Documentation

Software

Software List:

- UHD Drivers (B200)
  - https://files.ettus.com/manual/page_install.html
- srsLTE
  - https://github.com/srsLTE/srsLTE
- srsUE (Optional)
  - https://github.com/srsLTE/srsUE
- Asn1c
  - https://github.com/vlm/asn1c
- Project Code
  - https://github.com/bdub06/LTE

In order to get the USRP B200 mini working, download the drivers from the Ettus Research website.

srsLTE is the software suite that I used to sniff for pages over the air. It has multiple binaries that come pre built that allowed me to search for LTE devices within range and locate eNodeBs that are communicating with LTE devices in the area. *Cell_search* was used to locate LTE devices, and *pdsch_ue* was used to locate eNodeBs.

Example:

Search for devices that are on LTE band 4

```
Cell_search -b 4
```

Search for eNodeBs transmitting on X frequency and look for pages

```
Pdsch_ue -f XXXXX -r 0xFFFE
```

srsUE is a baseband implementation of LTE. This means that srsUE can turn the SDR into a UE device and capture LTE transmitted packets over the air. This suite was useful since captured packets would be written to a pcap file that can then be viewed in wireshark.

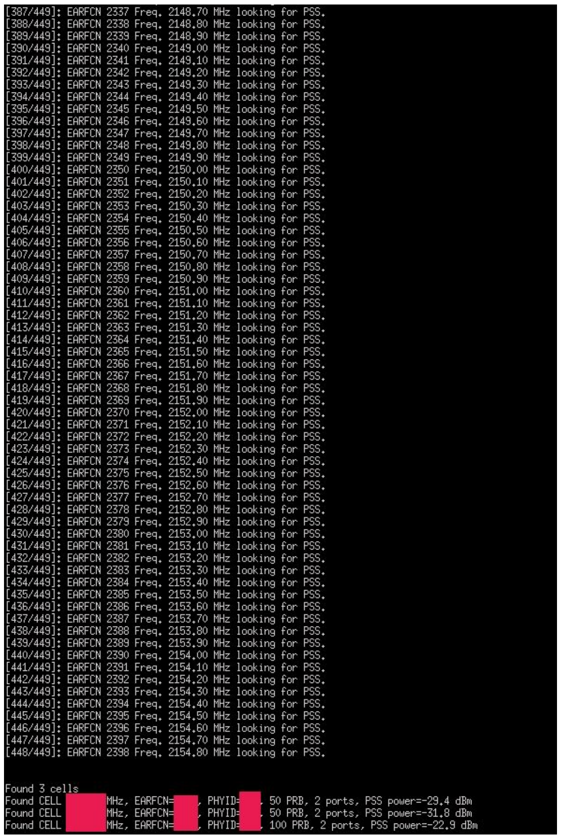
Asn1c is an ASN.1 to C converter. Inorder to make the page information human readable it was recommended to use ASN.1 to format the captured data from pdsch_ue. Asn1c was combined with srsLTE to streamline the page capture and collection of paging messages.

All software was installed and ran on an Ubuntu 16.04 LTS system on the 4.4.0-75-generic kernel.
Instructions on how to install and combine the software can be found in the README file in the git project.

Software Modifications

In order to streamline the capture process I had combined srsLTE with asn1c so that while srsLTE was capturing packets I could monitor the S-TMSIs in real time. Due to the lack of time I was not able to properly combine srsLTE and asn1c so that they could be compiled together. Directions on how to frankenstein the two libraries together can be found in the git repository README.

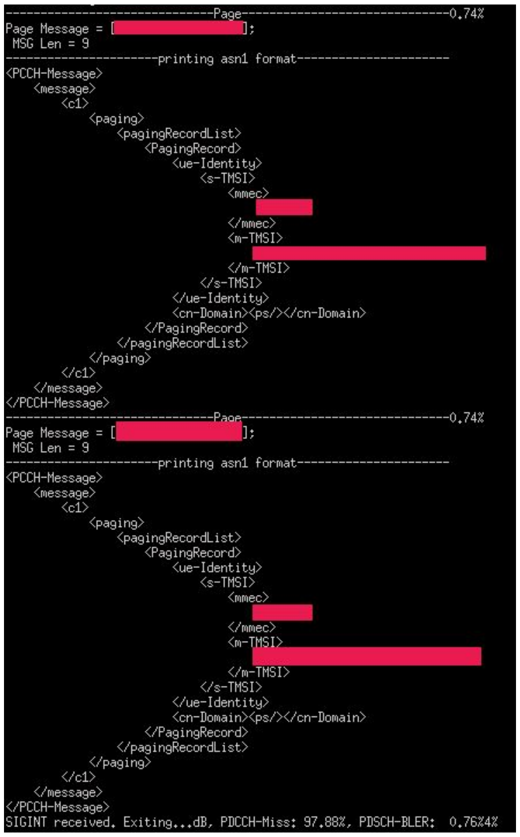
When the software is modified correctly the results of running the binaries should be similar to the images below.



```
[387/449]: EARFCN 2337 Freq. 2148.70 MHz looking for PSS.
[388/449]: EARFCN 2338 Freq. 2148.80 MHz looking for PSS.
[389/449]: EARFCN 2339 Freq. 2148.90 MHz looking for PSS.
[390/449]: EARFCN 2340 Freq. 2149.00 MHz looking for PSS.
[391/449]: EARFCN 2341 Freq. 2149.10 MHz looking for PSS.
[392/449]: EARFCN 2342 Freq. 2149.20 MHz looking for PSS.
[393/449]: EARFCN 2343 Freq. 2149.30 MHz looking for PSS.
[394/449]: EARFCN 2344 Freq. 2149.40 MHz looking for PSS.
[395/449]: EARFCN 2345 Freq. 2149.50 MHz looking for PSS.
[396/449]: EARFCN 2346 Freq. 2149.60 MHz looking for PSS.
[397/449]: EARFCN 2347 Freq. 2149.70 MHz looking for PSS.
[398/449]: EARFCN 2348 Freq. 2149.80 MHz looking for PSS.
[399/449]: EARFCN 2349 Freq. 2149.90 MHz looking for PSS.
[400/449]: EARFCN 2350 Freq. 2150.00 MHz looking for PSS.
[401/449]: EARFCN 2351 Freq. 2150.10 MHz looking for PSS.
[402/449]: EARFCN 2352 Freq. 2150.20 MHz looking for PSS.
[403/449]: EARFCN 2353 Freq. 2150.30 MHz looking for PSS.
[404/449]: EARFCN 2354 Freq. 2150.40 MHz looking for PSS.
[405/449]: EARFCN 2355 Freq. 2150.50 MHz looking for PSS.
[406/449]: EARFCN 2356 Freq. 2150.60 MHz looking for PSS.
[407/449]: EARFCN 2357 Freq. 2150.70 MHz looking for PSS.
[408/449]: EARFCN 2358 Freq. 2150.80 MHz looking for PSS.
[409/449]: EARFCN 2359 Freq. 2150.90 MHz looking for PSS.
[410/449]: EARFCN 2360 Freq. 2151.00 MHz looking for PSS.
[411/449]: EARFCN 2361 Freq. 2151.10 MHz looking for PSS.
[412/449]: EARFCN 2362 Freq. 2151.20 MHz looking for PSS.
[413/449]: EARFCN 2363 Freq. 2151.30 MHz looking for PSS.
[414/449]: EARFCN 2364 Freq. 2151.40 MHz looking for PSS.
[415/449]: EARFCN 2365 Freq. 2151.50 MHz looking for PSS.
[416/449]: EARFCN 2366 Freq. 2151.60 MHz looking for PSS.
[417/449]: EARFCN 2367 Freq. 2151.70 MHz looking for PSS.
[418/449]: EARFCN 2368 Freq. 2151.80 MHz looking for PSS.
[419/449]: EARFCN 2369 Freq. 2151.90 MHz looking for PSS.
[420/449]: EARFCN 2370 Freq. 2152.00 MHz looking for PSS.
[421/449]: EARFCN 2371 Freq. 2152.10 MHz looking for PSS.
[422/449]: EARFCN 2372 Freq. 2152.20 MHz looking for PSS.
[423/449]: EARFCN 2373 Freq. 2152.30 MHz looking for PSS.
[424/449]: EARFCN 2374 Freq. 2152.40 MHz looking for PSS.
[425/449]: EARFCN 2375 Freq. 2152.50 MHz looking for PSS.
[426/449]: EARFCN 2376 Freq. 2152.60 MHz looking for PSS.
[427/449]: EARFCN 2377 Freq. 2152.70 MHz looking for PSS.
[428/449]: EARFCN 2378 Freq. 2152.80 MHz looking for PSS.
[429/449]: EARFCN 2379 Freq. 2152.90 MHz looking for PSS.
[430/449]: EARFCN 2380 Freq. 2153.00 MHz looking for PSS.
[431/449]: EARFCN 2381 Freq. 2153.10 MHz looking for PSS.
[432/449]: EARFCN 2382 Freq. 2153.20 MHz looking for PSS.
[433/449]: EARFCN 2383 Freq. 2153.30 MHz looking for PSS.
[434/449]: EARFCN 2384 Freq. 2153.40 MHz looking for PSS.
[435/449]: EARFCN 2385 Freq. 2153.50 MHz looking for PSS.
[436/449]: EARFCN 2386 Freq. 2153.60 MHz looking for PSS.
[437/449]: EARFCN 2387 Freq. 2153.70 MHz looking for PSS.
[438/449]: EARFCN 2388 Freq. 2153.80 MHz looking for PSS.
[439/449]: EARFCN 2389 Freq. 2153.90 MHz looking for PSS.
[440/449]: EARFCN 2390 Freq. 2154.00 MHz looking for PSS.
[441/449]: EARFCN 2391 Freq. 2154.10 MHz looking for PSS.
[442/449]: EARFCN 2392 Freq. 2154.20 MHz looking for PSS.
[443/449]: EARFCN 2393 Freq. 2154.30 MHz looking for PSS.
[444/449]: EARFCN 2394 Freq. 2154.40 MHz looking for PSS.
[445/449]: EARFCN 2395 Freq. 2154.50 MHz looking for PSS.
[446/449]: EARFCN 2396 Freq. 2154.60 MHz looking for PSS.
[447/449]: EARFCN 2397 Freq. 2154.70 MHz looking for PSS.
[448/449]: EARFCN 2398 Freq. 2154.80 MHz looking for PSS.


Found 3 cells
Found CELL [redacted] MHz, EARFCN=[redacted], PHYID=[redacted], 50 PRB, 2 ports, PSS power=-29.4 dBm
Found CELL [redacted] MHz, EARFCN=[redacted], PHYID=[redacted], 50 PRB, 2 ports, PSS power=-31.8 dBm
Found CELL [redacted] MHz, EARFCN=[redacted], PHYID=[redacted], 100 PRB, 2 ports, PSS power=-22.9 dBm
```

Cell_search results



```
------------------------------Page------------------------------0.74%
Page Message = [[redacted]];
 MSG Len = 9
---------------------printing asn1 format---------------------
<PCCH-Message>
    <message>
        <c1>
            <paging>
                <pagingRecordList>
                    <PagingRecord>
                        <ue-Identity>
                            <s-TMSI>
                                <mmec>
                                    [redacted]
                                </mmec>
                                <m-TMSI>
                                    [redacted]
                                </m-TMSI>
                            </s-TMSI>
                        </ue-Identity>
                        <cn-Domain><ps/></cn-Domain>
                    </PagingRecord>
                </pagingRecordList>
            </paging>
        </c1>
    </message>
</PCCH-Message>
------------------------------Page------------------------------0.74%
Page Message = [[redacted]];
 MSG Len = 9
---------------------printing asn1 format---------------------
<PCCH-Message>
    <message>
        <c1>
            <paging>
                <pagingRecordList>
                    <PagingRecord>
                        <ue-Identity>
                            <s-TMSI>
                                <mmec>
                                    [redacted]
                                </mmec>
                                <m-TMSI>
                                    [redacted]
                                </m-TMSI>
                            </s-TMSI>
                        </ue-Identity>
                        <cn-Domain><ps/></cn-Domain>
                    </PagingRecord>
                </pagingRecordList>
            </paging>
        </c1>
    </message>
</PCCH-Message>
SIGINT received. Exiting...dB, PDCCH-Miss: 97.88%, PDSCH-BLER:  0.76%4%
```

pdsch_ue results

Resources:
https://github.com/nyee32/senior_project

Hardware

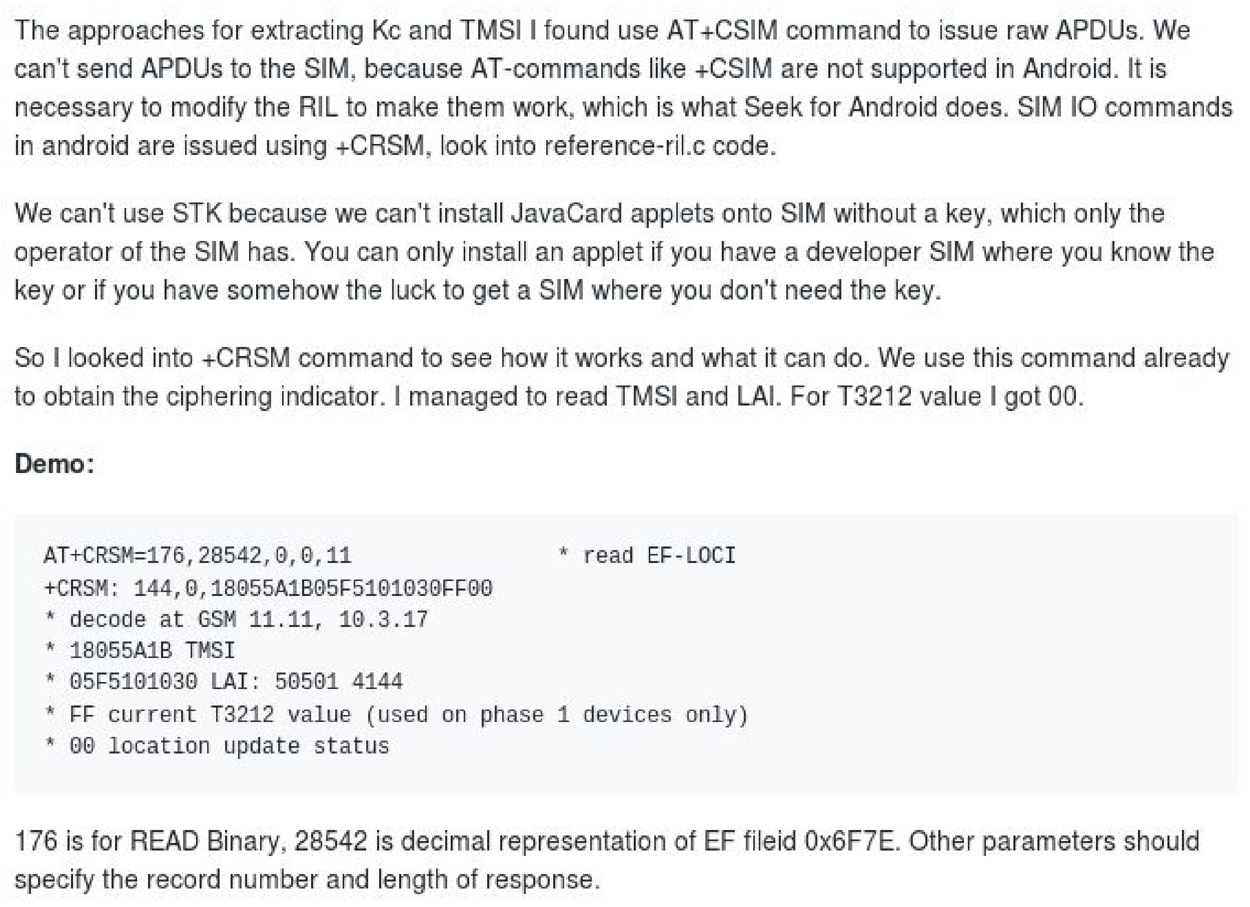
srsLTE supports a list of SDRs. A list can be found on their GitHub page. In this project I only used the ETTUS B200 mini along with the VERT900 antenna connected to the RX port of the B200. A USB 3.0 cable **must** be used with the B200 otherwise there will be sync errors.

The devices used was a rooted Nexus 5X with a Tracfone sim card that was connected to the AT&T network. I also installed the app LTE Discovery and an TMSI catcher detector called AIMSICD. I also used my personal T-Mobile sim card for reasons explained in the Locate device section.

Resources:
https://github.com/CellularPrivacy/Android-IMSI-Catcher-Detector/wiki
https://play.google.com/store/apps/details?id=net.simplyadvanced.ltediscovery&hl=en

Locate device

Due to lack of documentation and cell provider implementation, retrieving the GUTI value from a sim card is not trivial. I have tried to send serial commands to the sim card in order to try to get the TMSI and explore other sections of memory in the sim card. The steps I used are below.



The approaches for extracting Kc and TMSI I found use AT+CSIM command to issue raw APDUs. We can't send APDUs to the SIM, because AT-commands like +CSIM are not supported in Android. It is necessary to modify the RIL to make them work, which is what Seek for Android does. SIM IO commands in android are issued using +CRSM, look into reference-ril.c code.

We can't use STK because we can't install JavaCard applets onto SIM without a key, which only the operator of the SIM has. You can only install an applet if you have a developer SIM where you know the key or if you have somehow the luck to get a SIM where you don't need the key.

So I looked into +CRSM command to see how it works and what it can do. We use this command already to obtain the ciphering indicator. I managed to read TMSI and LAI. For T3212 value I got 00.

Demo:

```
AT+CRSM=176,28542,0,0,11               * read EF-LOCI
+CRSM: 144,0,18055A1B05F5101030FF00
* decode at GSM 11.11, 10.3.17
* 18055A1B TMSI
* 05F5101030 LAI: 50501 4144
* FF current T3212 value (used on phase 1 devices only)
* 00 location update status
```

176 is for READ Binary, 28542 is decimal representation of EF fileid 0x6F7E. Other parameters should specify the record number and length of response.

To be able to gain access to the sim card use the Android ADB emulator. Once ADB is active navigate to the dev directory. Depending on the device, the device will be listening to AT commands on 1 of the smd# files. To determine which smd file, use echo -c “AT\r” >/dev/smd# then cat the file to see if there is an OK response. If there is then this file will be used to send the AT commands described above, else send the AT commands to the next smd file until a OK response is returned.

Eventually I resorted to capturing the M-TMSI by sniffing for packets and while the SDR was sniffing I would make call to the Nexus 5X. I did this multiple times then compared all the captured M-TMSI for common identifiers. After identifying a few common M-TMSIs I would start the sniffing process and grep for the common M-TMSIs while making calls to the device to see if any of these M-TMSIs were paged while I was making the call.

Something I noticed using the Tracfone sim card was when the device was receiving a voice call or a text message the device would switch to using the UMTS band to establish the connection. This only happened multiple times and I was unable to determine when it would happen. Since srsLTE only listens for LTE pages, when the call was paged on UMTS, srsLTE never picked up the page. In this case I had to use my own sim card which as connect to the

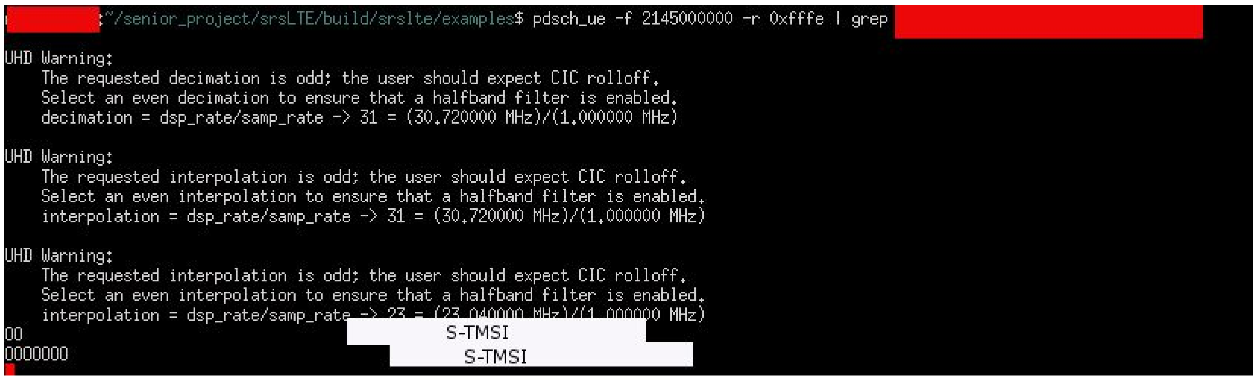
T-Mobile network. The T-Mobile sim card did not have the same behavior, so I was able to pick up M-TMSI of my own device.

Once the M-TMSI value has been identified, search for you can search for it while sniffing like using the command below.

Run pdsch_ue to sniff for pages then pipe it to fgrep to look for the M-TMSI

```
Pdsch_ue -f XXXXX -r 0xFFFE | fgrep -x -f <M-TMSI>
```

If the M-TMSI is found by pdsch_ue the results should look similar to the picture below.



```
[redacted]:~/senior_project/srsLTE/build/srslte/examples$ pdsch_ue -f 2145000000 -r 0xfffe | grep [redacted]

UHD Warning:
    The requested decimation is odd; the user should expect CIC rolloff.
    Select an even decimation to ensure that a halfband filter is enabled.
    decimation = dsp_rate/samp_rate -> 31 = (30.720000 MHz)/(1.000000 MHz)

UHD Warning:
    The requested interpolation is odd; the user should expect CIC rolloff.
    Select an even interpolation to ensure that a halfband filter is enabled.
    interpolation = dsp_rate/samp_rate -> 31 = (30.720000 MHz)/(1.000000 MHz)

UHD Warning:
    The requested interpolation is odd; the user should expect CIC rolloff.
    Select an even interpolation to ensure that a halfband filter is enabled.
    interpolation = dsp_rate/samp_rate -> 23 = (23.040000 MHz)/(1.000000 MHz)
00                                            S-TMSI [redacted]
0000000                                          S-TMSI [redacted]
[redacted]
```

Resources:
https://github.com/CellularPrivacy/Android-IMSI-Catcher-Detector/issues/96
*Look at andr3jx comment.
https://developer.android.com/studio/command-line/adb.html
http://stackoverflow.com/questions/8284067/send-at-commands-to-android-phone