Detecting GSM Hopping Channels using OsmocomBB

Jan Ruge - Physical Layer Security - github.com/bolek42/pygsm.git

TECHNISCHE UNIVERSITÄT DARMSTADT

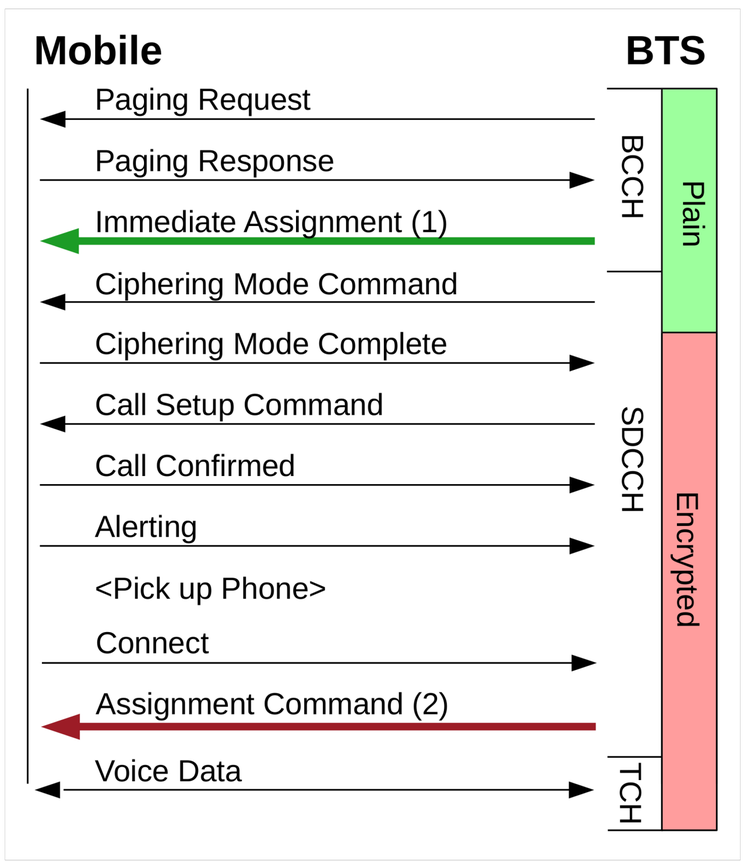
GSM Call Setup

Capturing voice Data is one major threat for GSM. TO understand the technical problem, first the call setup procedure is shown. This usually invokes the following logical channels, that may use frequency hopping:

Broadcast Control Channel (BCCH)
- An idle phone listens to this non hopping channel
- For incoming services, the phone gets paged
- Not used for excessive uplink

Stand-Alone Dedicated Control Channel (SDCCH)
- Bidirectional control channel
- Used for call Setup or SMS transmission
- Usually frequency hopping

Traffic Channel (TCH)
- Used to transmit, e.g. voice Data
- Often frequency hopping



The *hopping Parameter for SDCCH are always transmitted in Plaintext.* As the call setup is encrypted, the *hopping parameter for TCH are not known*. It is possible to break the A5.1 encryption using a *known plaintext attack,* but this approach requires several minutes and is not feasible for recovering TCH hopping parameter. On both SDCCH and TCH *System Information Type 5* (Neighbour Cells) and *System Information Type 6* (Cell Identification) messages with a known structure are transmitted every 104th frame. They could be used to mount the known plaintext attack.

OsmocmBB

As *frequency hopping is too fast for most SDRs*, an attacker who wants to sniff a hopping channel has to capture all channels allocated by the cell at he same time. This results in the following technical problems:

1. The attacker has to capture a lot of Data
   - Cell allocation usually spreads over the whole Band
   - Lots of Bandwidth to capture
   - High end SDR required

2. The attacker has to channelize Signal
   - Computation Intense Task
   - Difficult to perform in real time
   - Extra step in Airprobe Hopping

Multiple cheap RTL-SDRs could be used, to capture all channels, but the signals has to be synchronized in time domain to reassemble the packets. In case of GSM cheap alternative to SDRs is an *Osmocom-bb compatible phone which is already capable of frequency hopping* and can be operated with an open source firmware. The project includes a complete GSM mobile station implementation, that could also be used as a single time slot sniffer or selective jammer.

Frequency Hopping

GSM Hopping Channels change frequency every frame (4.62ms) , using a PRNG. Such a channel is described by the following parameter:

HSN: Hopping Sequence Number (0...63)
MAIO: Mobile Allocation Index Offset (|MA|)
MA: Mobile Allocation (list of used channels)
TS: Timeslot

```
function current_channel(hsn, maio, ma, fn)
      s = prng(hsn, fn)
      mai = (s+maio) % len(ma)
      return ma[mai]
```

The channel number is computed by the hopping parameter and the frame number. *HSN is the seed for the PRNG* and MAIO describes the offset in the Cell Allocation list. MAIO is used to make use of all available channels without collisions. As time slot 0 on CH 0 is always used for BCCH, this cannot be part of the Mobile Allocation.

fn	0	1	2	3	4	5	6
CH 0	0	2	1	1	0	1	2
CH 1	1	0	2	2	1	2	0
CH 2	2	1	0	0	2	0	1

HSN = 42 Timeslots: 0 1 2 3 4 6 7

It is obvious, that for each time slot, the HSN has to be static, otherwise collisions will occur between different channels. Even though, the HSN could theoretically be randomized for each time slot. To observe the reuse of hopping parameter in practice, a wrapper for OsmocomBB and a GSMTAP parser was implemented. By initiating a call *the encrypted and unencrypted hopping parameter can be compared.*

17/17	Hopping SDCCH and different parameter for TCH
17/17	Use all channels for hopping (MA=CA)
10/17	No frequency hopping for TCH
7/7	Reuse HSN from SDCCH for TCH

In all cases HSN is known to the attacker and the Mobile Allocation is not randomized. The only unknown parameter are MAIO and the time slot, which can be easily bruteforced, as there are only 8*|CA| possibilities.

Fast power measurements

If an attacker knows HSN, it is possible to *discover active and newly allocated hopping channels, by correlating signal power with the hopping sequence*. The firmware of OsmocomBB already implements neighbour cell discovery, that allows to specify a channel number and time slot for power measurements. The *mframe* scheduler will execute specific tasks for each frame. Usually neighbour discovery is done on frames 0,10,20,30,40 modulo 51. This was modified to do power measurements for every frame.

src/target/firmware/layer1/mframe_sched.c
```
static const struct mframe_sched_item mf_neigh_pm51_c0t0[] = {
  {.sched_set = NEIGH_PM, .modulo = 1, .frame_nr = 0 },
  {.sched_set = NULL }
};
```

Each measurement is done in three ticks, each one frame. The resulting offset of two frames has to be compensated in the result collection.

1. Signaling the DSP to receive Data
2. DSP is receiving
3. Capture results

Unused fields in the measurement report message were used to get the frame number. By performing one measurement on every frame (4.62ms), on average *each hopping channel will be hit every* ***|CA| * 36.96ms***. This implementation allows real time detection of active channels by using one mobile.