

# Security of GSM and LTE Networks

## Teaching Resources

Joshua S Curry and Denis A Nicole  
Electronics and Computer Science  
University of Southampton\*

13th September, 2021

### 1 About this document

This document serves as a teaching introduction to GSM and LTE Networks through a practical laboratory developed to support the knowledge area of ‘Physical Layer and Telecommunications Security’ of CyBOK.

It aims to give you everything you need to run the lab, alongside setup instructions and model answers to accelerate the design and delivery of teaching. These notes are not intended as a step-by-step guide and instructors will need a clear knowledge of the surrounding area to deliver this content. In addition to this, whilst information on legal processes is correct to the best of the authors’ knowledge at the time of writing, it does not constitute legal advice nor is permission to operate any transmitting or receiving equipment.

---

\*This project was funded under *Cybersecurity Body of Knowledge (CyBOK)* subaward 2021–2471.

# Contents

<b>1</b>	<b>About this document</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>About GSM</b>	<b>4</b>
3.1	Frequency bands . . . . .	4
3.2	Legislation . . . . .	4
3.3	Licensing . . . . .	5
<b>4</b>	<b>The Laboratory</b>	<b>5</b>
4.1	Prerequisite Knowledge . . . . .	5
4.2	Base Station Setup . . . . .	6
4.2.1	Hardware . . . . .	6
4.2.2	Software . . . . .	7
4.3	Important Notes . . . . .	8
4.3.1	Licensing and Configuration . . . . .	8
4.3.2	Check Output . . . . .	8
4.3.3	Do not change MCC/MNC . . . . .	8
4.3.4	Whitelist mobile devices . . . . .	8
4.4	Mobile Station Setup . . . . .	8
4.4.1	SIM Cards . . . . .	9
4.5	Laboratory Layout . . . . .	9
<b>5</b>	<b>Extension</b>	<b>9</b>
5.1	Join multiple clients . . . . .	9
5.2	SMS Exploration . . . . .	9
5.3	Enabling (weak) encryption . . . . .	10
5.4	Introducing Mobile Data . . . . .	10
5.5	IMSI Catching . . . . .	10
<b>6</b>	<b>Further Work - 4G/LTE</b>	<b>10</b>
<b>A</b>	<b>Laboratory Setup - Raspberry Pi</b>	<b>11</b>
<b>B</b>	<b>Running the Laboratory</b>	<b>13</b>
<b>C</b>	<b>Sample Captures</b>	<b>15</b>
<b>D</b>	<b>4G Installation</b>	<b>15</b>

## 2 Introduction

In the last 30 years, mobile phones have evolved from a bulky technology aimed at the business sector, into a ubiquitous technology universally available across the world.

Whilst handsets have evolved in recent years, from the classic Nokia 3310 released at the turn of the century to the Android and iPhones of the modern age, the backend infrastructure supporting their operation has remained largely the same.

The second generation of cellular networks, more commonly known as 2G, was released as part of the GSM standard in 1991. There have been numerous extensions and modifications to the standard, resulting in the development of 3G, 4G and 5G but, due to its ubiquitous presence in the world for over 30 years, most modern mobile handsets still support operating to the original 2G specification. Indeed, it is likely that 2G will outlast 3G because of its presence in long-life industrial *internet of things* devices.

The original security model of GSM networks relies heavily on equipment to create a functional wireless network being inherently expensive and complex, and thus unlikely to be possessed by nefarious actors. For example, so-called *IMSI catchers* used to be expensive tools of law enforcement and intelligence agencies; nowadays, you could (illegally) set one up for £300.

This reliance on networks only being run by a trusted set of operators can be seen throughout the original standard, with a distinct lack of authentication of base stations by mobile stations, weak encryption standards and a reliance on globally-unique identifiers which could be maliciously used for tracking.

This walled-garden approach has been largely challenged in recent years with the advent of Software-Defined Radio (SDR). In stark contrast to 1991, radio equipment can now be purchased for mere hundreds of pounds which can fully emulate a GSM base station and be used to perform a wide variety of attacks on even the latest cellular devices.

The problems are not only in the GSM local link. At the network level, there are serious vulnerabilities in the [SS7 signalling standard](#). Using social engineering, [SIM swap](#) fraud offers further opportunities for criminality.

Many of these aforementioned security issues should have been mitigated with the most recent revisions of the GSM standard. LTE (*long term evolution*, effectively 4G), for instance, requires *base transceiver stations (BTS)* to be authenticated by *mobile equipment (ME)* as well as vice-versa to create a secure connection between the client and operating infrastructure.

Despite these advances in technology, new security features deployed in more modern standards are undermined by the fact that the old standards are kept in operation. In the United Kingdom for instance, 2G networks are still widespread and utilised widely by most network operators. Indeed, as of mid 2021, most networks do not use true LTE (*Voice over Long Term Evolution: VoLTE*) for voice calls but routinely downgrade to 2G. Thus LTE routinely remains vulnerable to older GSM attacks.

This means that the security of 2G networks is as relevant today as it always has been, especially with such a focus in recent years on multi-factor authentication using simple SMS messages as a low-cost *two factor authentication (2FA)* for authenticating and in some cases recovering users online accounts.

This document establishes an experimental 2G network and sample laboratory which can be used as an introduction to teaching this topic.

## 3 About GSM

### 3.1 Frequency bands

Cellular networks operate in several frequency bands depending on region; many of these have moved into and out of mainstream usage in recent years due to shifts to newer technology.

In the United Kingdom, there are typically two main bands in which conventional 2G networks operate. These are commonly known as the E-GSM 900 and DCS1800 bands.

Frequency Band	ARFCNs	Downlink	Uplink
E-GSM 900	0-124, 975-1023	925-960MHz	880-915MHz
DCS1800	512-885	1805-1880MHz	1710-1785MHz

Table 1: UK 2G Frequency Bands

Naturally, current-generation mobile handsets support multiple bands for internal travel, and as such many other frequency bands are also supported, but unavailable for usage in the UK due to conflicting use for other services such as TV distribution and other wideband communications.

### 3.2 Legislation

In the United Kingdom, transmit and receive access to radio frequency bands are covered by the [Wireless Telegraphy Act of 2006](#), which stipulates that it is illegal to transmit or listen on bands for which the user does not have a licence, unless explicitly permitted.

In the context of this laboratory, this means that it is illegal to broadcast on any frequency bands which are licensed to mobile operators without a licence, and it is equally an offence to listen in to (and in some cases decrypt) mobile phone communications for which the user is not the intended recipient. Mobile phone operator licences are extremely expensive and spectrum is [auctioned by Ofcom](#). In consequence of the substantial investment, unauthorised access to these parts of the spectrum is rigorously policed.

It is likely that individual users own their phones (ME), but they are not likely to own the SIM (*Subscriber Identity Module*) inside or, indeed, have direct access to the cryptographic keys contained in it. A commercial SIM will routinely allow connection to a 2G test network, but this is probably outside its permitted use. To be safe, you need to purchase dedicated SIMs for your test network. For

LTE networks, this is essential as you will otherwise be unable to authenticate the network to the phone.

### 3.3 Licensing

There are two main licences which can be sought to run a test mobile phone network for teaching and research purposes. The first of these is an [Innovation and Trial](#) licence, which is offered by Ofcom on a yearly basis for non-operational purposes; it cannot be used to run a production network.

Innovation and trial licensing allows the applicant to apply to operate on any mobile frequency, with any specific emission type. In the case of a typical test GSM network, this would be a **271KHz-wide** channel of emission class of **271KF7W** in one of the two tabulated frequency bands above.

The second type of licence has only become available recently, and allows for the licensing of a small amount of spectrum in the DECT Guard Band (1876.7-1880 MHz), which also overlaps some of the DCS1800 spectrum. This [Shared Access licence](#) from Ofcom allows for the licensing of up to one 3.3MHz channel, which is enough to run a 2G, 3G or 4G network.

Both of these types of licence are location-restricted, with the Innovation and Trial licence requiring a fixed address for the base station, and the Shared Access licence allowing low-power transmission at a radius of 50m from a fixed location. Other characteristics such as transmission power and antenna gain must be taken into account and licensees must ensure that they do not contravene the licensing conditions. There is a six-week delay on the granting of Innovation and Trial licences as a frequency has to be negotiated and nearby phone operators have to give their consent. The Shared Access licence should be obtainable more quickly but this is not always the case; one of our three such applications (in a relatively rural area of Dorset) was declined because of band occupancy.

Please do communicate freely with [Ofcom](#); their staff are genuinely trying to be helpful and they generally adopt a light touch. Do not be tempted to operate without a licence or to conceal what you are doing.

## 4 The Laboratory

The suggested laboratory utilises the [Osmocom](#) network stack in combination with some off-the-shelf hardware to create a test GSM network. It is important that an appropriate licence as indicated above is obtained for the premises of operation, and that careful consideration is given to configuration parameters to avoid accidentally routing emergency calls or data from other devices (see [4.3](#)).

### 4.1 Prerequisite Knowledge

In order to get the most out of this laboratory, students should be familiar with a number of concepts relating to telecommunications and networking.

- Common network layers and their application to standard IP networks.

- A basic understanding of radio networks, the concept of frequency, bandwidth, common modulation schemes and channelisation.
- Packet captures and use of Wireshark (only simple application usage required, no knowledge of filters)
- How to use a command line (for example `bash`), how to provide arguments and read manual pages (`man`)
- A broad understanding of how a device joins a GSM network, endpoint identifiers such as IMEI and IMSI and network identifiers such as MCC and MNC (covered elsewhere in this document).

## 4.2 Base Station Setup

A typical GSM network consists of many components, designed for the management of a large number of clients at many, possibly international, locations. For this laboratory, a somewhat simpler setup was created, consisting of inexpensive off-the-shelf hardware and open source software.

### 4.2.1 Hardware

The hardware for this laboratory is split into two main groups. The base station hardware, used by the instructor to transmit a functional GSM mobile network, can be found in Table 2.

Item	Description	Approx. Price (£)
Raspberry Pi 4 4GB	Single-board computer	50
LimeSDR Mini	Software-Defined Radio	150
2pcs 5V 2A USB PSU	Power supplies for Pi and LimeSDR	15
USB3 Y-Cable	Allows external 5V PSU to LimeSDR	5

Table 2: Base Station Hardware Components

The requisite mobile device hardware to connect to the base station can be found in Table 3. We strongly recommended that you utilise a sacrificial mobile device for the client, with a dedicated non-commercial SIM as through the exploration of the laboratory private information such as the IMEI (handset identity) and IMSI (SIM card identity) will be visible to participants.

Item	Description	Approx Price (£)
Mobile Phone	Phone capable of 2G	15
sysmoISIM-SJA2	Reprogrammable SIM Card	50

Table 3: Mobile Device Hardware Components

The hardware required by each student to capture data from this base station can be found in Table 4.

Item	Description	Approx. Price (£)
LimeSDR Mini	Software-Defined Radio	150

Table 4: Student Hardware Components

#### 4.2.2 Software

Whilst the software landscape of a production GSM network can span tens of different components, a GSM BTS (*Base Transceiver Station*) for testing can be created using a recent Raspberry Pi device and just five main pieces of software.

**osmo-nitb**. Starting at the highest-level part of the BTS, **osmo-nitb** or “network in a box” implements most of the major parts of a functional network, including the HLR (*Home Location Register*), which is used to store information about subscribers, and the MSC (*Mobile Switching Centre*), which deals with operational messaging of clients. In a production GSM network, these components would be run at the premises of the mobile operator or in a data centre.

**osmo-bts-trx** connects to **osmo-nitb** and implements layers 2 and 3 of a typical BTS system, allowing LAPDm (*Link Access Protocol on the Dm Channel*) messages to be transferred to **osmo-trx-lms** for transmission. This can be re-alised in a production network as a controller positioned at the physical site of the transmitting station.

**osmo-trx-lms**. Finally, at the RF end of the device chain, OsmoTRX implements Layer 1 (the physical layer) of a BTS, and allows the LimeSDR Mini to be used as a transceiver for this data. In a production network, this represents a single physical radio transceiver at a cell site. It is worth noting that this part of the chain can be switched out; different variants of OsmoTRX can be used with USRP SDRs.

The three software components described so far will provide a working base station with general paging capability, and form the major part of this implementation of a base station powered by low-cost hardware. In order to provide call-routing capability, it would be required to bridge the network into an existing PBX solution such as [Asterisk](#).

**Asterisk** is a free and open source IP PBX, allowing for the routing of calls and creation of a ‘two factor’ phone call service which will be utilised in this laboratory.

**osmo-sip-connector** allows **osmo-nitb** to bridge calls from subscriber devices to SIP, which can then be routed into Asterisk. This allows calls to be made from a SIP client to devices connected to the GSM network, and vice-versa.

A full guide to installing these pieces of software on a Raspberry Pi can be found in [Appendix A](#).

## 4.3 Important Notes

In order to start the base station and begin the lab, follow the instructions in Appendix B. Before running the laboratory, there are a number of important things to check to ensure that you stay within legal operating parameters.

### 4.3.1 Licensing and Configuration

As mentioned above, operating a GSM base station without a licence is illegal in the UK. Make sure that you have the relevant licence to broadcast.

Make sure that you are operating on a frequency and power level within that licence. A key configuration line to note is the `arfcn 808` setting in `openbsc.conf`, which sets the specific frequency of operation. Make sure that this is within your licensing parameters before running.

### 4.3.2 Check Output

It is generally good practice to check your output with another SDR on another computer using a standard SDR viewer such as [SDR Console](#), to make sure that you are operating within the parameters specified above. Listening to *your own network* is lawful as you *are* the intended recipient.

### 4.3.3 Do not change MCC/MNC

The sample configuration contains a MCC (country code) of 001 and a MNC (network code) of 01. These are defined as a “testing” country and network code respectively; they are not to be used for production networks. It is important not to change these to replicate real networks as doing so would create an IMSI catcher onto which nearby phones would roam, potentially disrupting emergency calls.

### 4.3.4 Whitelist mobile devices

In its default state, OpenBSC requires devices to be whitelisted to join the network. This is important as otherwise you could disrupt the cell service of local users who might “roam” onto your test base station. Even though devices should be constrained to their own networks, the authors have seen in practice that devices attempt to roam onto the network unprompted, so it is important to ensure that authorisation remains enabled to prevent disruption of service for users.

## 4.4 Mobile Station Setup

Any modern device which supports 2G in the DCS1800 band can be used to connect to the network. During practical testing, the authors have found that device reboots are occasionally needed to join the network, presumably due to synchronisation issues between the test BTS and the commercial cellular networks.

#### 4.4.1 SIM Cards

The hardware above includes a Sysmocom reprogrammable SIM card. The use of a test SIM is important as its unique IMSI identifier is likely to be discovered by participants of the laboratory: a commercial IMSI should be considered private information and its use for our laboratory might be considered unauthorised. Hence, caution should be followed before connecting personal devices to the test BTS as commercial information will likely be visible.

### 4.5 Laboratory Layout

The included laboratory document poses a number of questions to students, and is split into four sections.

Task 1 asks participants to investigate a number of common parameters of 2G networks, investigating the control and data channels in use alongside fixed network parameters. Task 2 asks participants to find these parameters on the test GSM network, either on a live network using **grgsm\_livemon** or from captures using **grgsm\_decode**. Task 3 utilises the regular repetition of SMS messages configured in Appendix B to allow users to capture and decode SMS messages over the air and find a sample two-factor authentication code; the same path is followed with Task 4, which asks participants to do the same with an authentication voice call.

Ideally, using *LimeSDR Mini* SDRs on a live network is the preferred route for teaching, as it allows participants to gain vital practical experience with software defined radio technology. If this is not possible, for example to facilitate remote working, example captures can be found in Appendix C.

## 5 Extension

This laboratory provides the basic foundation to get started teaching mobile network technology from a security context. We suggest that, for more challenging assessments, the laboratory can be easily modified with new data, more clients, and potentially in small numbers even the ability to run smaller test cells and perform live attacks. Cryptography can also be enabled

### 5.1 Join multiple clients

One single client connected to the base station leads to a very easy RF environment to capture. To enhance the laboratory, multiple clients could be connected and could even interact with each other through the BTS and to services externally to add “noise” to the captured data and make it more interesting to extract target data.

### 5.2 SMS Exploration

Textual SMS messaging is only a small part of the GSM specification, with USSD codes, binary SMS and cell broadcast providing a whole host of other functionality, some of which can be used to subvert handsets. It is suggested that Task 3 and Task 4 could be extended by harnessing some of this technology

provided by the Osmocom stack for more interesting packet captures and more data to investigate.

### 5.3 Enabling (weak) encryption

GSM includes a number of weak encryption schemes which can be cracked in a matter of minutes or hours by modern computing hardware. The BTS described here has encryption disabled for ease of initial teaching, but it can easily be enabled in the configuration using a set key on the reprogrammable SIM card to require participants to crack the keys to and decrypt the laboratory data.

### 5.4 Introducing Mobile Data

Alongside **osmo-nitb**, other elements of the Osmocom stack can be installed to allow for the capture of mobile data from other applications and services to be added to the laboratory. Due to the complexity of configuring these services, they are not included in this document, but allow the attack surface for participants to expand to include a wide range of IP services.

### 5.5 IMSI Catching

In small groups, dependent on the licence parameters, participants can develop an IMSI catcher for devices connected to the test network. This would essentially involve creating another BTS using the software defined here and encouraging a mobile device (with a reprogrammable SIM associated to the test MCC/MNC) to roam onto the network, allowing the participants to capture data via a man-in-the-middle attack on test devices. Due to licensing constraints it is recommended not to run this with large numbers of people, as the organiser will need to ensure that they are all operating within the licence parameters.

## 6 Further Work - 4G/LTE

Although downgrade attacks and the continued widespread usage of 2G technology mean that the glaring security vulnerabilities in the 2G specification are very relevant to the current security landscape, the introduction of newer technologies such as 4G is also worth exploring from a security context.

The main issue with exploring more recent protocols is that the open-source tooling for reverse engineering them has yet to catch up with the open-source network implementations. As part of this publication, a test network based on *srsLTE* was developed, the installation process for which is described in Appendix D. Although this gives a fully-functional 4G network on the same DECT guard band frequency as utilised above, it is difficult with currently-available tooling to capture packets from, and reverse engineer, such a network.

This will likely change in the near future, as the security community begins investigating the newer telecommunications technologies and developing open-source toolsets to work with them. There is some (not security-specific) work on teaching LTE available as commercial courseware from [DreamCatcher](#). Unlike our laboratories, they appear not to choose to acquire a spectrum licence

and support only a directly wired radio frequency connection between their LimeSDR base and an LTE USB “dongle”.

## A Laboratory Setup - Raspberry Pi

This laboratory was installed on a Raspberry Pi 4 Model B with 4GB RAM. For simplicity, Ubuntu Server 20.04 was utilised as the operating system, installed from the *Raspberry Pi Imager* utility. With minor modifications and potentially with compilation a few of the components from source, the following instructions should also work on the default Raspbian OS.

As described above in table 2, the Raspberry Pi will need to be connected with the LimeSDR Mini through an adapter cable. This setup is shown in Figure 1.

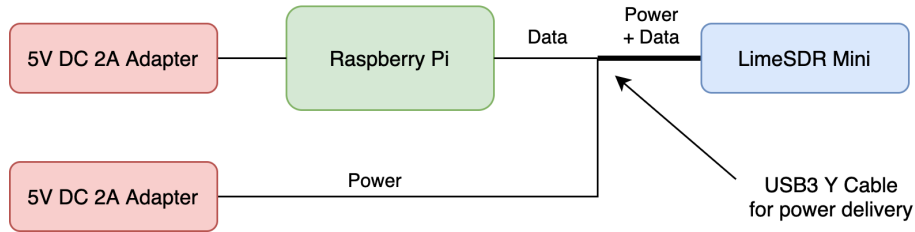


Figure 1: Raspberry Pi Connections

Firstly, SDR support needs to be built. To do this we will install *LimeSuite* and *SoapySDR*. This part of the software installation is well-documented during the setup of *srsLTE*, and is identical to that found [on the srsLTE web site](#).

```
1 sudo apt install build-essential cmake
```

Listing 1: Install LimeSuite and SoapySDR dependencies

With dependencies installed, LimeSuite and SoapySDR can be downloaded and built.

```

1 git clone https://github.com/pothosware/SoapySDR.git
2 cd SoapySDR
3 git checkout tags/soapy-sdr-0.7.2
4 mkdir build && cd build
5 cmake ..
6 make -j4
7 sudo make install
8 sudo ldconfig
9
10 sudo apt install libusb-1.0-0-dev
11 git clone https://github.com/myriadrf/LimeSuite.git
12 cd LimeSuite
13 git checkout tags/v20.01.0
14 mkdir build && cd build
15 cmake ../
16 make -j4
17 sudo make install
18 sudo ldconfig
19 cd ..
20 cd udev-rules
21 sudo ./install.sh

```

```

22
23 # Test installation
24 SoapySDRUtil --find

```

Listing 2: Install LimeSuite and SoapySDR. Listing from [here](#)

The following commands install **osmo-nitb** and **osmo-bts-trx**. If you are not using Ubuntu, you may have to download and build these from source in a similar fashion to other components.

```

1 sudo apt update
2 sudo apt install osmocom-nitb
3 sudo apt install osmo-bts

```

Listing 3: Install Osmocom components from apt

Now, we need to install **osmo-trx-lms**. This is done by compiling from source.

```

1  ### Build libosmocore
2  git clone git://git.osmocom.org/libosmocore
3  cd libosmocore
4  autoreconf -fi
5  ./configure
6  make -j5
7  sudo make install
8  sudo ldconfig
9  cd ../
10
11 ### Build osmo-trx-lms
12 git clone git://git.osmocom.org/osmo-trx
13 cd osmo-trx
14 autoreconf -fi
15 ./configure --without-uhd --with-lms
16 make -j5
17 sudo make install
18 sudo ldconfig
19 cd ../

```

Listing 4: Install LimeSuite and SoapySDR. Listing adapted from [here](#).

At this point, you should be able to see your SDR when you run **SoapySDRUtil --find**, and you should be able to access the three main installed Osmocom components by simply typing their names.

Remember to plug in your LimeSDR using the USB3 Y cable to supply extra power. If you do not do this, you may encounter unexpected behaviour, including Raspberry Pi reboots etc.

```

1 josh@testpi:~$ SoapySDRUtil --find
2 #####
3 ##      Soapy SDR -- the SDR abstraction library      ##
4 #####
5
6 Found device 0
7   addr = 24607:1027
8   driver = lime
9   label = LimeSDR Mini [USB 2.0] XXXXXXXXXXXXXXXX
10  media = USB 2.0
11  module = FT601
12  name = LimeSDR Mini
13  serial = XXXXXXXXXXXXXXXX

```

```

14      josh@testpi:~$ osmo-nitb --version
15      OpenBSC version 1.1.0
16      [...]
17
18      josh@testpi:~$ osmo-bts-trx --version
19      OsmoBTS version 0.8.1
20      [...]
21
22      josh@testpi:~$ osmo-trx-lms --version
23      OsmoTRX version 1.0.0.5-158e
24

```

Listing 5: Test all is working

In order to route phone calls, Asterisk and osmo-sip-connector need to be installed.

```

1  git clone git://git.osmocom.org/osmo-sip-connector.git
2  cd osmo-sip-connector/
3  autoreconf -fi
4  ./configure
5  make
6  sudo make install

```

Listing 6: Install osmo-sip-connector. Listing adapted from [here](#).

```

1  sudo apt install asterisk

```

Listing 7: Install asterisk

With all components now installed, the configuration needs to be obtained and updated. Sample configuration files are provided, and it is recommended that you read and understand them before use as they contain parameters such as the ARFCN which you will need to adjust to suit your licence.

```

1  # Download sample configuration
2  git clone https://github.com/uoscsa/gsm-conf conf
3
4  # Swap out Asterisk Configs
5  sudo apt install asterisk
6  sudo service asterisk stop
7  mv /etc/asterisk /etc/asterisk-old
8  mv ./conf/asterisk /etc/asterisk
9  sudo service asterisk start
10
11 # Check asterisk accepted new configuration
12 sudo service asterisk status
13 # Should be "active (running)"

```

Listing 8: Install configuration

With all components now installed, the laboratory can be run by following the instructions in [Appendix B](#).

## B Running the Laboratory

In order to run the laboratory, a number of components need to be started in a specific order. It is recommended to utilise a terminal multiplexer such as [tmux](#) to open several console windows simultaneously.

```

1  *** open terminal
2
3
4  #Start network controller
5  osmo-nitb -c ~/osmo/conf/openbsc.cfg -l ~/osmo/hlr.sqlite3 -P -C
6      --debug=DRLL:DCC:DMM:DRR:DRSL:DNM -M /tmp/bsc_mncc
7
8  #open new terminal
9  #Start layer 2/3
10 osmo-bts-trx -c ~/osmo/conf/osmo-bts.cfg
11
12 #open new terminal
13 #Start SIP connector
14 osmo-sip-connector -c ~/osmo/conf/osmo-sip-connector.cfg
15
16 #open new terminal
17 #Connect to asterisk service for remote monitoring
18 sudo asterisk -rvvvvvvvv
19
20 #open new terminal
21 #Run the SDR
22 osmo-trx-lms -C ~/osmo/conf/limesdr.cfg
23
24 #open new terminal
25 #Connect to osmo-nitb over telnet and enable client logging:
26 telnet localhost 4242
27 OpenBSC> enable
28 OpenBSC# logging enable
29 OpenBSC# logging level mm info

```

Listing 9: Start the base station

At this stage, you should be able to check your network output with another SDR as mentioned above, and see it operating within your licensed frequency band. Now, you will need to connect a mobile device to it.

In order to connect a mobile device to your network, you will usually need to manually “select” the network. The process to do this varies per device, and usually requires disabling ‘automatic’ network selection, scanning for networks and selecting your test network. The authors have found that some devices require rebooting before the setting takes effect; this may be because the test network is not synchronised to the output of other nearby base stations.

When the device attempts to connect, you should see output in both the **osmo-nitb** and **telnet** windows, containing the IMSI of the device attempting connection. It can now be authorised to connect by typing the following into the telnet session:

```

1  # telnet
2  OpenBSC> enable
3  OpenBSC# subscriber imsi xxxxxxxxxxxxxxxx authorized 1
4

```

Listing 10: Authorise Client

Now, when your client reboots and reconnects to the network, it should have successfully joined the BTS.

You can send a test SMS message using the following command:

```

1
2 # find subscriber we just authorised
3 OpenBSC> show subscriber imsi xxxxxxxxxxxxxxxx
4 # note down id.
5
6 OpenBSC> subscriber id X sms sender id 1 send Your Two-Factor
  Authentication Code is 12345678. Congratulations! You managed
  to sniff the GSM Protocol Over-The-Air.

```

Listing 11: Send SMS

A sample python script `sendsms.py` is included in the supplied `gsm-conf.zip` configuration sample repository; this sends SMS messages at a fixed interval.

Your mobile device should also be able to call extensions defined in the Asterisk `extensions.conf` including any SIP clients you connect, and the number **805** to receive a static two-factor authentication code by voice.

These two methods, SMS and voice call, should be initiated periodically during the laboratory for participants to capture.

## C Sample Captures

A sample set of captures from the test GSM network is supplied in the archive `gsm-captures.zip`. These two captures contain a SMS message and phone call, both of which can be decoded with `grgsm_decode` to allow the user to skip the live-network part of the laboratory. These could be particularly useful as a contingency if a live network environment is not possible.

To use these captures, simply remove references to `grgsm_livemon` in the lab notes and instead reference `grgsm_decode`.

## D 4G Installation

In this document, we utilised the Osmocom stack to create an example 2G phone network for experimentation.

A similar stack, named *srsLTE*, exists for experimentation with 4G networks. However, as explained above, the open-source reverse engineering tooling is not yet in a practical state to easily create packet captures and investigate the protocol with the same level of ease provided by `gr_gsm`.

Furthermore, the creation of a reliable software-defined 4G cell requires extra hardware if used with Raspberry Pi 4, as the Pi itself is unable to effectively handle the load of running the srsLTE physical layer software alongside the controller. As such, it is recommended to run the controller on an external x86/64 computer or virtual machine. Through experimentation, it was found that both components could be run on a Raspberry Pi 4 4GB with the CPU Governor set to maximum performance, but this is not a reliable configuration.

Alongside extra hardware, the 4G standard, unlike 2G, requires the base station and mobile device to authenticate each other as part of the join process. This requires that a SIM card with known cryptographic keys is placed in the handset

and the keys are also recorded in a database known to the controller software so that authentication of the network can take place. For 2G, using your own SIM is not a *technical* requirement, but to make LTE work you will have to obtain your own card such as the one from Sysmocom card detailed above.

The following installation guide can be used to install srsRAN , which contains srsEPC and srsENB (controller and eNodeB) which are required to operate a 4G cell.

On your Raspberry Pi, make sure you are running the **Ubuntu 20.04** OS and install srsRAN as per the guide above. This should result in you being able to run `srsepc` and `srsenb` without errors.

Now, make sure that you have the example configuration directory, which contains sample configs populated to run both the EPC and the ENB on the Raspberry Pi itself.

```
1  # Download sample configuration
2  git clone https://github.com/uoscsa/gsm-conf conf
3  cd ./conf/srslte
4  sudo cp enb.conf epc.conf user_db.csv /root/.config/srslte
```

Listing 12: Install configuration

You will need to edit `user_db.csv` to contain the details provided to you by Sysmocom at the time of purchase of your SIM cards.

Now, you should be able to run `srsepc` and `srsenb`, verify operation with another SDR, and authenticate a mobile device to the network by scanning and connecting.

Should you wish to change the network name as we did in our testing, it can be changed by editing `srsRAN/srsepc/src/mme/nas.cc`, line 1512.