse convention)

based on RS232 (0V used instead of negative voltage)

I/O line is in high level always when the data is not sent

serial communication

asynchronous: so the sender adds the start bit at the beginning of each transmission (low voltage), 8 data bits, followed by the parity bit, and stop bits (used as guard time)

transmission speed: frequency/divider, divider says how many periods of the carrier frquency needed to encode a bit. Typically divider is 372 and 512, 5MHZ max frequency, 32 minimum usabe divider so at most 156 250 bit/s

**Memory cards**

simplest protocols, variety,

prepaid telephone cards

**ISO** protocols (15 versions described), mosty used T=0, T=1

## T=0

- France, early standard, simple, minimal memory usage

- used for GSM worldwide

- byte oriented

- command structure: CLA, INS,P1,P2,P3, data field, P3 specifies the length of data field

error handling:

- retransmission of byte immediately if an error detected (not after a block of bytes)

- reporting error I/O line down after byte transmission, in halfway of etu (so in a "wrong place"), during the guard time

- this mechanism technically problematic sometimes, since in some cases below 1 etu nothing is detectable

- alternative mechanism: add 1 to the received command byte as confirmation (note that the comand bytes are coded by even numbers) - becoming technically obsolete solution

SM hard - many versions (with overhead)

interpretation of an APDU by a relatively simple state machine


# T=1 protocol

- asynchronous

- half-duplex (both directions communication but not at the same time)

- block oriented (a block is the minimal unit to be transmitted)

- follows layer concept of communication protocols (data for higher leyers sent transparently by data link layer)

Block:

- contains application data and control data of the protocol

- structure of a block:

  - prologue field: 1 byte NAD(node address- destination and source), 1 byte PCB (protocol control block - e.g. sending sequence number(mod 2)), 1 byte LEN (length)

  - information field: APDU

  - epilogue field: EDC . It contains either a CRC error detection code from ISO or just LCR - XOR of all bytes (faster, easier but less usable as error detection)

- send/receive counter: binary

- parameters for transmission to avoid deadlocks:

  - $\rightarrow$ CWT -character waiting time, in order to avoid deadlock, (time interval between the leading edges of characters), CWT $= 2^{CWI} + 11$ etu, CWI parameter is taken from ATR, default CWI=13

$\rightarrow$ BWT block waiting time: between leading edge of XOR in the epilogue field and leading edge of NAD in the response of the card

$$\text{BWT} = 2^{\text{BWI}} \cdot 960 \cdot \tfrac{372}{f} \sec + 11 \, \text{etu}$$

for the default BWI=4 it means $\approx 1.6 \sec$

$\rightarrow$ BGT - block guard time: time between communications in opposite directions

- waiting times can be changed during protocol execution

- block chaining: if block are larger than the buffer of the sender or receiver, then the data field is splitted and a chain of blocks is used. a flag "more" is used to indicate a chain

- error handling:

  - sophisticated mechanism

    - error -> the sender receives an R block indicating an error $\Rightarrow$ retransmission of the last block

    - if retransmission ok, then resynchronisation with an S block, card acknowledges and the counters (on both sides) reset to 0

    - if a block cannot be sent then RESET and the whole session terminated, data transmitted so far is lost

Other protocols with potentially high importance:

**USB** (Universal Serial Bus)

- T=0, T=1 are no for slow data transmission,

- target: 12Mbit/s USB communication

- the main problem is time synchronization, USB assumes stability of frequency on both sides, but the smartcard has frequency from the reader – this is unstable. Extra circuit on the smart card required

- problem also with the code: USB requires a few KB of code for interface on card

- USB is encoding bits via difference of electrical levels on a pair of signal lines (more reliable than the encoding on a line), C4 and C8 used for USB (AUX1 and AUX2 conatacts, normally unused)

- encoding method NRZI (nonreturn to zero) encoding: a 0 encoded by polarity reversal, for 1 polarity is unchanged,

- clock for the bus retreived from the polarity reversals, what to do if there is a long block of 1's? after six ones obligatory bit stuffer - a zero.

- logical connection: 4bit "endpoints". EP0 for control, other endpoints are unidirectional

- transfer modes:

  - control transfers: initiated on EP0, acknowledged on both directions,

- interrupt transfers for small amounts of data – not really an interrupt, since the time periods for interrupt transfers are fixed

- bulk transfers: non-time critical for large amounts of data

- isochronous transfers: for time critical transfers

- least significant bit sent first (little endian), data sent in frames at fixed time intervals (about 1ms), a frame starts with a SOF (start of frame) packet containing a packet identifier, frame number and CRC code

**SWP** (Single Wire Protocol)

- NFC controller has to find a way to communicate with the SIM card while the card is communicating with the telephone with another channel

- C6 contact used as a single line (previous usage as external programming voltage for EEPROM not needed anymore)

- full duplex

- complicated encoding at the electrical level (card is always the slave):

  - S1 (master-to-slave): high voltage at 75% time – a 1, hogh voltage at 25% of time - a 0

  - S2 (slave to master): influences the voltage at the high voltage period of S1: 0 by reducing the voltage, 1 by keeping it high

————————————————————————————————————————————————-

# Contactless transmission

flexible batteries - expensive, so energy must come together with communication

**INDUCTIVE COUPLING:**

range depends on power consumption inside: no computation on card, then range about 1m, with writing on the card- range 10cm to enable enough energy, 1m – few tens of microwats, writing already 100 microwats, processor already 10miliwat, legal restrictions on energy sent by readers

**wavelength** 2400m or 22m depending on frequency: much smaller than the distance between the card and reader: physical transformer model can be used

**voltage**: rectified on the chip, proportional to the signal frequency, depends on the number of turns in the coil and the coil area, strong current needed on coils

picture page 286

**transmission back:** (small) fluctuations of current on the side of smart card, in order to be able to separate from the strong signal form the reader: a different frequency for modulation is used, the reader applies a filter to separate them

picture page 287

**CAPACITIVE COUPLING**:

by putting very close (on surface), coupling area both on terminal and on the card, two fields on card, two on reader, too low energy to run microprocessor directly so inductive coupling used anyway for providing energy

**Collision avoidance:**

collisions between cards: possible as a single radio channel.

Methods of separation:

- space division multiple access

- time division multiple access - TDMA

- frequency division multiple access - FDMA

- code division multiple access - CDMA – codes on the same frequency. Decoding: searching how the transmission received could be composed of individual transmissions (main idea of UMTS)

## CLOSE COUPLING  CICC

up to 1cm – slot or surface operation, must tolerate electrostatic discharge,

**power transmission**: via inductive coupling, power frequency 4.9 MHZ,

two pairs H1,H1 and H3, H4 have phase shift of 90 degrees,

two pairs to make it resistant to card position

**data transmission:**

- terminal to card (PSK phase shift keying): 4 alternating magnetic fields, all 4 fields shifted by $90^o$, to change a bit all four fields shift simultaneously by $90^o$, at the beginning a 1 is always sent picture page 294

- data transmission from the card to the terminal: modulation at 307.2kHz, change of bit by shift of phase by $180^o$ of the subcarrier.  picture page 294

 **Capacitive data transmission**

- timing constraints: how much time when power off, when power rises, for the first transmission to the card, for the first transmission from the card

## PROXIMITY COUPLING  ISO 14443

- normal operation up to 10cm, large antennas for signal detection from bigger distances

- inductive coupling, magnetic field strength defined to be in some interval

- transmission frequency about $f_c = 13$ Mhz

- many different applications

- communication interfaces: type A, type B (terminals periodically perform both just to be able to talk with both types of cards)

## type A:

- initialization: bitrate $f_c/128$, later a different bitrate chosen: $f_c/k$ where $k = 128, 64, 32, 16$

so the bitrate is 106kbits/s or 212 or 424 or 847

**communication from terminal to card**:

- amplitude modulation - with signal going to zero (short interrupts)

- small interrupts (3 $\mu$s) $\Rightarrow$ no energy transmitted to the card but the card survives operating

- encoding: modified Miller encoding:

  - $\rightarrow$ transmitting a 1: pause after half bit interval

  - $\rightarrow$ transmitting a 0: no pause

  - $\rightarrow$ transmitting more zeroes: a pause between two consecutive zeroes, the first 0 has a pause before

  - $\rightarrow$ start: a pause

  - $\rightarrow$ example: | 0 | 1 | 0 | 0 | 1 |

    □■■■■■□■■■■■□■■■■■□■

  - $\rightarrow$ end of message logical zero 0 followed by one bit period with no pause

**from card to terminal**

- load modulation with a subcarrier

- transmitting a 1: carrier modulate by subcarrier in the first half of the bit interval

- transmitting a 0: carrier modulate by subcarrier in the second half of the bit interval

- start of a message: carrier modulated by the subcarrier in the first half of the bit interval

- end of the message: no modulation by a subcarrier

fig. 10.21 page 302

## type B:

- no interruptions of energy supply

**terminal to card:**

- subcarrier modulation

- modulation index 10% (slight changes of the amplitude)

- encoding: NRZ (non-return-zero), simply a higher amplitude means a 1 and the lower one means a 0.

**card to terminal:**

- modulation with BSPK (binary phase shift keying)- shift of $180^0$

- no change of the character = no pahse shift, if there is a change then a phase shift,

- initially: subcarrier with no phase shifting for some period of time

- the first character is understood to be a logical 1

figure page 304

**Anticollision mechanism**

two cards communicating with the reader during initialization phase

**type A**

- Manchester encoding used to ensure collision detection:

    - subcarrier modulated in the half of the interval

    - "first high- then low" or "first low-then high" to encode two bit values

    - if the signals are synchronous than overlapping produces a "high-high" and an evidence of collision

- all cards are transmitting synchronously their IDs

- after Reset the card in the Idle state and can only answer the Type A Request (REQA) and Type A Wake-up (WUPA),

- then the cards send synchronously Answer to Request type A (ATQA)

- the terminal receives the responses and knows that at least one card is present

- message format ATQA: the first part for the reader the second part for the cards,

  in the first part bytes of address specified, in the second part all matching cards respond.

- The length of the parts vary, but the total length is the always the same. The first part has at least 16 bits and no more than 55 bits.

- ANTYCOLLISION or SELECT: the message specifies UID (card ID - either random for the session or a fixed one) and the number of bits relevant. Only the cards that match the UID (on a given portion), reply

- step by step until only one card responds

**type B**

- predefined time slots

- dynamic slotted Aloha, number of slots in the REQB

- all cards Idle until REQB, then choose a random slot to send its data

# SIM Cards and telecommunication systems

Problems to be solved:

- how to authenticate the subscribers?

- how to protect the communication confidentiality?

- how to know the position of the subscriber?

- distributing security functions?

- how to work in roaming with partially trusted partners?

- interoperability

**GSM**

- suscriber ID: IMEI (international mobile equipment identity) - ID of a phone, IMSI-international mobile subscriber information - ID of a user (never transmitted in clear), TMSI - temporary mobile subscriber identity – temporary and related to local area

- SIM card (Subscriber Identity module):

    - dedicated commands

    - storage for some user data

    - user management

    - security functions: authentication, encryption

- key management:

    a secret key Ki on the smartcard, shared with the provider, a copy available in AuC

    possible on-the-fly derivation  Ki=f(IMSI), where f is a private punction of the provider

- authentication:

    1. BSS (or MSC) generates RAND at random and sends to the mobile station

    2. mobile station: SRES:= $g$(RAND,Ki), where $g$ is a provider specific function (e.g. might be A3)

    3. SRES sent to BSS, where the result is checked

- encryption:

    1. RAND reused

2. mobile station: Kc:=$h$(RAND,Ki), where $h$ is a provider specific function (e.g. might be A8)

3. BSS: Kc:=$h$(RAND,Ki)

4. encryption with the session key Kc and a stream cipher. The key for a frame:

   $r$(TDMA frame number, Kc) where $r$ is a provider specific function (e.g. A5),

   the encrypted frame is the

   frame plaintext XOR frame key

5. the communication: down and uplink interleaved, short frame

- problems with distributing the user secret keys:

  - instead of providing the key Ki of the user, AuC distributes on demand valid the triples

    (RAND, SRES, Kc)

  - the local provider from the Visited Network does not have to implement the provider specific algorithms, or even to know them $\Rightarrow$ easy roaming and cooperation with different vendor systems

- attack (IMSI catcher device used):

  - the mobile user gets authenticated, but the network is not authenticated,

  - use a fake base station, starting authentication, accepting whatever SRES comes and switching off the encryption

  - connecting to other subscriber via a different link (via a prepaid card, not showing subscriber number)


- attack: only the links between the mobile phoine and the base station are secured, no end-to-end encryption or authentication. E.g. point-to-point microwave links used in the access network, where eavsdropping is easy

- protecting identity: TMSI used instead of IMSI after first authentication of a user in a service area

- local authentication: not all time  the triples from the AuC used, instead the sessions are linked in the Visited Network

- cloning a SIM card is a problem (the provider may create a copy of the card e.g. for eavsdropping)

UMTS

- USIM card used instead of SIM

- CDMA coding - more sophisticated and in principle better. In practice leads to problems, and LTE is simplifies a lot (a nd makes it more efficient)

- reuse of concepts: limited trust in the Visited Network, identity protection

- correcting security problems:

  - fake base station

  - encryption in the access network

  - no cleartext transmissions

- authentication tuples: (RAND, XRES, CK, IK,AUTN), XRES=expected response, CK=cipher key, IK=integrity key

- XRES=f2(RAND,Ki), CK=f3(RAND,Ki), IK=f4(RAND,Ki), AUTN=SQN$\otimes$ AK||AMF||MAC, AK=f5(RAND, Ki), MAC=f1(RAND,AMF,Ki), AMF - system parameters

- AKA (Authentication and Key Agreement protocol):

  1. service network to mobile user: RAND(i), AUTN(i) (taken from the list)

  2. mobile user computes verifies AUTN(i) (SQn must be in a correct range), recomputes XRES(i) via definition, replies with XRES(i)

  3. use CK(i) for encryption and IK(i) for integrity

- problematic syncronisation with SQN