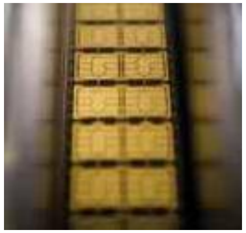
GSM Network

Carte à puce et Java Card

ATAC 2011-2012

Jean-Louis Lanet

Jean-louis.lanet@unilim.fr



Université de Limoges

SUFOP Service Universitaire de Formation Permanente

Outline

- GSM Security consideration
- GSM Architecture
- GSM Security Function
  - Authentication
  - Encryption
- GSM Standard GSM 11.11
  - Basic definition
  - Security features
  - Commands

SIM standardization group 1998

- *Billions of Calls,*
- *Millions of Subscribers,*
- *Thousands of different types of telephones,*
- *Hundreds of countries,*
- *Dozens of Manufacturers,*

…and only one Card… the SIM

GSM Security Concerns

- Operators
  - Bills right people
  - Avoid fraud
  - Protect Services
- Customers
  - Privacy
  - Anonymity
- Make a system at least secure as PSTN

GSM Security Goals

- Confidentiality and Anonymity on the radio path
- Strong client authentication to protect the operator against the billing fraud
- Prevention of operators from compromising of each others' security
  - Inadvertently
  - Competition pressure

GSM Security Design Requirements

- The security mechanism
  - MUST NOT
    - Add significant overhead on call set up
    - Increase bandwidth of the channel
    - Increase error rate
    - Add expensive complexity to the system
  - MUST
    - Cost effective scheme
  - Define security procedures
    - Generation and distribution of keys
    - Exchange information between operators
    - Confidentiality of algorithms

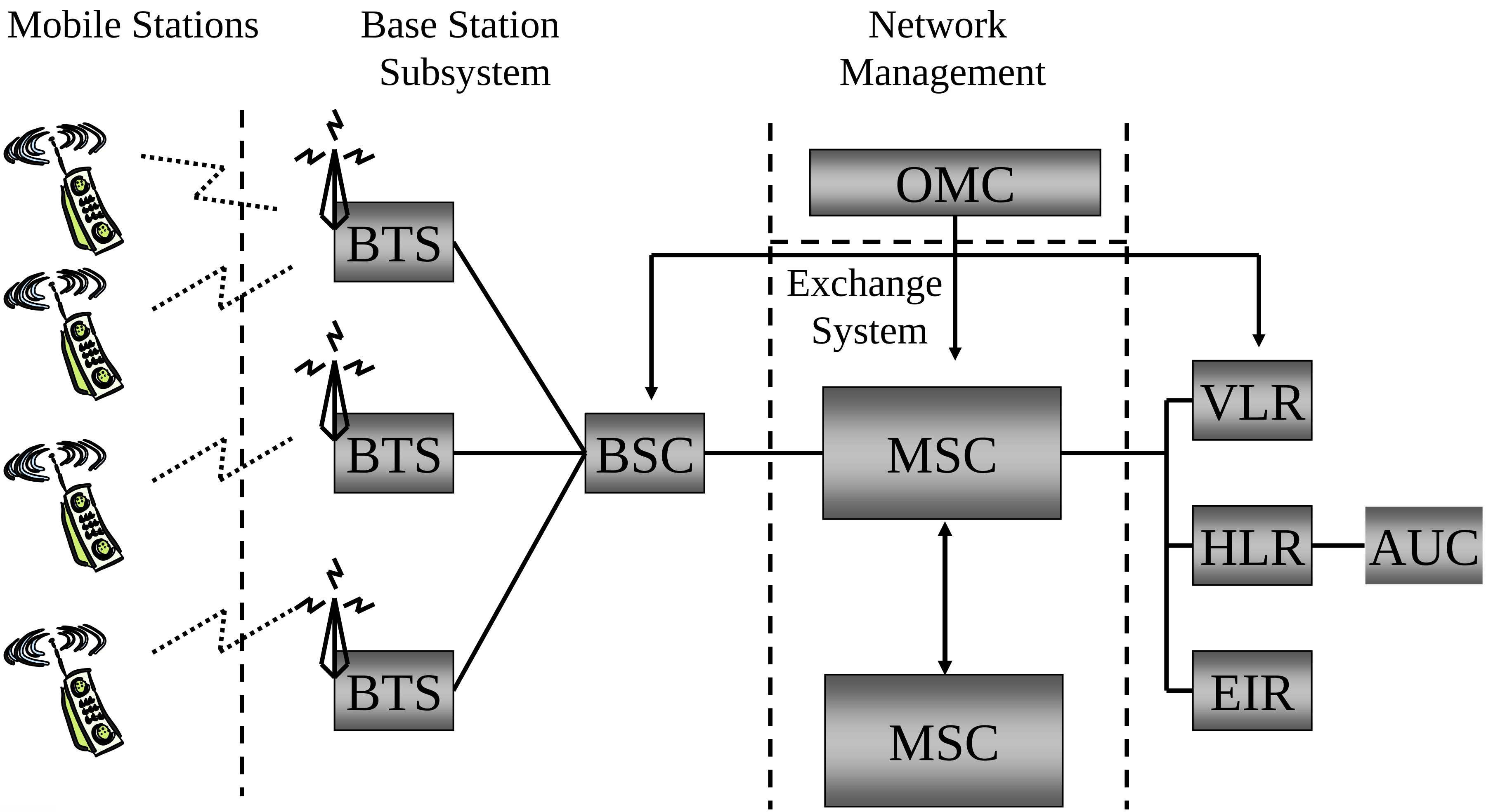
GSM Security Features

- ***Key management is independent of equipment***
  - Subscribers can change handsets without compromising security
- ***Subscriber identity protection***
  - Not easy to identify the user of the system intercepting a user data
- ***Detection of compromised equipment***
  - Detection mechanism whether a mobile device was compromised or not
- ***Subscriber authentication***
  - The operator knows for billing purposes who is using the system
- ***Signaling and user data protection***
  - Signaling and data channels are protected over the radio path

GSM Mobile Station

- Mobile Station
  - Mobile Equipment (ME)
    - Physical mobile device
    - Identifiers
      - IMEI – International Mobile Equipment Identity
  - Subscriber Identity Module (SIM)
    - Smart Card containing keys, identifiers and algorithms
    - Identifiers
      - **K_i – Subscriber Authentication Key**
      - **IMSI – International Mobile Subscriber Identity**
      - **TMSI – Temporary Mobile Subscriber Identity**
      - **MSISDN – Mobile Station International Service Digital Network**
      - **PIN – Personal Identity Number protecting a SIM**
      - **LAI – location area identity**

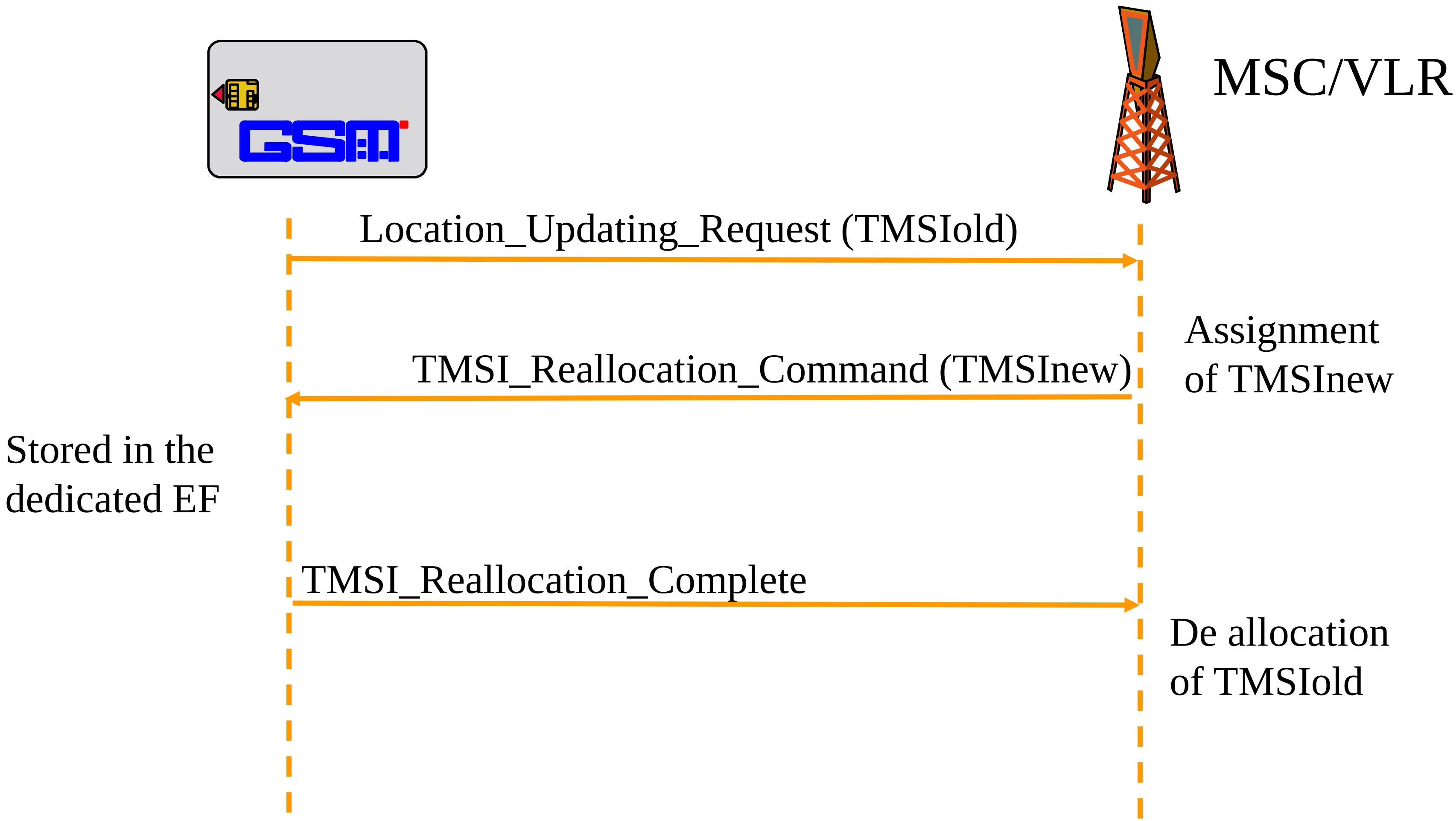
GSM Architecture



Subscriber Identity Protection

- TMSI – Temporary Mobile Subscriber Identity
  - Goals
    - TMSI is used instead of IMSI as an a temporary subscriber identifier
    - TMSI prevents an eavesdropper from identifying of subscriber
  - Usage
    - TMSI is assigned when IMSI is transmitted to AuC on the first phone switch on
    - Every time a location update (new MSC) occurs the networks assigns a new TMSI
    - TMSI is used by the MS to report to the network or during a call initialization
    - Network uses TMSI to communicate with MS
    - On MS switch off TMSI is stored on SIM card to be reused next time
  - The Visitor Location Register (VLR) performs assignment, administration and update of the TMSI

TIMSI-IMSI


MSC/VLR
Location_Updating_Request (TMSIold)
Assignment of TMSInew
TMSI_Reallocation_Command (TMSInew)
Stored in the dedicated EF
TMSI_Reallocation_Complete
De allocation of TMSIold

Key Management Scheme

- **K_i** – Subscriber Authentication Key
  - Shared 128 bit key used for authentication of subscriber by the operator
  - Key Storage
    - Subscriber's SIM (owned by operator, i.e. trusted)
    - Operator's Home Locator Register (HLR) of the subscriber's home network
- SIM can be used with different equipment

Université de Limoges
SUFOP Service Universitaire de Formation Permanente

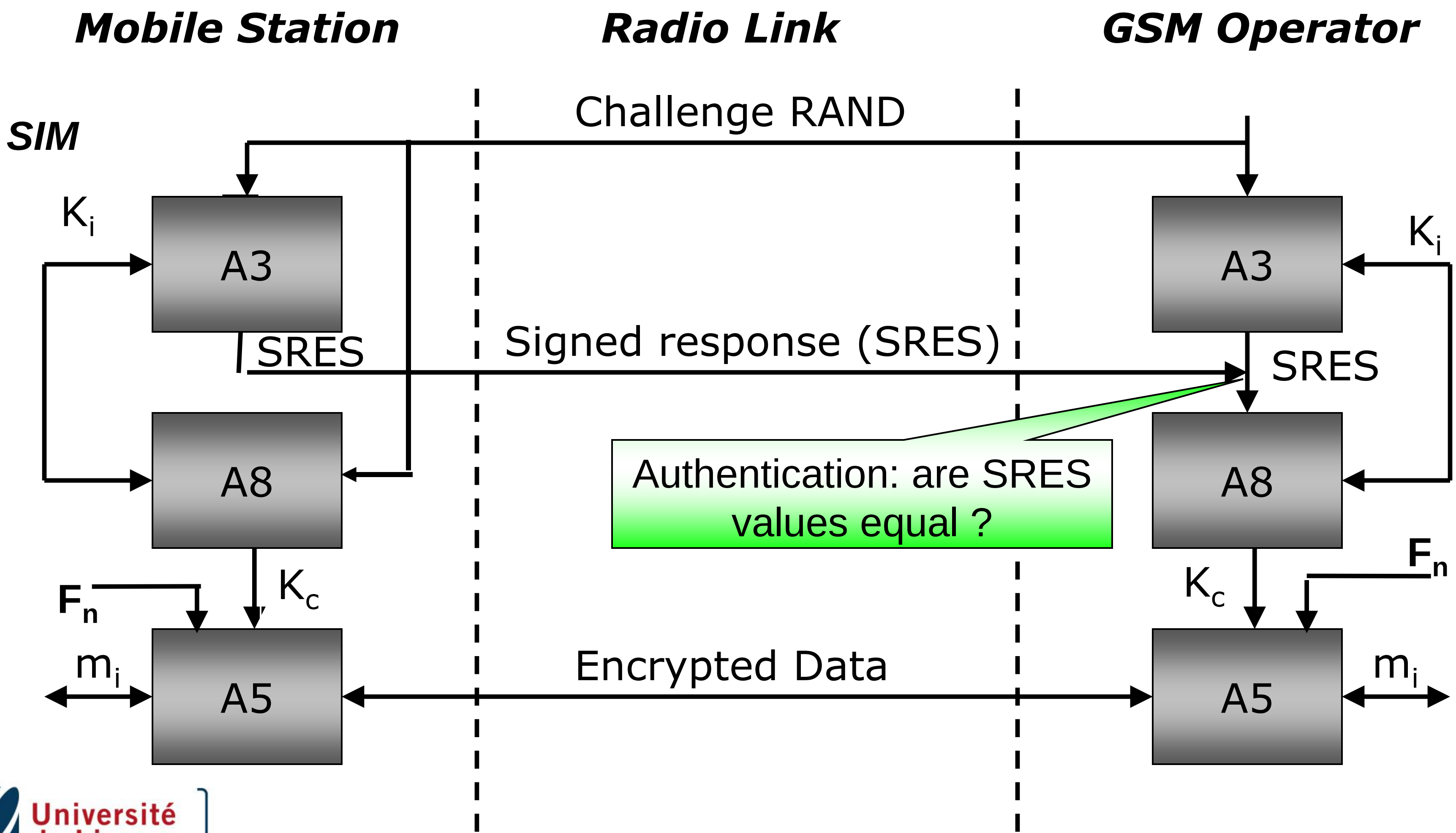
Detection of Compromised Equipment

- International Mobile Equipment Identifier (IMEI)
  - Identifier allowing to identify mobiles
  - IMEI is independent of SIM
  - Used to identify stolen or compromised equipment
- Equipment Identity Register (EIR)
  - Black list – stolen or non-type mobiles
  - White list - valid mobiles
  - Gray list – local tracking mobiles
- Central Equipment Identity Register (CEIR)
  - Approved mobile type (type approval authorities)
  - Consolidated black list (posted by operators)

Authentication

- Authentication Goals
  - Subscriber (SIM holder) authentication
  - Protection of the network against unauthorized use
  - Create a session key
- Authentication Scheme
  - Subscriber identification: IMSI or TMSI
  - Challenge-Response authentication of the subscriber by the operator
  - Unilateral Authentication
    - Counterfeit network,
    - Eavesdrop on call using a suitable piece of equipment !

Authentication and Encryption Scheme

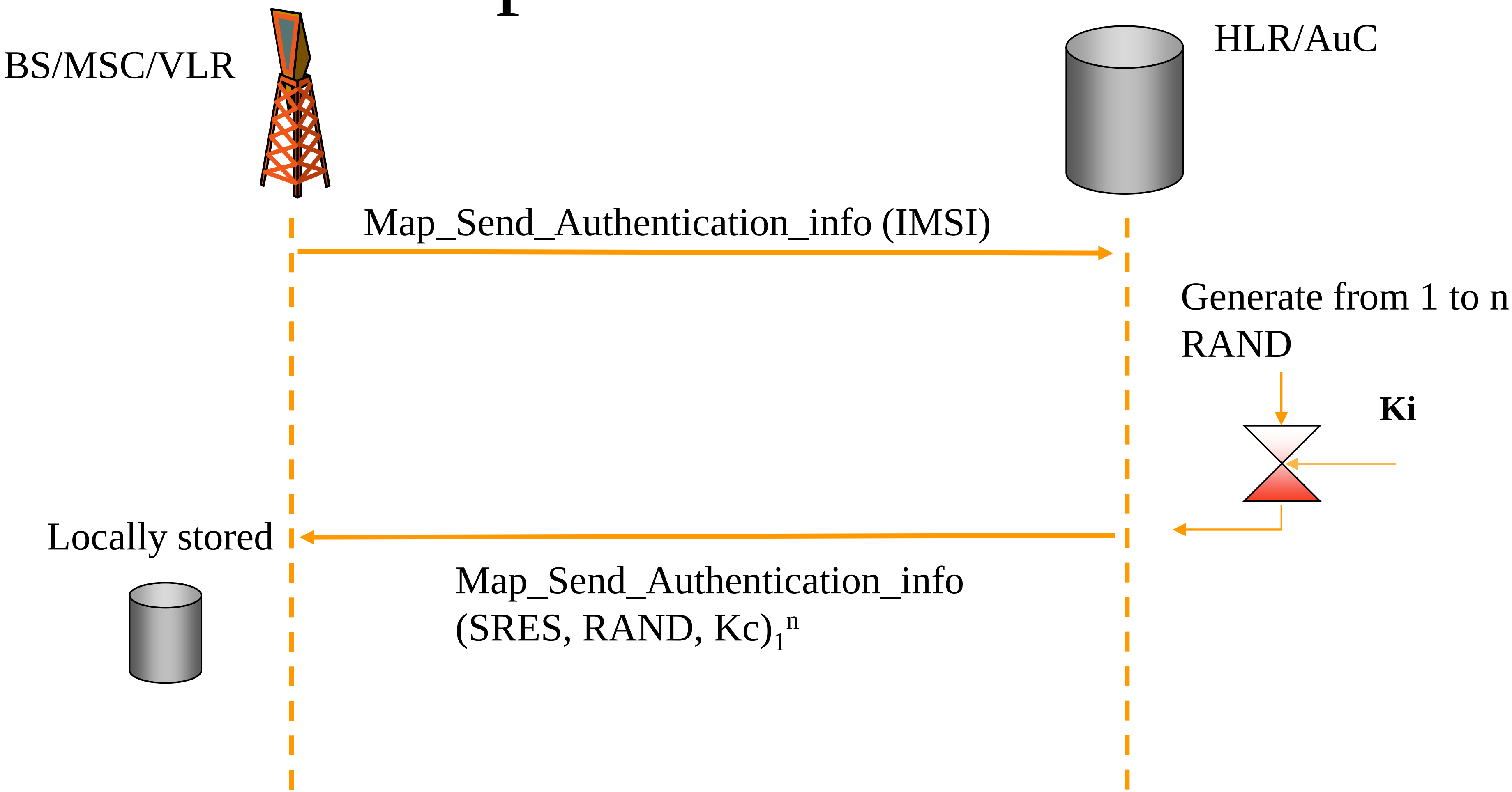


Université de Limoges
SUFOP Service Universitaire de Formation Permanente

Authentication

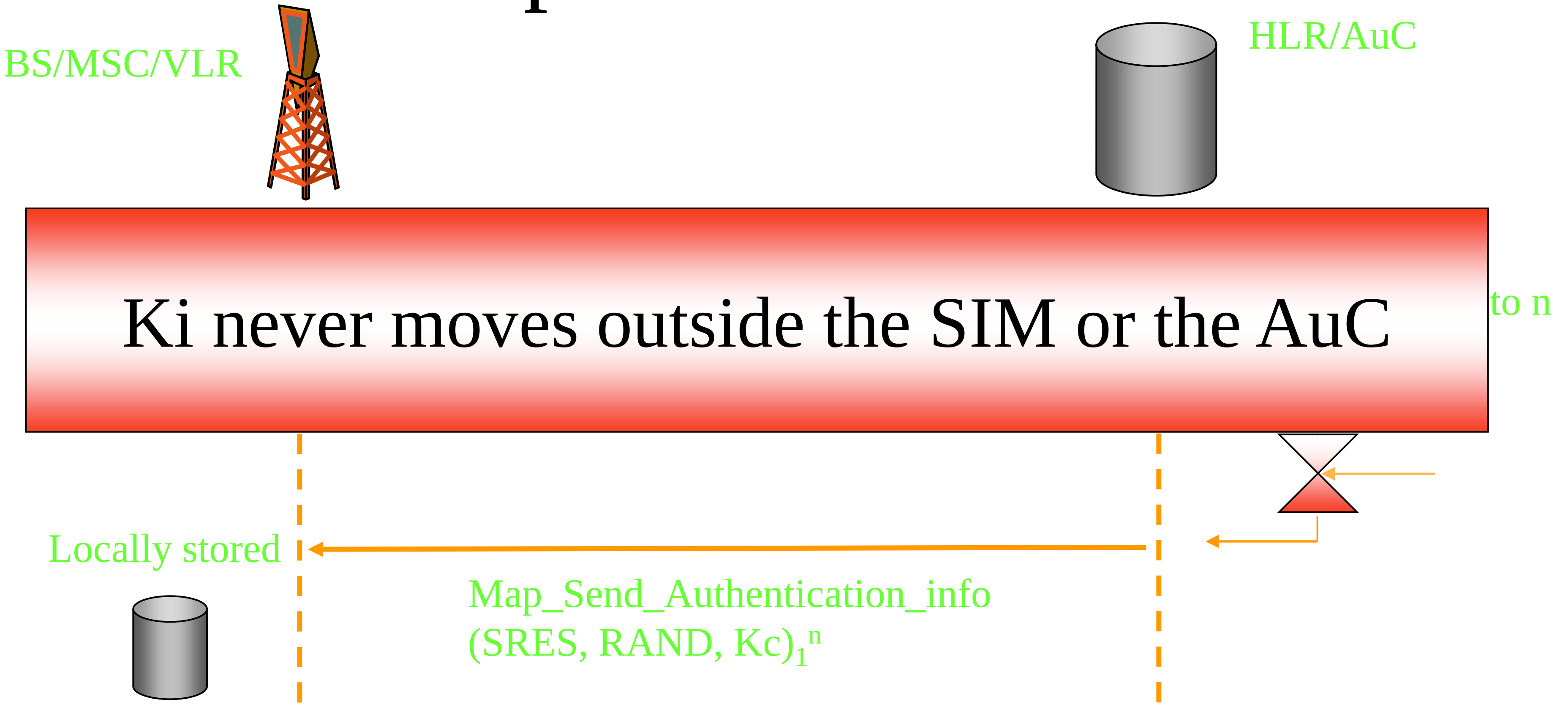
- AuC – Authentication Center
  - Provides parameters for authentication and encryption functions (RAND, SRES, K_c)
- HLR – Home Location Register
  - Provides MSC (Mobile Switching Center) with triples (RAND, SRES, K_c)
  - Handles MS location
- VLR – Visitor Location Register
  - Stores generated triples by the HLR when a subscriber is not in his home network
  - One operator doesn't have access to subscriber keys of the another operator.

Triplet Generation



Université de Limoges
SUFOP Service Universitaire de Formation Permanente

Triple Generation

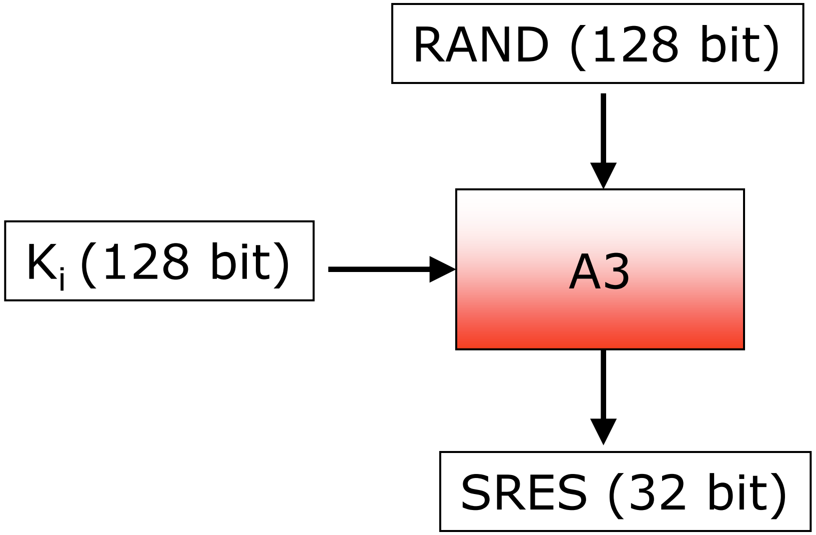


Ki never moves outside the SIM or the AuC

In case of roaming the triple is sent to the foreign network not the Key

A3 – MS Authentication Algorithm

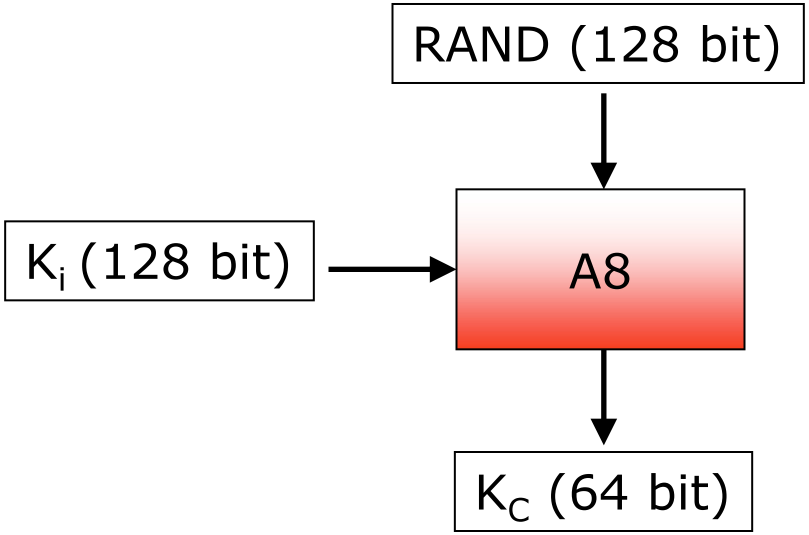
- Goal
  - Generation of SRES response to MSC's random challenge RAND



Université de Limoges
SUFOP Service Universitaire de Formation Permanente

A8 – Voice Privacy Key Generation Algorithm

- Goal
  - Generation of session key K_c
    - A8 specification was never made public

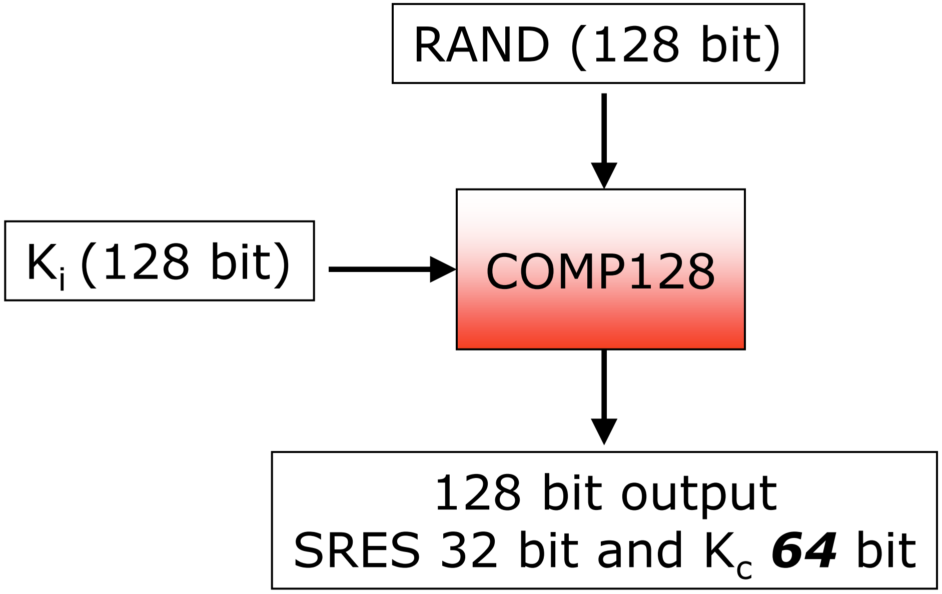


Logical Implementation of A3 and A8

- Both A3 and A8 algorithms are implemented on the SIM
  - Operator can decide, which algorithm to use.
  - Algorithms implementation is independent of hardware manufacturers and network operators.

Logical Implementation of A3 and A8

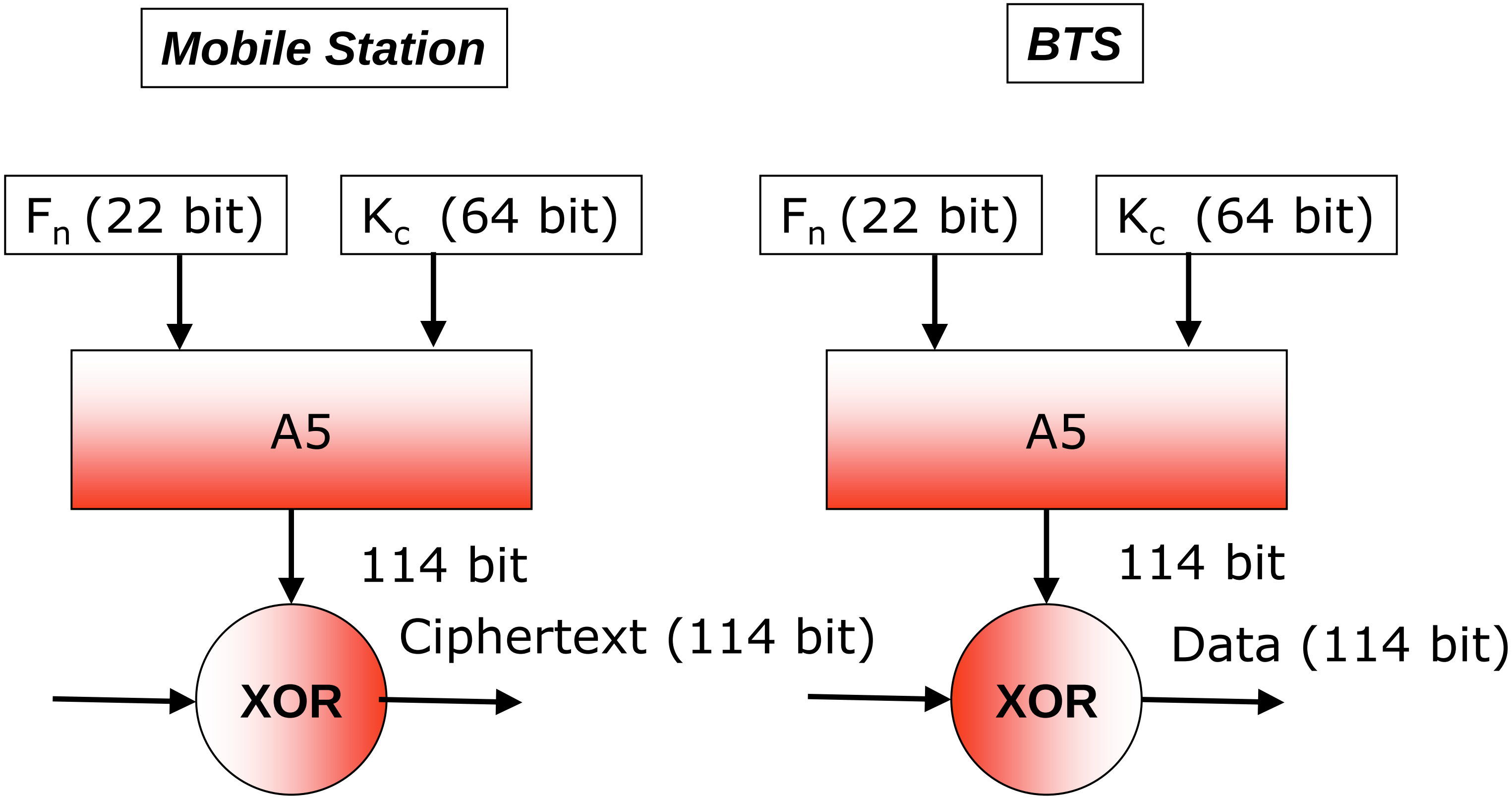
- COMP128 is used for both A3 and A8 in most GSM networks.
  - COMP128 is a keyed hash function



A5 – Encryption Algorithm

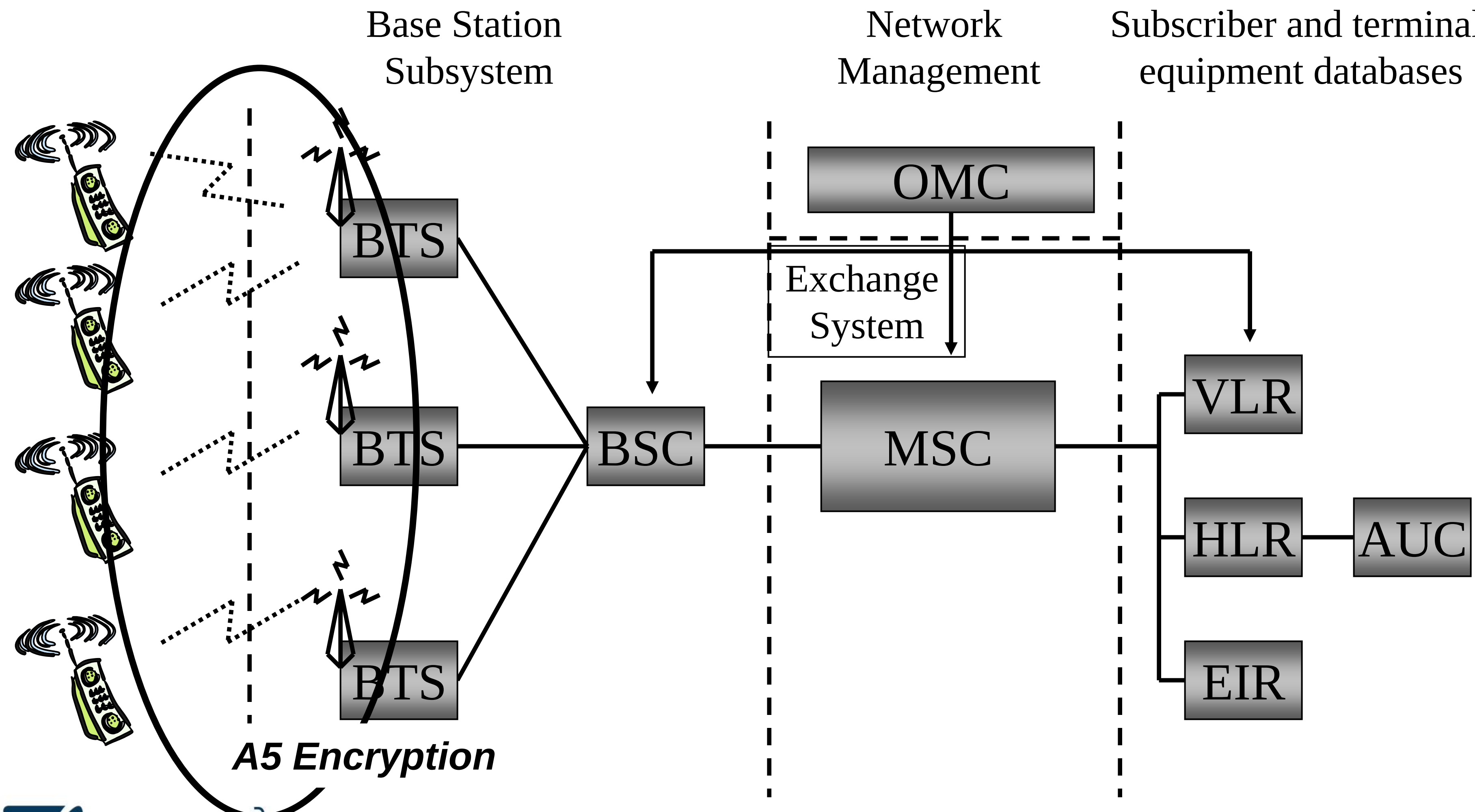
- A5 is a stream cipher
  - Implemented very efficiently on hardware
  - Design was never made public
  - Leaked to Ross Anderson and Bruce Schneier
- Variants
  - A5/1 – the strong version
  - A5/2 – the weak version
  - A5/3
    - GSM Association Security Group and 3GPP design
    - Based on Kasumi algorithm used in 3G mobile systems

Logical A5 Implementation



Real A5 output is 228 bits for both directions

A5 Encryption



Part II

Accessing the Sim application, the file system

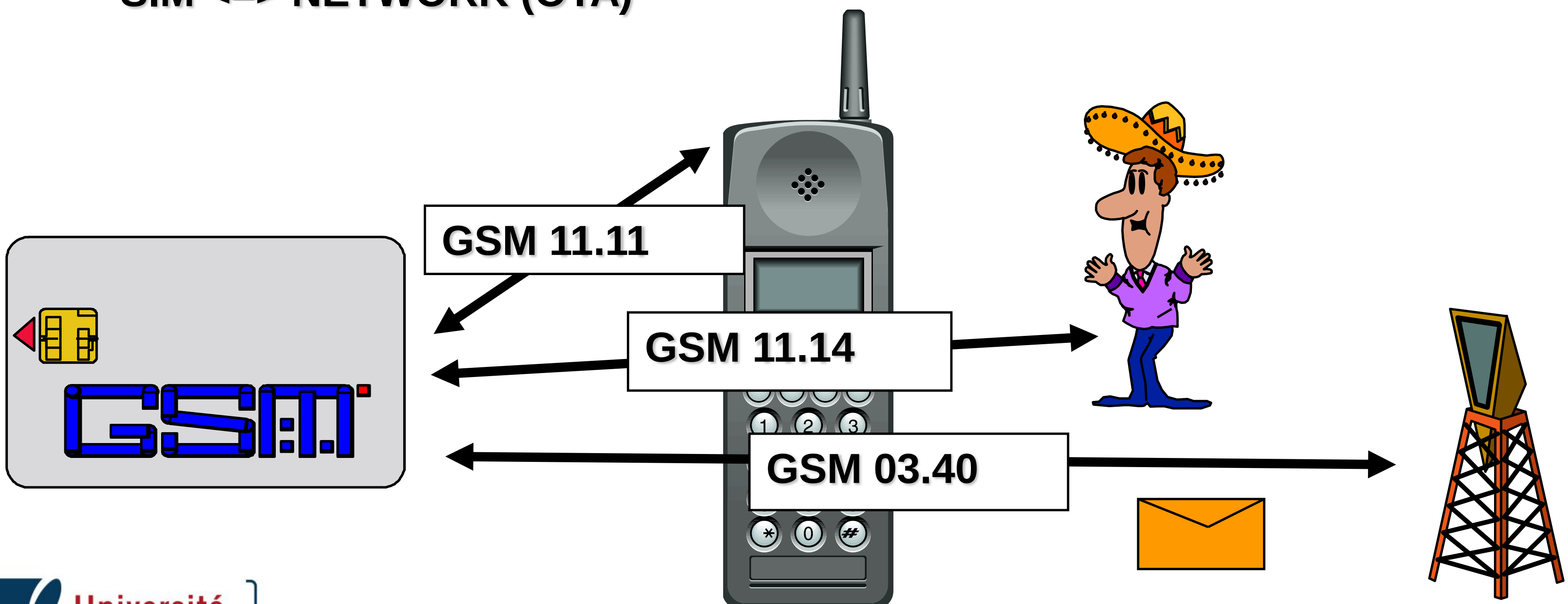
GSM standards

For coherent Communication between

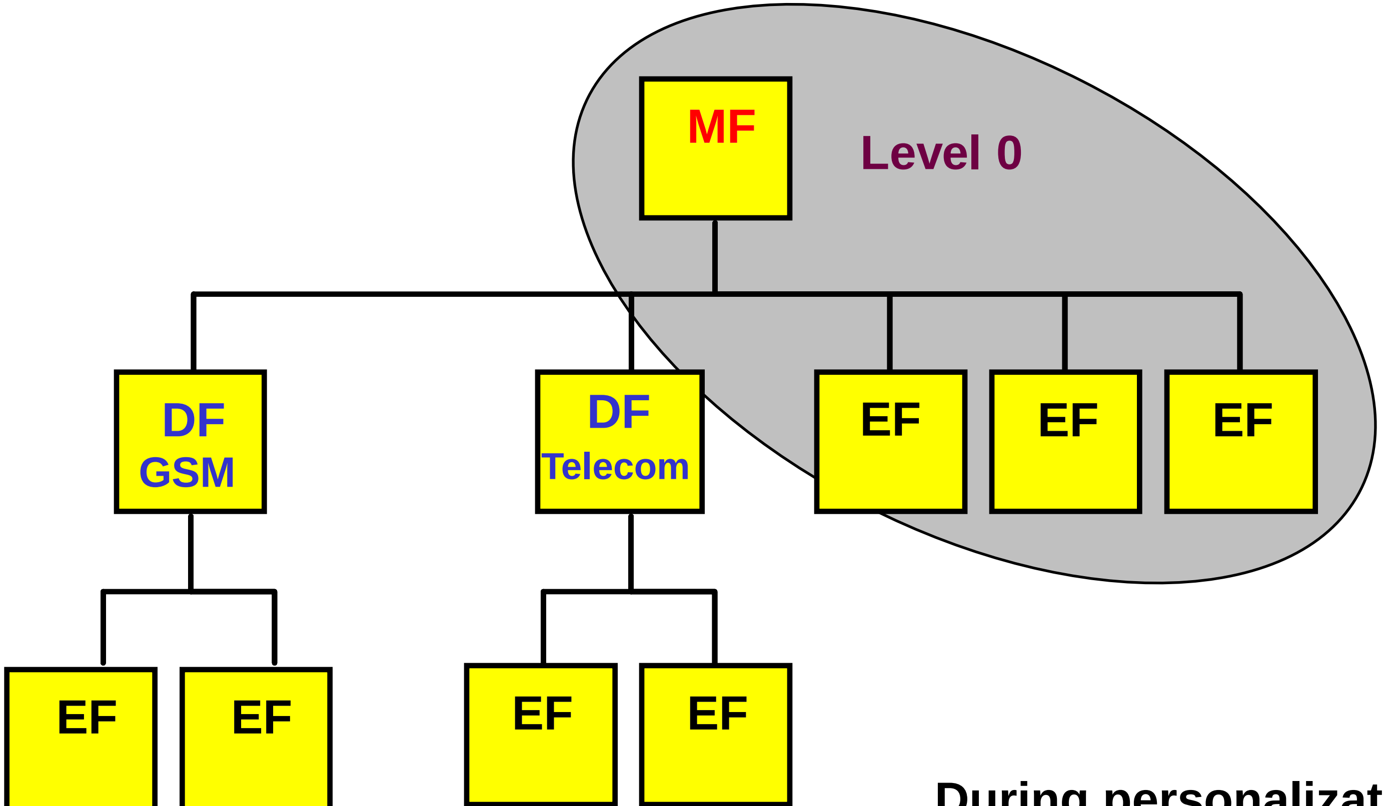
SIM <=> MOBILE

SIM <=> SUBSCRIBER

SIM <=> NETWORK (OTA)



Typical SIM card organization



GSM operator directory
Telecom user directory

During personalization
IMSI and Ki and unblocking
Key are provided
by the operator

The files

- Root directory : 3F 00
- Two main directories : GSM (DFGSM, 7F20) and TELECOM (DFTELECOM, 7F10).
- The identity is coded on two bytes, the first :
  - - '3F': Master File;
  - - '7F': 1st level Dedicated File;
  - - '5F': 2nd level Dedicated File;
  - - '2F': Elementary File under the Master File;
  - - '6F': Elementary File under a 1st level Dedicated File;
  - -'4F': Elementary File under 2nd level Dedicated File.
- After ATR (*Answer To Reset*), the master file (MF) is implicitly selected

Université de Limoges
SUFOP Service Universitaire de Formation Permanente

GSM directory

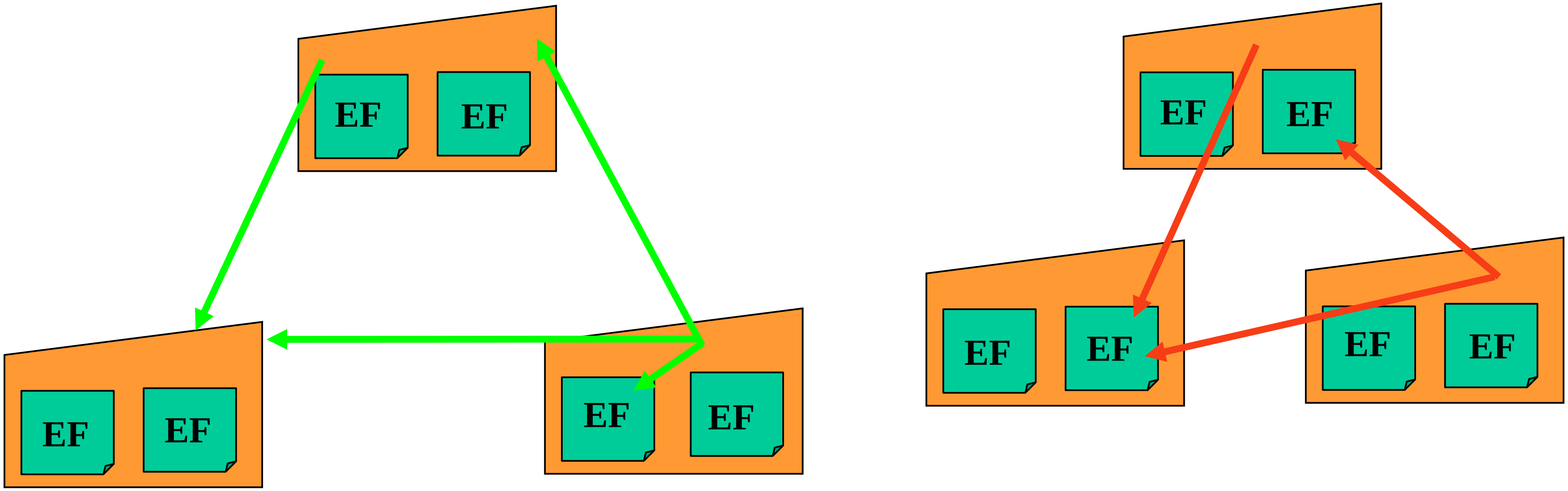
- The file EF_{IMSI} (6F07) includes the IMSI.
- The file EF_{LOCI} (6F7E) includes the parameters : TMSI, LAI.
- EF_{LP} (Language preference)
- EF_{Kc} (Ciphering key Kc) includes the Kc and the sequence number of the key.
- EF_{SST} (SIM service table) lists the available services in the SIM.
  - Service n°1 : CHV1 disable function
  - Service n°2 : Abbreviated Dialling Numbers (ADN)
  - Service n°3 : Fixed Dialling Numbers (FDN)
  - Service n°4 : Short Message Storage (SMS)
  - etc.
- EF_{ACM} (Accumulated call meter) is the total number of unit used for the current call and all the previous.
- EF_{MSISDN} (MSISDN) includes the phone number of the subscriber MSISDN.

Telcom Directory

- EF_{ADN} (6F3A) include the short diary,
- EF_{FDN} (6F3B) the contact list,
- EF_{SMS} (6F3C) the received and sent SMS,
- etc.
- These files are accessible in read and write mode and are protected with the Pin code.

File selection

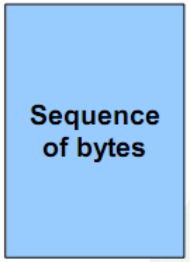
- Only one file is selected at a time,
- The MF is always selectable and is implicitly selected after a reset
- FID are not unique => restriction in selection



Université de Limoges
SUFOP Service Universitaire de Formation Permanente

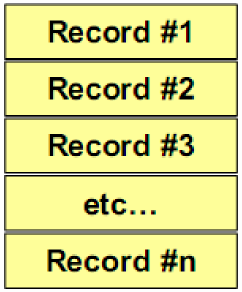
EF File structures

- Four data structure
  - Binary (transparent) files (data accessible through an address)
  - Sequential record fixed size or variable size
  - Cyclic buffer
- Transparent file
  - No internal structure
  - Accessed for reading or writing in bytes or blocks with an offset value
  - Often used with a small amount of data,
  - Commands READ BINARY, WRITE BINARY and UPDATE BINARY



Transparent

EF File structures

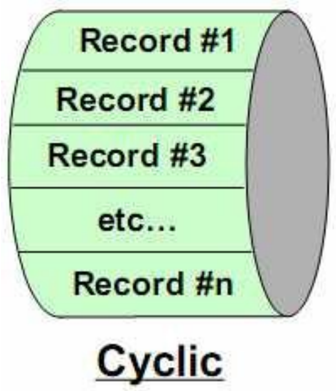


Linear Fixed

- Linear fixed file structure
  - Linking fixed length records,
  - The smallest unit is a record,
  - Commands: READ RECORD, WRITE RECORD and UPDATE RECORD, e.g. phone book
  - From 1..254
- Linear variable file structure
  - Same commands,
  - Need additional info concerning the length of each records,
  - Optimise the memory usage e.g. the phone book…

EF File structures

- Cyclic file structure
  - Based on the linear fixed file structure,
  - The EF contains a pointer on the last written record numbered 1, the previous 2, etc…
  - Can be accessed by addressing the first, the last, the next or previous record.



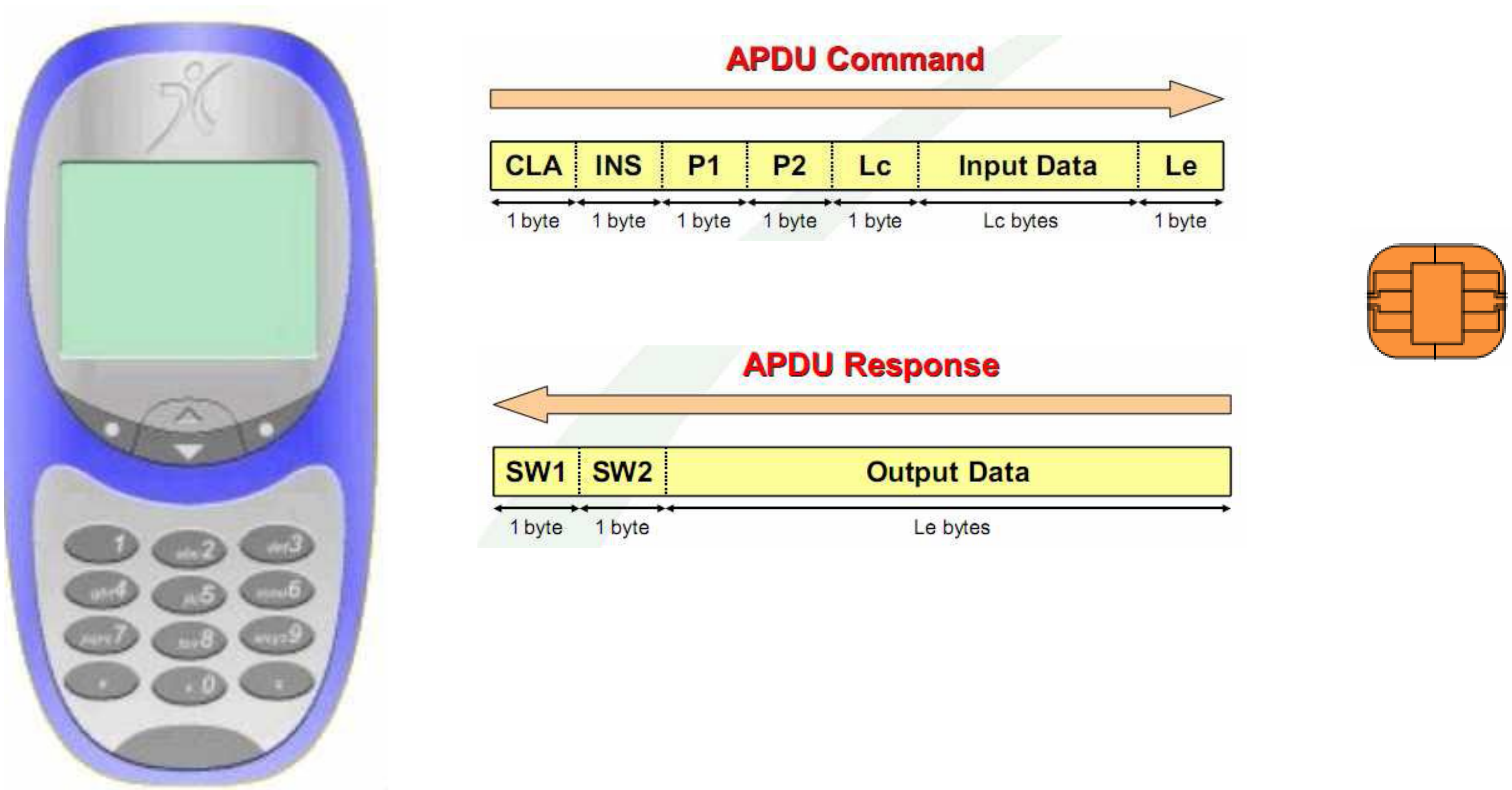
File Access Conditions

- Security is based on file access privileges,
- Access information coded in the header, defined when a file is created and usually cannot be changed later.
- For MF and DF
  - no information stored for data access (read and write)
  - But for creation and deletion of files.
- The PINs are stored in separate elementary files, EF_{CHV1} and EF_{CHV2} for example

File attributes

- Five kinds of EF files
  - **Always (ALW):** Access of the file can be performed without any restriction.
  - **Card holder verification 1 (CHV1):** Access can only be possible when a valid CHV1 value is presented
  - **Card holder verification 2 (CHV2):** Access can only be possible when a valid CHV2 value is presented
  - **Administrative (ADM):** Allocation of these levels and the respective requirements for their fulfilment are the responsibility of the appropriate administrative authority
  - **Never (NEV):** Access of the file is forbidden

The APDU commands



Université de Limoges
SUFOP Service Universitaire de Formation Permanente

GSM commands CLA = A0

- Data access commands
  - *Select (header)*, select a EF or MF, with a *getResponse* the status of the file
  - *Status*, sent by the terminal to a proactive SIM
  - *ReadBinary*, *UpdateBinary* read or update the current file
  - *Seek*, next record in the current file
  - *Increase*, add a record in a cyclic file
- Security related commands
  - *Verify*, *Change*, *Disable*, *Enable*, *Unblock* a CHV
  - *Invalidate*, *Rehabilitate* a file
  - *Run GSM Algorithm* run the A3 algorithm

The SELECT command

- A0 A4 00 00 02 XX XX (XX XX : FID of the EF of DF to be selected).
- The response of the selection request shall include:
  - size of the unused memory,
  - name of the DF file
  - kind of DF (MF or not)
  - PIN code request
  - number of included DF

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,

  - Any idea ?

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)
  - Select DF_{GSM} (7F20)
  - Select EF_{IMSI} (6F07)
  - Read 9 bytes with READ BINARY

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)

==> **A0 A4 0000 02 3F00**

<== **9F22**

  - Select DF_{GSM} (7F20)
  - Select EF_{IMSI} (6F07)
  - Read 9 bytes with READ BINARY

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)
  - Select DF_{GSM} (7F20)

  ==> **A0 A4 0000 02 7F20**

  <== **9F22**

  - Select EF_{IMSI} (6F07)
  - Read 9 bytes with READ BINARY

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)
  - Select DF_{GSM} (7F20)
  - Select EF_{IMSI} (6F07)

  ==> **A0 A4 0000 02 6F07**

  <== **9F0F**

  - Read 9 bytes with READ BINARY

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)
  - Select DF_{GSM} (7F20)
  - Select EF_{IMSI} (6F07)

    <== **9F0F**
  - 9FXX which mean success with XX bytes of response data, you can pull the response with GET RESPONSE command ‘C0′**,**

==> **A0 C0 0000 0F**

<== **00 00 00 09 6F 07 04 00 15 F5 15 01 02 00 00 9000**

Example

- How to read the IMSI ?
  - GET RESPONSE command ‘C0′,

```
0000 0009 6F07 04 00 15F515 01 02 0000 9000
|    |    |    |  |  |      |  |  |    status
|    |    |    |  |  |      |  |  structure 00 = transparent
|    |    |    |  |  |      |  length of data following
|    |    |    |  |  |      status
|    |    |    |  |  access    READ|UPDATE INCREASE|
|    |    |    |  |            RFU|REHABILITATE|INVALIDATE
|    |    |    |
|    |    |    file type 04 = EF
|    |    file  id
|    size
RFU
```

Example

- How to read the IMSI ?
  - File is EF_{IMSI} (6F07) of the GSM directory,
  - Select the Master File (3F00)
  - Select DF_{GSM} (7F20)
  - Select EF_{IMSI} (6F07)
  - Read 9 bytes with READ BINARY

==> **A0 B0 0000 09**

<== **08 29 80 02 12 34 54 90 03 9000**

EF_{IMSI} (IMSI)

Byte 1 length of IMSI

Byte 2-9 IMSI 8 bytes

Université de Limoges
SUFOP Service Universitaire de Formation Permanente

Access Right

- When you’re granted to CHV1 you can read its value but neither change it nor deactivate it.
  - Access Conditions:
    - READ CHV1
    - UPDATE ADM
    - INVALIDATE ADM
    - REHABILITATE CHV1
  - Access rights are coded as:
    - READ|UPDATE
    - INCREASE|RFU
    - REHABILITATE|INVALIDATE
    - knowing that '0' means always , '1' CHV1, 'F' never and '4'...'E' ADM.
    - 15 F5 15

PIN code commands

- PIN code is coded on 8 bytes. The non significant bytes are coded with FF.
  - my sim pin code is 0973, must be coded as 30 39 37 33 FF FF FF FF
- VERIFY CHV : verify the Pin
  - A0 20 00 P2 08 PIN (P2=01 for CHV1 (user PIN code), = 02 for CHV2).
- DISABLE PIN disable PIN usage.
  - A0 26 00 01 08 PIN
- ENABLE PIN enable PIN usage
  - A0 28 00 01 08 PIN
- CHANGE CHV modify the value of the PIN code
  - A0 24 00 01 10 previous_PIN new_PIN
- UNBLOCK CHV unblock a card that has its PIN code blocked (CHV1).
  - A0 2C 00 01 10 PUK PIN.

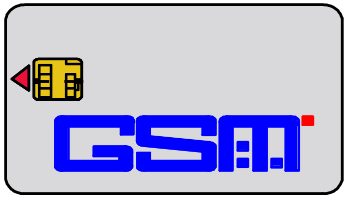
Read a GSM File

Select (DF $_{GSM}$)

SW1=9F, SW2=xx

GetResponse

[Nb EF files, access condition], SW1=90, SW2=00



Select (EF $_{LOCI}$)

SW1=9F, SW2=xx

GetResponse

[Type of EF file, access condition], SW1=90, SW2=00

ReadBinary

P1 & P2 provide the offset and the number of data to read in the file

[data], SW1=90, SW2=00

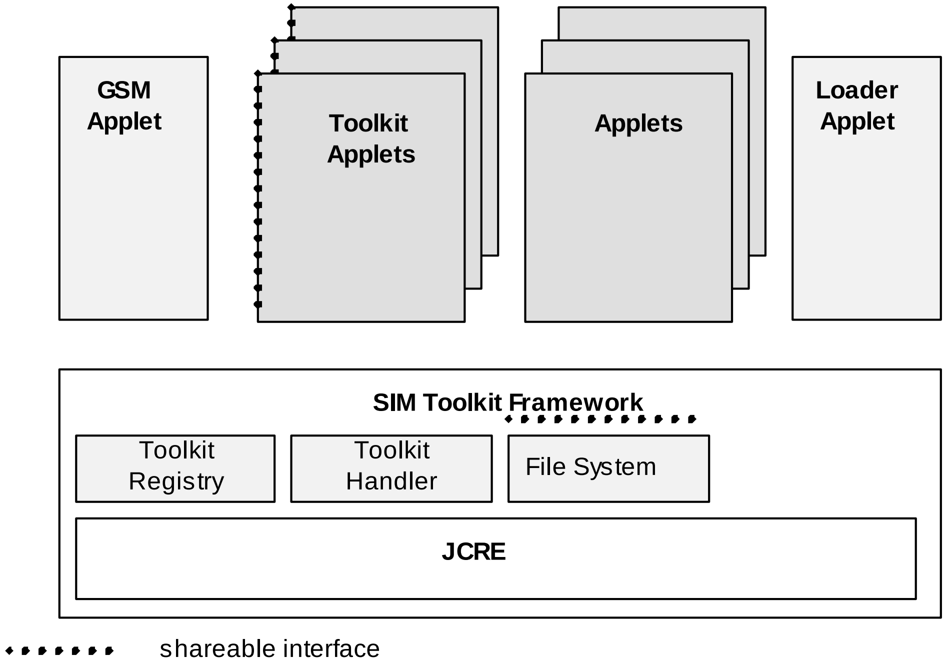
Part III

The SIM Toolkit framework

SIM Application Toolkit (SAT)

- Specified by the standard 3GPP TS 11.14,
- Additional framework that allows the SIM to interact with the mobile
- Identified with the content of the EF_{SST}
- Event programming application
- SMS used as an administrative mean (3GPP byte code interpreter) for RPC by the network admin or third tiers applications.

Architecture of the Java-SIM



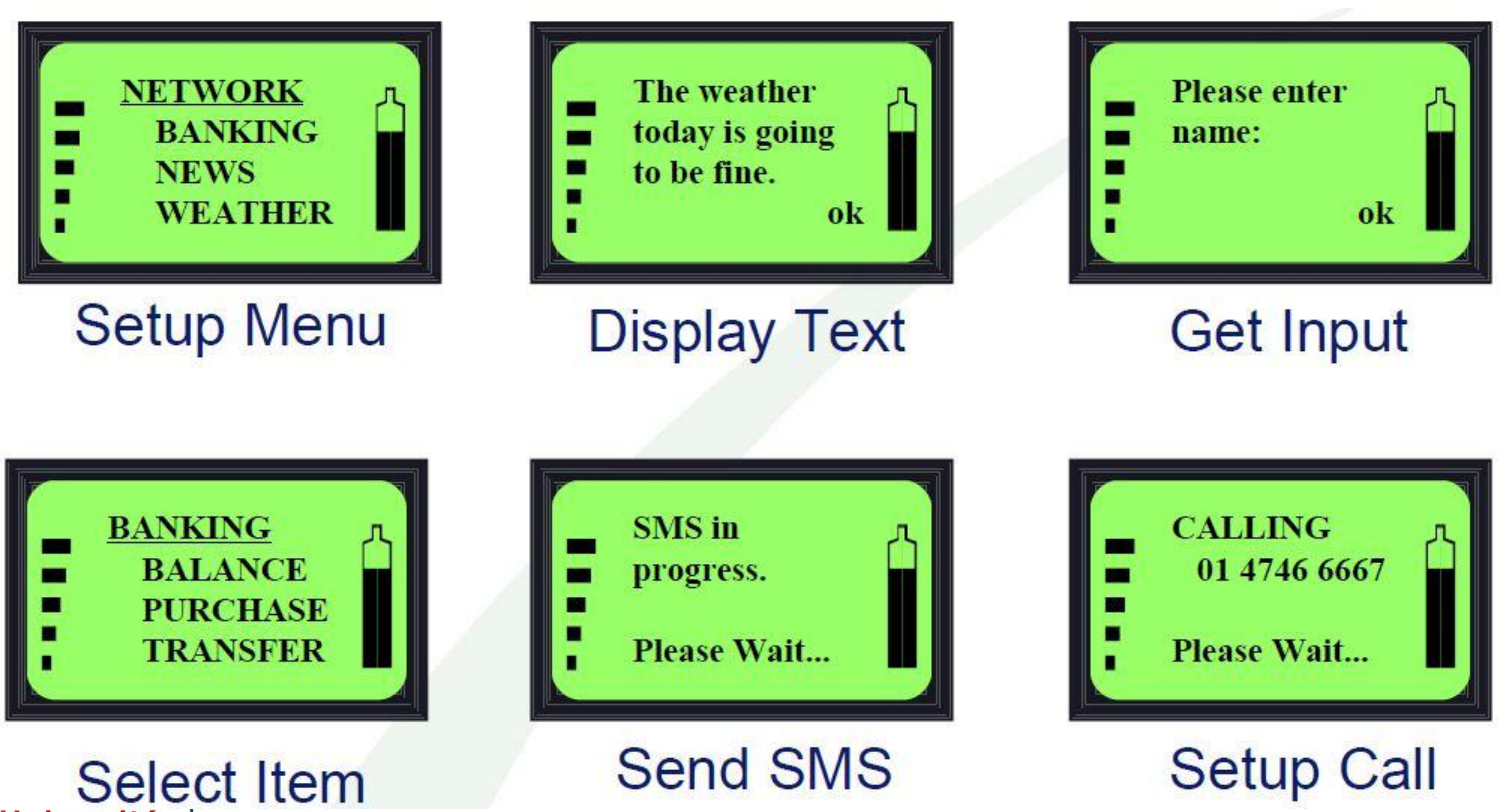
SIM Toolkit apps

- The SAT applications:
  - can initiate actions (pro-active commands),
  - can be externally triggered with events,
  - can get the characteristics of the mobile (a mean for the ME to tell the card what is able to do)
  - Four new APDU commands are defined to manage SIM Toolkit features
    - Fetch,
    - Terminal Response
    - Envelope
    - Terminal Profile

Pro active mode

- Only if the terminal supports this mode,
  - Get the terminal profile during initialization,
  - 20 bytes are sent back by the terminal
    - each bit codes facilities (TRUE= supported)
    - e.g. second byte, bit 8 Display Text is supported
- Command with the ME display :
  - Display Text, Set up menu, Send DTMF, Play Tone, Language Notification,…
- Commands with the keyboard/display
  - Get Inkey, get Input,…
- With the Radio equipment of the terminal
  - Set up, Send SMS, Send Sup. Services, Provide Local information,…
  - Launch browser, Perform Card APDU

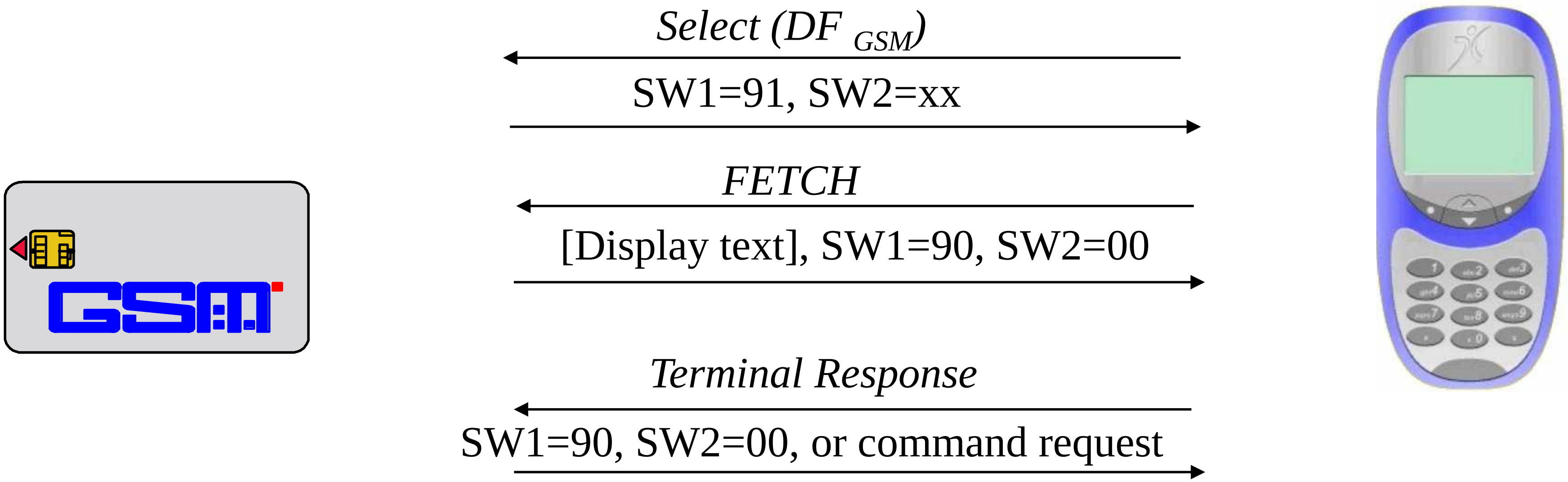
Pro Active Commands



To send a command

- Wait a poll request by the ME : command Status,
  - each second the ME polls the card (can be modified with a command),
- Wait for a regular command (e.g : Select File,..)
  - Answer with a 91xx status word,
  - The handset sends the SIM a FETCH command with an expended data length of xx size,
  - The handset parse it and execute the proactive command,
  - Send a response to the Card : Terminal Response which depends on the pro active command.

Pro active command



Pro active command

Status

SW1=90, SW2=00

...

Status

SW1=91, SW2=xx

Fetch

[Display text], SW1=90, SW2=00

Terminal Response

SW1=90, SW2=00, or command request

Structure of proactive command

- SIM gives the handset a sequence of Tag length Value (TLV),
  - The tag is always 1 byte and the length 1 byte (00..7F) or 2 bytes (81 + 80..FF),
  - Tags are well defined and can depend on the context,
    - Tag ME to SIM : 0xD1, 0xD2, 0xD3 & 0xD4
    - SIM to ME : 0xD0
  - A TLV can include other TLV and becomes a compound TLV,
  - Example :
    - Play music on our phone,
    - Expect a command or a status,
    - Send a request 91 23,
    - Wait for a Fetch,
    - Send the correct PLAY TONE command…

Structure of proactive command

- SIM gives the handset a sequence of Tag length Value (TLV),
  - The tag is always 1 byte and the length 1 byte (00..7F) or 2 bytes (81 + 80..FF),
  - Tags are well defined and can depend on the context,
  - A TLV can include other TLV and becomes a compound TLV,
  - Example :
    - Play music on our phone,
    - Expect a command or a status,
    - Send a request 91 23,
    - Wait for a Fetch,
    - Send the correct PLAY TONE command…

Start with ?

Play Tone

- The SIM toolkit application can request the application to play a short tone
  - Busy, CallWaiting, Congestion, Dial, Dropped, Error, GeneralBeep, NegativeBeep, PositiveBeep, RadioAck, Ringing

1	0xD0	Tag Proactive command
2	0x15	Length 21 bytes
3	0x01	Tag : command detail
4	0x03	Length
5	0x01	Identifier
6	0x20	Play Tone
7	0x00	Qualifier
8	0x02	Device identity
9	0x02	Length
10	0x81	Source (uicc)
11	0x82	Destination (me)
12	0x05	Alpha identifier
13	0x03	Length
14	0x42	Ascii value ‘S’
15	0x4F	Ascii value ‘O’
16	0x4F	Ascii value ‘O’
17	0x0E	Tone Value
18	0x01	Length
19	0x01	Play the dial tone
20	0x04	Duration
21	0x02	Length
22	0x01	Unit in second
23	0x05	Number of unit

ME response

- After the handset blast the dial tone (#1) for 5 seconds, it sends a status response,
  - Terminal response APDU, 0x80 0x14 0x00 0x00 0x00 0x0C data
  - The byte 12 is equivalent to a 90 00 from the ME

1	0x01	Command detail
2	0x03	Length
3	0x01	Identifier
4	0x20	Play Tone
5	0x00	Qualifier
6	0x02	Device identity
7	0x02	Length
8	0x82	Source (me)
9	0x81	Destination (uicc)
10	0x03	Result
11	0x01	Length
12	0x00	Success

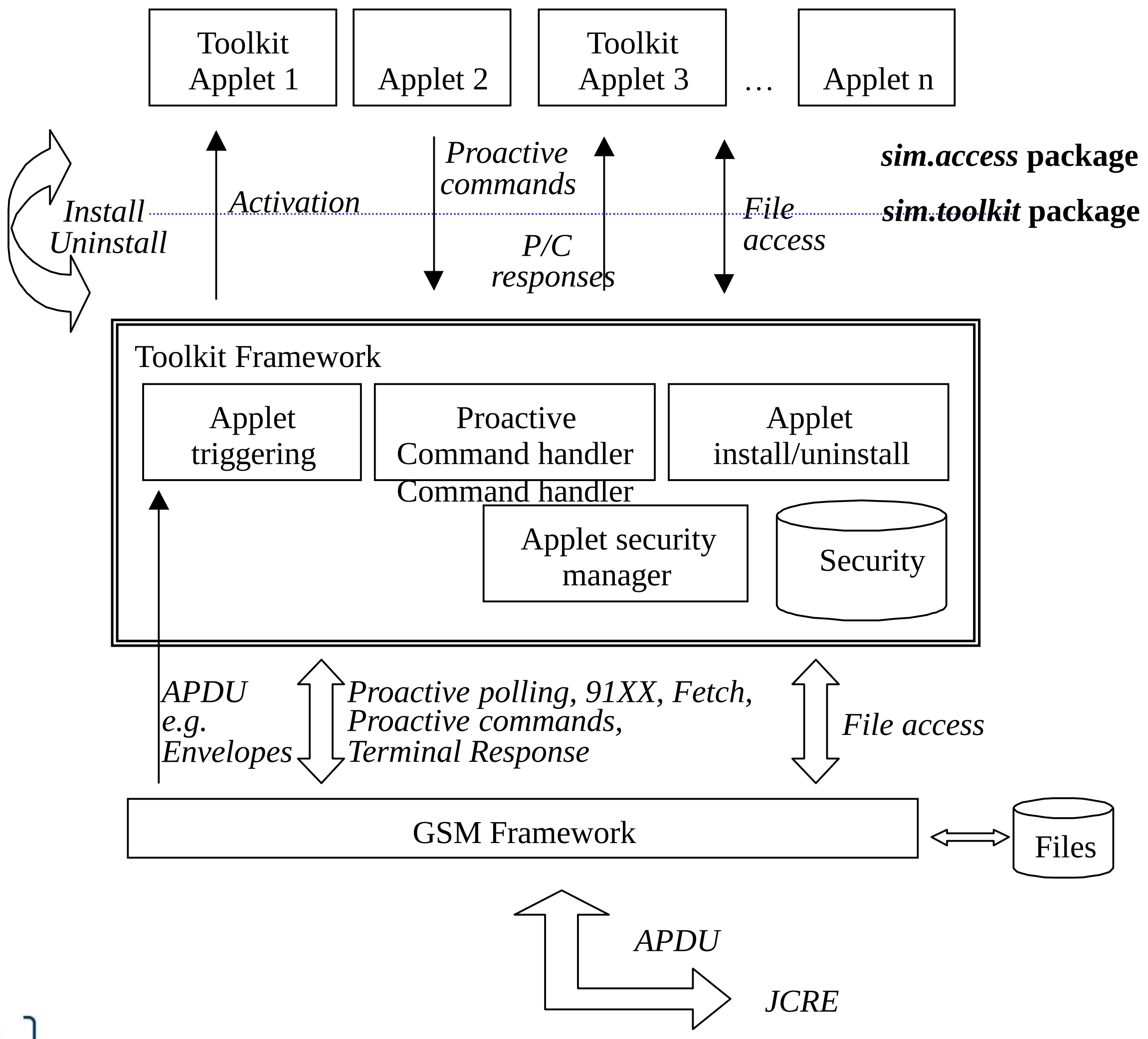
Event command

- The SIM can register for events
  - Can use the pro active command Setup Event list,
  - SMS-PP, Menu Selection, MT Call, Location Status, Browser Termination,…
  - The handset uses the Envelope APDU to send a description of the event to the card,
  - Structured with TLV as proactive commands,
  - Events can be routed by the handset from a service provider server.

Java Card SIM Applet

- The difference between Java Card Applet and Toolkit the latter does not handle APDU directly
- The API provides two packages:
  - The *sim.access* package, which allows applets to access the GSM files
  - The *sim.toolkit* provides methods to register to events, generate pro active commands,
  - The interface *ToolkitConstants*, encapsulates constants related to the Toolkit applets.
  - The *ToolkitInterface* must be implemented by a toolkit applet so that it can be triggered by the toolkit handler according to the registration information.

```
import sim.toolkit.*;
import sim.access.*;
public class MyToolkitApplet extends Applet implements
   ToolkitInterface, ToolkitConstants;
```



Université de Limoges
SUFOP Service Universitaire de Formation Permanente

sim.toolkit Classes

EditHandler	This class is the basic class for the construction of a list of simple TLV elements
EnvelopeHandler	The EnvelopeHandler class contains basic methods to handle the Envelope data field.
EnvelopeResponseHandler	The EnvelopeResponseHandler class contains basic methods to handle the Envelope response data field.
ProactiveHandler	This class is the basic class for the definition of Proactive commands
ProactiveResponseHandler	The ProactiveResponseHandler class contains basic methods to handle the Terminal Response data field.
ViewHandler	The ViewHandler class offers basic services and contains basic methods to handle TLV list.
ToolkitRegistry	The Registry class offers basic services and methods to allow any Toolkit applet to register its configuration during the install phase.
MEProfile	The MEProfile class contains methods to question the handset profile.

sim.toolkit Exceptions

ToolkitException	This exception extends the Throwable class and allows the classes of this package to throw specific exceptions in case of problems.

Toolkit registry

```
public class MyToolkitApplet extends Applet implements
   ToolkitInterface,ToolkitConstants {
public MyToolkitApplet() {
   reg = ToolkitRegistry.getEntry();
   menuId = reg.initMenuEntry(menuEntry, (short)0,
    (short)menuEntry.length, PRO_CMD_SET_UP_CALL, false, 0, 0);
   reg.disableMenuEntry(menuId);
   reg.setEvent(EVENT_FORMATTED_SMS_PP_ENV);

}

public static void install(byte bArray[], short bOffset, byte
bLength) throws ISOException {
   MyToolkitApplet applet = new MyToolkitApplet();
   applet.register();
}
```

Université de Limoges

SUFOP Service Universitaire de Formation Permanente

Toolkit registry

```
public class MyToolkitApplet extends Applet implements
   ToolkitInterface,ToolkitConstants {
public MyToolkitApplet() {
   reg = ToolkitRegistry.getEntry();
   menuId = reg.initMenuEntry(menuE
    (short)menuEntry.length, PRO_C
   reg.disableMenuEntry(menuId);
   reg.setEvent(EVENT_FORMATTED_SMS_

}

public static void install(byte bArray[], short bOffset, byte
bLength) throws ISOException {
   MyToolkitApplet applet = new MyToolkitApplet();
   applet.register();
}
```

The applet can be triggered by both selection mechanisms.

Université de Limoges

SUFOP Service Universitaire de Formation Permanente

Toolkit registry

```
public class MyToolkitApplet extends Applet implements
   ToolkitInterface,ToolkitConstants {
public MyToolkitApplet() {
   reg = ToolkitRegistry.getEntry();
   menuId = reg.initMenuEntry(menuEntry, (short)0,
    (short)menuEntry.length, PRO
   reg.disableMenuEntry(menuId);
   reg.setEvent(EVENT_FORMATTED_SMS_PP_ENV);

}

public static void install(byte bArray[], short bOffset, byte
bLength) throws ISOException {
   MyToolkitApplet applet = new MyToolkitApplet();
   applet.register();
}
```

Register for some events

How to handle pro active commands ?

- The SIM application toolkit protocol (*i.e.* 91xx, Fetch, Terminal Response) is handled by the GSM applet and the Toolkit Handler, the toolkit applet shall not handle those events.
- The SIM Toolkit Framework shall provide a reference of the `sim.toolkit.ProactiveHandler` to the toolkit applet so that when the toolkit applet is triggered it can :
  - initialise the current proactive command with the `init()` method ;
  - append several Simple TLV to the current proactive command with the `appendTLV()` methods ;
  - ask the SIM Toolkit Framework to send this proactive command to the ME and wait for the reply, with the `send()` method.

Example

```
private static final byte MY_COMMAND = PRO_CMD_DISPLAY_TEXT;
private static final byte MY_TAG = TAG_TEXT_STRING;
private byte[] text = new byte[12];
text[0] = (byte)'L';
text[1] = (byte)'I';
text[2] = (byte)'M';
ProactiveHandler proHdlr;
proHdlr = ProactiveHandler.getTheHandler();
proHdlr.init(MY_COMMAND, (byte)0, DEV_ID_ME);
proHdlr.appendTLV((byte)(MY_TAG | TAG_SET_CR), DCS_8_BIT_DATA,
   text, (short)0, (short)3);
result = proHdlr.send();
```

Université de Limoges

SUFOP Service Universitaire de Formation Permanente

Proactive Response Handler

```
private byte[] data;
data = new byte[32]; // build a buffer
ProactiveResponseHandler ProRespHdlr;
ProRespHdlr = ProactiveResponseHandler.getTheHandler();
byte result = ProRespHdlr.getGeneralResult();
respHdlr.findTLV(TAG_DEVICE_IDENTITIES, 1);
byte sourceDev = ProRespHdlr.getValueByte((short)0);
byte destinDev = ProRespHdlr.getValueByte((short)1);
if (ProRespHdlr.findTLV(TAG_TEXT_STRING, (byte)1) ==
TLV_FOUND_CR_SET) {
   if ((short len = ProRespHdlr.getValueLength()) > 1) {
        ProRespHdlr.copyValue((short)1, data, (short)0,
        (short)(len - 1));
        }
   }
```

Envelope Handler

```
private static final byte MY_TAG = (byte)0x54;
private byte[] data;
data = new byte[32];
void processToolkit (byte event) throws ToolkitException {
   // get the EnvelopeHandler system instance
   EnvelopeHandler theEnv = EnvelopeHandler.getTheHandler();
   // look for MY_TAG TLV
   if (theEnv.findTLV(MY_TAG, (byte)1) != TLV_NOT_FOUND) {
   // check first element byte
    if (theEnv.getValueByte((short)0) == (byte)1) {
        // copy element part into data buffer
        theEnv.copyValue((short)1, data,(short)0,
            (short)(theEnv.getValueLength() - 1));
        }
   }
}
```

Université de Limoges
SUFOP Service Universitaire de Formation Permanente

The complete example: Hello World

```
import javacard.framework.*;
import sim.toolkit.*;
public class HelloWorld extends Applet implements
   ToolkitConstants, ToolkitInterface {
private final byte COMMAND_QUALIFIER = (byte)0x80;
private final byte[] MENU_ENTRY =
{'C','r','y','p','t','i','s'};
private final byte[] HELLO_WORLD =
{'H','e','l','l','o',' ','w','o','r','l','d',' ','!'};
private ToolkitRegistry registry;
public HelloWorld() {
   registry = ToolkitRegistry.getEntry();
   registry.initMenuEntry(menuEntry, (short)0,(short)
   MENU_ENTRY.length, PRO_CMD_DISPLAY_TEXT, false, 0, 0);
}
```

```
public static void install(byte bArray[], short bOffset,byte
   bLength) throws ISOException {
   HelloWorld applet = new HelloWorld();
   applet.register();
}
public void processToolkit (byte event) throws
   ToolkitException {
 ProactiveHandler proHdlr = ProactiveHandler.getTheHandler();
 if (event == EVENT_MENU_SELECTION) {
   proHdlr.init((byte) PRO_CMD_DISPLAY_TEXT,(byte)
   COMMAND_QUALIFIER, DEV_ID_ME);
   proHdlr.appendTLV((byte)(TAG_TEXT_STRING),HELLO_WORLD,(shor
   t)0, (short) HELLO_WORLD.length);
   proHdlr.send();
   }
}
```

sim.access Interfaces & Classe

SIMView	SIMView is the interface between the GSM application and any SIM Toolkit applet.

SIMSystem	The Class SIMSystem provides a way to get access to the GSM file system. In any case, the SIM Toolkit applet will only access to methods of the SIMView interface. No instance of this class is needed.

Example

```
import javacard.framework.*; import sim.toolkit.*;
public class MyApplet extends Applet implements ToolkitInterface {
private SIMView simView; private byte[] buffer;
private ToolkitRegistry registry;
public MyApplet () {
 registry = ToolkitRegistry.getEntry();
 simView = SIMSystem.getTheSIMView();
 buffer = new byte[32];
}
public static void install(APDU apdu) throws ISOException {
 MyApplet applet = new MyApplet();
 applet.register();
}
public void getADN(short adnNumber) {
 simView.select(SIMView.FID_EF_TELECOM);
 simView.select(SIMView.FID_EF_ADN);
 simView.readRecord((short)adnNumber,  SIMView.MODE_ABSOLUTE,
   (short)0, buffer, (short)0, (short)32);
 }
}
```

Université de Limoges

SUFOP Service Universitaire de Formation Permanente

Any question ?

Université de Limoges

SUFOP Service Universitaire de Formation Permanente